SAPIENZA
UNIVERSITÀ DI ROMA

"Sapienza" University of Rome
Faculty of Information Engineering,
Informatics and Statistics
Department of Computer Science

Optimization

Lecture notes integrated with the book "Combinatorial Optimization",
W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, A. Schrijver

Author
Simone Bianco

March 6, 2024

Contents

Information and Contacts

Personal notes and summaries collected as part of the *Optimization* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
https://github.com/Exyss/university-notes. Anyone can feel free to report inaccuracies, improvements or requests through the Issue system provided by GitHub itself or by contacting the author privately:

- Email: **bianco.simone@outlook.it**
- LinkedIn: **Simone Bianco**

The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

Suggested prerequisites:

Preventive learning of material related to the *Algebra* course is recommended

Licence:

1 Graph flows

1.1 Networks and flows

Definition 1: Graph

A **graph** is a mathematical structure $G = (V, E)$, where V is the set of **vertices** in G and $E \subseteq V \times V$ is the set of **edges** that link two vertices in G.

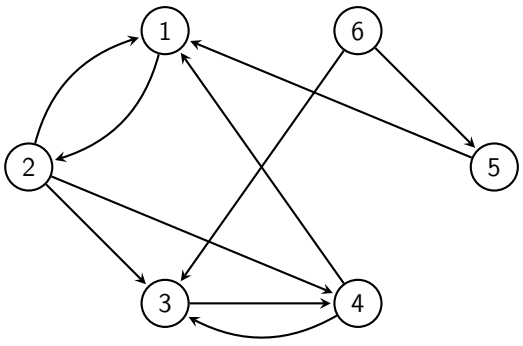
We will assume that each graph is *simple*, meaning that there are no multiple edges between the same nodes and no loops (that being an edge from a vertex to itself).

From now on, we will assume that $|V(G)| = n$ and $|E(G)| = m$.

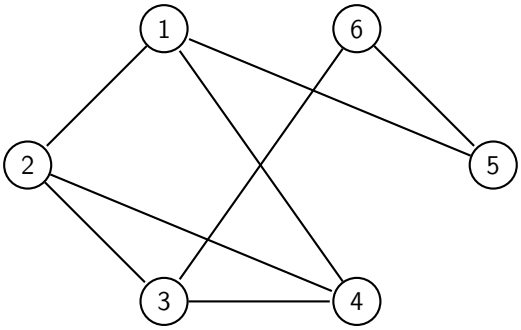
A graph can be **directed** or **undirected**. In a directed graph we consider the edges $(u, v) \in E(G)$ and $(v, u) \in E(G)$ as two different edges, while in an undirected graph they represent the same edge.

Example:

Directed graph



Undirected graph

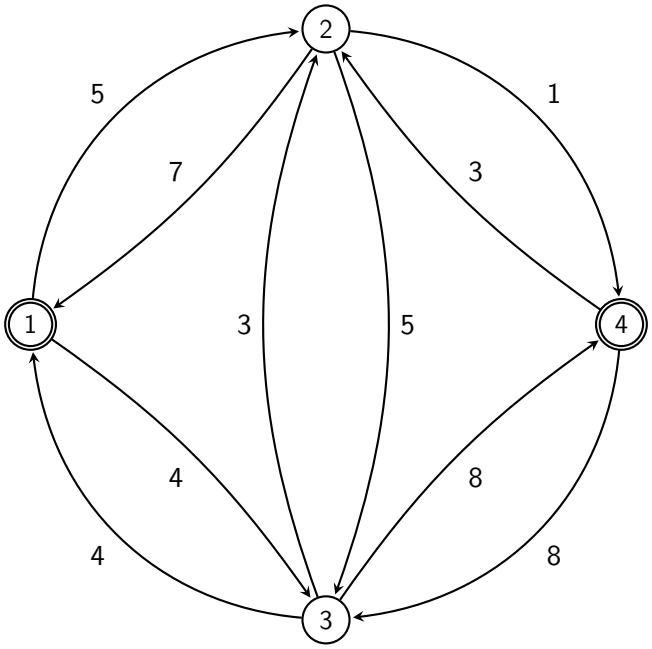


Definition 2: Network

A **network** is a mathematical structure $N = (G, s, t, c)$ where:

- $G = (V, E)$ is a directed graph
- s and t are two vertices of G ($s, t \in V(G)$), respectively called the **source** and the **sink**
- $c : E(G) \to \mathbb{R}^+$ is a weight function on the edges called **capacity**
- $(u, v) \in E(G) \implies (v, u) \in E(G)$

Example:



The numbers on the edges represent the capacities of the edges, while the nodes 1 and 4 are chosen respectively as the source and the sink of the network

Essentially, a network effectively describes a *water system* made up of *pipes* (the edges) that can transport a maximum amount of fluid inside them (the capacity of the edges). In fact, the last property of a network defines the idea of a bi-directional *flow of water* inside the pipes. In particular, we note that each pipe can have a different capacity based on the direction of the flow.

Definition 3: Flow

Given a network $N = (G, s, t, c)$, a **flow** is a weight function $f : E(G) \to \mathbb{R}$ on the edges defined by the following properties:

- **Skew-symmetric**: $\forall (u, v) \in E(G) \quad f(u, v) = -f(v, u)$, meaning that the incoming flow in an edge is the inverse of the outgoing flow in the corresponding opposite edge
- **Capacity bounded**: $\forall (u, v) \in E(G) \quad f(u, v) \leq c(u, v)$, meaning that the flow can't be greater than the supported capacity
- **Conservation of flow**: $\forall v \in V(G) - \{s, t\}$ it holds that

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u, v) = \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(v, w)$$

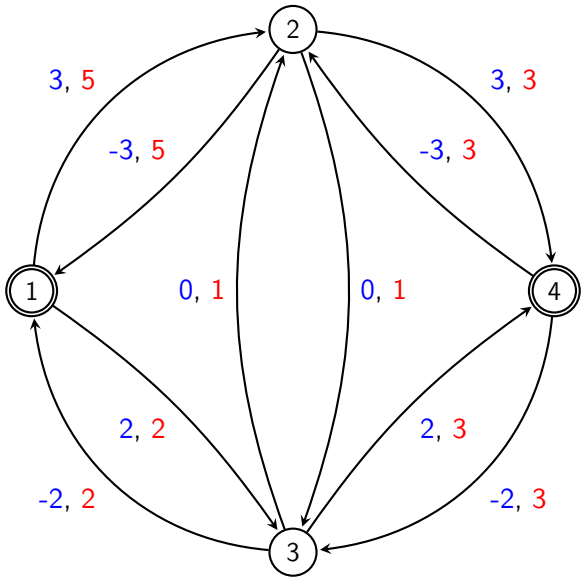
  meaning that the incoming flow in v is the same as the outgoing flow from v

Definition 4: Flow value

Given a network $N = (G, s, t, c)$ and a flow f on N, we define the **value of** f, noted by $\text{val}(f)$, as the sum of the flow outgoing from the source s or the sum of the flow incoming to the sink t:

$$\text{val}(f) := \sum_{(s,u)\in E(G)} f(s, u) = \sum_{(v,t)\in E(G)} f(v, t)$$

Example:



For each edge, the numbers in blue and red respectively represent its flow and its capacity. The flow value of the given flow is 5.

Observation 1: Nullification of flow for middle edges

Given a network $N = (G, s, t, c)$ and a flow f defined on G, for each vertex different from s and t it holds that:

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) = \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(v,w) \iff \sum_{(u,v)\in E(G)} f(u,v) = 0$$

Proof.

Due to the conservation of flow and the skew-symmetric properties, for all nodes $v \neq s, t$ we know that:

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) = \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(v,w) = \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} -f(w,v)$$

which is possible if only if:

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) = - \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(w,v) \iff$$

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) + \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(w,v) = 0$$

Again, by the skew-symmetric property we get that:

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) + \sum_{\substack{(v,w)\in E(G):\\ f(v,w)>0}} f(w,v) = 0 \iff$$

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) + \sum_{\substack{(w,v)\in E(G):\\ f(w,v)<0}} f(w,v) = 0$$

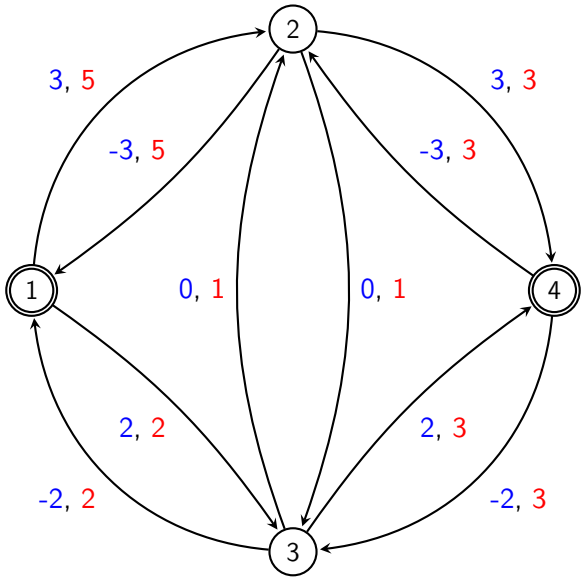
Then, by adding each edge incoming in v that has no flow we conclude that:

$$\sum_{\substack{(u,v)\in E(G):\\ f(u,v)>0}} f(u,v) + \sum_{\substack{(w,v)\in E(G):\\ f(w,v)<0}} f(w,v) + \sum_{\substack{(x,v)\in E(G):\\ f(x,v)=0}} f(x,v) = 0 \iff \sum_{(u,v)\in E(G)} f(u,v) = 0$$

□

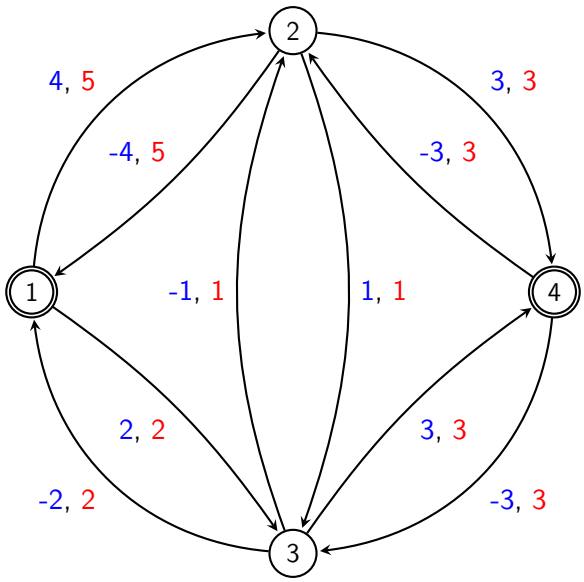
1.2 Residual graphs, flow increase and st-cuts

Consider the network shown in the last example of the previous section.



We notice that some *pipes* aren't completely *"filled up"*, meaning that their capacity could support a bigger flow. In particular, due to conservation of flow, not all pipes can be filled to the maximum capacity. In fact, we know that flow outgoing from the source must still be equal to the incoming flow of the sink.

Thus, we can increase the flow value by 1 only by using the remaining capacities in the path $1 \to 2 \to 3 \to 4$



The flow value of the new flow is 6.

Definition 5: Residual capacity

Given a network $N = (G, s, t, c)$ and a flow f on N, the **residual capacity** is a weight function $r : E(G) \to \mathbb{R}^+$ defined as:

$$r(u, v) = c(u, v) - f(u, v)$$

Observation 2

Given a network $N = (G, s, t, c)$ and a flow f on N, we note that:

$$r(u, v) + r(v, u) = c(u, v) + c(v, u)$$

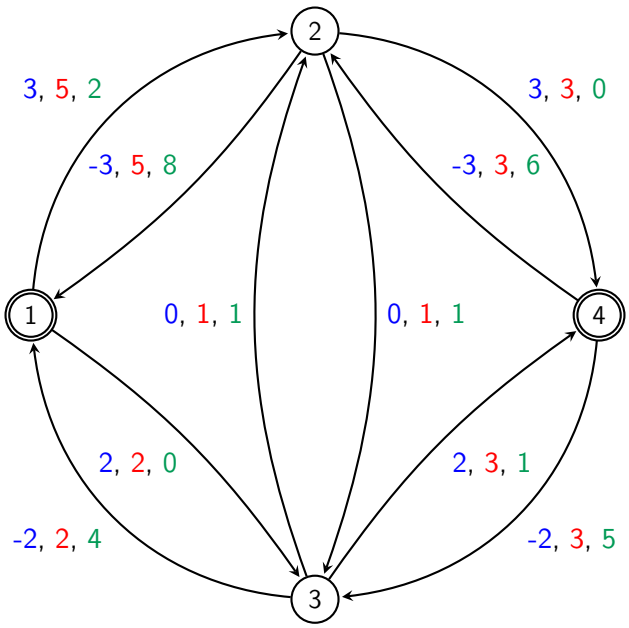
Definition 6: Residual graph

Given a network $N = (G, s, t, c)$ and a flow f on N, we define $R \subseteq G$ as the **residual graph** of G if:

$$(u, v) \in E(R) \iff r(u, v) > 0$$

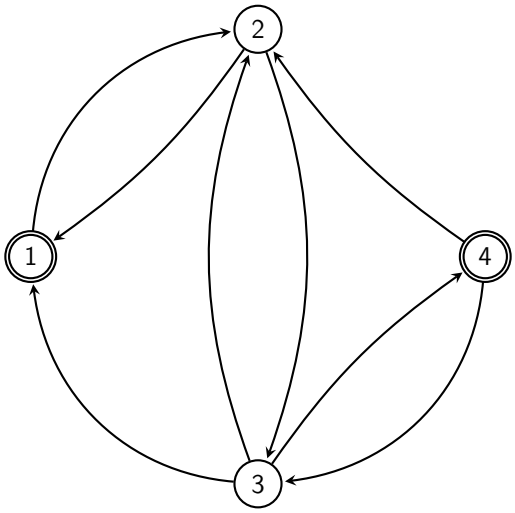
Example:

Consider the first graph previously shown with flow value 5. We now add the residual capacities obtained through that flow.



For each edge, the number in green represents its residual capacity

Thus, the residual graph is the following:



Proposition 1: Flow-augmenting path

Given a network $N = (G, s, t, c)$ and a flow f on N, let $R \subseteq G$ be the residual graph of G on f and let P be a direct path $s \to t$ in R.

Given $\alpha := \min_{(u,v)\in E(P)} r(u, v)$, we define $f' : E(G) \to R$ as:

$$f'(u,v) = \begin{cases} f(u,v) + \alpha & \text{if } (u,v) \in E(P) \\ f(u,v) - \alpha & \text{if } (v,u) \in E(P) \\ f(u,v) & \text{otherwise} \end{cases}$$

The function f' is a flow for N such that $\text{val}(f') = \text{val}(f) + \alpha$.

Proof.

Suppose that $(u, v) \in E(P)$:

- By using the skew-symmetric property of f, we get that:

$$f'(u,v) = f(u,v) + \alpha \implies -f'(u,v) = -f(u,v) - \alpha = f(v,u) - \alpha$$

  Also, since $(u, v) \in E(P)$, for (v, u) we get that

$$f'(v,u) = f(v,u) - \alpha = -f'(u,v)$$

- By assumption, we know that $f'(u, v) = f(u, v) + \alpha$. Thus, by definition of α we get that:

$$\alpha \leq r(u,v) = c(u,v) - f(u,v) \implies$$
$$f'(u,v) = f(u,v) + \alpha \leq f(u,v) + c(u,v) - f(u,v) = c(u,v)$$

If $(v, u) \in E(P)$, we can get the same result by repeating the same steps. Otherwise, the flow remains unchanged, meaning that the property is already satisfied. Thus, we conclude that f' is *skew-symmetric* and *capacity bounded*.

Given a vertex $x \in E(G)$, if $x \notin E(P)$ then the flos of its edges are unchanged. If $x \in E(P)$, we procede by splitting the edges in G through P:

$$\sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0}} f'(u,x) = \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (u,x)\in E(P)}} f'(u,x) + \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (x,u)\in E(P)}} f'(u,x) =$$

$$\sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (u,x)\in E(P)}} [f(u,x)+\alpha] + \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (x,u)\in E(P)}} [f(u,x)-\alpha]$$

We notice that the amount of vertices in these sums is the same, implying that each time α gets summed in the first sum it also gets subtracted from the other sum:

$$\sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (u,x)\in E(P)}} [f(u,x)+\alpha] + \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (x,u)\in E(P)}} [f(u,x)-\alpha] =$$

$$\sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (u,x)\in E(P)}} f(u,x) + \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0,\\ (x,u)\in E(P)}} f(u,x) = \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0}} f(u,x)$$

By using the same argument, we get that:

$$\sum_{\substack{(x,w)\in E(G):\\ f'(x,w)>0}} f'(x,w) = \sum_{\substack{(x,w)\in E(G):\\ f'(x,w)>0}} f(x,w)$$

Finally, through the the *conservation of flow* of f, we conclude that f' also satisfies such property:

$$\sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0}} f'(u,x) = \sum_{\substack{(u,x)\in E(G):\\ f'(u,x)>0}} f(u,x) = \sum_{\substack{(x,w)\in E(G):\\ f'(x,w)>0}} f(x,w) = \sum_{\substack{(x,w)\in E(G):\\ f'(x,w)>0}} f'(x,w)$$

□

Definition 7: $st-$cut

Given a network $N = (G, s, t, c)$ and a flow f on N, we define a ***st*-cut of** G as a subset $\mathcal{U} \subseteq V(G)$ that makes a partition on $V(G)$ such that $s \in \mathcal{U}, t \notin \mathcal{U}$.

Additionally, the vertices inside of $\mathcal{U}$ are called the *s-part* of the cut, while the vertices outside of $\mathcal{U}$ are called the *t-part* of the cut.

Proposition 2: Flow of an st-cut

Given a network $N = (G, s, t, c)$, a flow f on N and an st-cut $\mathcal{U} \subseteq G$, we have that:

$$\text{val}(f) = \sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}, v\notin\mathcal{U}}} f(u, v)$$

Proof.

Consider the total sum of the flows of all the vertices in $\mathcal{U}$:

$$\sum_{x\in\mathcal{U}} \sum_{(u,v)\in E(G)} f(u, v)$$

By definition, we know that $s \in \mathcal{U}$. We can separate the flows outgoing from s from the rest of the flows:

$$\sum_{x\in\mathcal{U}} \sum_{(u,v)\in E(G)} f(u, v) = \sum_{(s,w)\in E(G)} f(s, w) + \sum_{x\in\mathcal{U}-\{s\}} \sum_{(u,v)\in E(G)} f(u, v)$$

Due to the Nullification of flow for middle edges, for all vertices $u, v \neq s$ we know that the total flow is qual to 0, implying that:

$$\sum_{(s,w)\in E(G)} f(s, w) + \sum_{x\in\mathcal{U}-\{s\}} \sum_{(u,v)\in E(G)} f(u, v) = \sum_{(s,w)\in E(G)} f(s, w) + 0 = \text{val}(f)$$

We now consider again the initial total sum. We notice that:

$$\sum_{x\in\mathcal{U}} \sum_{(u,v)\in E(G)} f(u, v) = \sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}}} f(u, v)$$

Additionally, for any (u, v) if $u, v \in \mathcal{U}$ then $f(u, v)$ cancels out with $f(v, u)$, implying that:

$$\sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}}} f(u, v) = \sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}, v\notin\mathcal{U}}} f(u, v)$$

By combining the shown equalities, we conclude the proof.

□

Definition 8: Capacity of an st-cut

Given a network $N = (G, s, t, c)$, a flow f on N and an st-cut $\mathcal{U} \subseteq G$, we define the **capacity of an st− cut on** G, noted by $c(V(G)\backslash\mathcal{U})$, as the sum of the capacities of the edges outgoing from *s-part* to the *t-part*.

$$c(V(G)\backslash\mathcal{U}) := \sum_{\substack{(u,v)\in E(G):\\ x\in\mathcal{U}, u\notin\mathcal{U}}} c(u,v)$$

Lemma 1

Given a network $N = (G, s, t, c)$, the maximum value of a flow on G at most the minimal capacity of an st-cut on G:

$$\max_{f\,:\,\text{flow on } G}(\text{val}(f)) \leq \min_{\mathcal{U}\,:\,st-\text{cut on } G}(c(V(G)\backslash\mathcal{U}))$$

Proof.

Let f be the flow on G that maximizes $\text{val}(f)$ and let $\mathcal{U} \subseteq G$ be the st-cut on G that minimizes capacity. By the Flow of an st-cut and by the *capacity bounded* property of f, we get that:

$$\text{val}(f) = \sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}, v\notin\mathcal{U}}} f(u,v) \leq \sum_{\substack{(u,v)\in E(G):\\ u\in\mathcal{U}, v\notin\mathcal{U}}} c(u,v) = c(V(G)\backslash\mathcal{U})$$

□

1.3 The Ford-Fulkerson algorithm

Algorithm 1: The Ford-Fulkerson algorithm

Given a network $N = (G, s, t, c)$, we define the following algorithm:

function FordFulkerson(G)
  Start with the trivial flow f with all 0s
  while True **do**
    Compute the residual graph $R \subseteq G$ on flow f
    Find a path P in R from $s \to t$
    if P does not exist **then**
      Return f
    else
      Increase f through the value α obtained by P
    end if
  end while
end function

We note that the FordFulkerson(N) terminates only if the augment value eventually becomes 0, implying that there is no flow-augmenting path to be used. Thus, if the capacities defined by c are all integers, the algorithm always terminates.

Moreover, the flow-augmenting path of each iteration of FordFulkerson(N) can be found with a simple DFS search, requiring $O(n+m)$, which is also the cost needed for computing the residual graph of each iteration. Thus, the **computational complexity** of the algorithm is $O(k(n + m))$, where k is the maximum number of iterations of the while loop.

It's easy to notice that this value k **depends too much on the shape of the graph** G. However, due to the *capacity bounded* property, in the worst case we have that k equals the maximum flow value. We also notice that, technically, the cost of this algorithm is **exponential**: the number of bits required to store k are $\log_2 k$, but the cost relies on $2^{\log_2 k} = k$ iterations.

Lemma 2

Given a network $N = (G, s, t, c)$, if the algorithm FordFulkerson(N) terminates then it returns a flow f such that there exists an st-cut $\mathcal{U} \subseteq G$ for which we have that $\text{val}(f) = c(V(G) \backslash \mathcal{U})$

Proof.

Let R be the residual graph of G on f = FordFulkerson(N) and let r be the residual capacity function obtained through f. We define $\mathcal{U} \subseteq G$ as:

$$\mathcal{U} = \{x \in V(G) \mid \exists s \to x \text{ in } G'\}$$

Since the algorithm terminates when there is no path from $s \to t$, we know that $t \notin \mathcal{U}$, implying that $\mathcal{U}$ is an st-cut of G. Thus, by the Flow of an st-cut, we have that:

$$\text{val}(f) = \sum_{\substack{(x,v)\in E(G):\\ x\in\mathcal{U}, v\notin\mathcal{U}}} f(x,u)$$

By way of contradiction, we suppose that $\exists (x,v) \in E(G') \subseteq E(G)$ such that $x \in \mathcal{U}, v \notin \mathcal{U}$. By definition of $\mathcal{U}$, we know that $s \to x$ in G', so by adding (x,v) to the path we get that $s \to x \to y$, which contradicts $y \notin \mathcal{U}$.

Thus, such edge can't exists, implying that $\forall (x,v) \in E(G)$ such that $x \in \mathcal{U}, v \notin \mathcal{U}$ it holds that $(x,v) \notin E(G')$, which by definition of residue graph implies that:

$$\forall (x,v) \in E(G) \text{ s.t. } x \in \mathcal{U}, v \notin \mathcal{U} \quad f(x,v) = c(x,v)$$

concluding that:

$$\text{val}(f) = \sum_{\substack{(x,v)\in E(G):\\ x\in\mathcal{U}, v\notin\mathcal{U}}} f(x,u) = \sum_{\substack{(x,v)\in E(G):\\ x\in\mathcal{U}, v\notin\mathcal{U}}} c(x,u) = c(V(G)\backslash\mathcal{U})$$

□

1.3.1 The max flow, min cut theorem

Theorem 1: Max flow, min cut theorem

Given a network $N = (G, s, t, c)$, the maximum value of a flow on G at most the minimal capacity of an st-cut on G:

$$\max_{f\,:\,\text{flow on } G}(\text{val}(f)) = \min_{\mathcal{U}\,:\,st-\text{cut on } G}(c(V(G)\backslash\mathcal{U}))$$

Proof.

By the Lemma 1, we already know that:

$$\max_{f\,:\,\text{flow on } G}(\text{val}(f)) \leq \min_{\mathcal{U}\,:\,st-\text{cut on } G}(c(V(G)\backslash\mathcal{U}))$$

We now consider $f' = \mathsf{FordFulkerson}(N)$. By the Lemma 2, we know that there exists an st-cut $\mathcal{U}' \subseteq G$ such that $\text{val}(f') = c(V(G)\backslash\mathcal{U}')$.

Thus, it's easy to conclude that:

$$\max_{f\,:\,\text{flow on } G}(\text{val}(f)) \geq \text{val}(f') = c(V(G)\backslash\mathcal{U}) \geq \min_{\mathcal{U}\,:\,st-\text{cut on } G}(c(V(G)\backslash\mathcal{U}))$$

□

Corollary 1: Optimality of Ford-Fulkerson

The Ford-Fulkerson algorithm returns a flow with **maximum value**

1.4 The Edmonds-Karp algorithm

Algorithm 2: The Edmonds-Karp algorithm

Given a network $N = (G, s, t, c)$, we define the following algorithm:

function EDMONDSKARP(N)
  Start with the trivial flow f with all 0s
  while True **do**
    Compute the residual graph $R \subseteq G$ on flow f
    Find the shortest path P in R from $s \to t$
    if P does not exist **then**
      Return f
    else
      Increase f through the value α obtained by P
    end if
  end while
end function

Note: the shortest path is the one with the fewest amount of edges.

Lemma 3

Given a network $N = (G, s, t, c)$, if the algorithm EdmondsKarp(N) terminates then it returns a flow f such that there exists an st-cut $\mathcal{U} \subseteq G$ for which we have that $\text{val}(f) = c(V(G) \backslash \mathcal{U})$

(*proof identical to* Lemma 2)

Corollary 2: Optimality of Edmonds-Karp

The Edmonds-Karp algorithm returns a flow with **maximum value**

The Edmonds-Keep algorithm looks identical to the Ford-Fulkerson algorithm, except for the type of path searched at each iteration. The idea behind finding the shortest path instead of a random path is to **limit** the number of iterations made by the algorithm by making them rely on the number of nodes instead of the maximum flow value. Thus, the **computational complexity** of the algorithm effectively becomes $O(mn(n + m))$, meaning that it's not an exponential algorithm. The following statements will be necessary to justify this result.

Observation 3

Given a graph G and two vertices $x, y \in V(G)$, let P be the shortest path from $x \to y$. Then, for each node $z_i \in V(P)$ the path P contains the shortest path P_i from $x \to z_i$

Proof.

For each node $z_1, \dots, z_k \in V(P)$ (x and y included), let P_i be the shortest path from $x \to z_i$. By way of contradiction, suppose that $P_i \not\subseteq P$.

Given an index $i \in [1, k]$, let $P_x, P_y \subseteq P$ be the sub-paths that partition P by z_i, meaning that $x, z_1, \dots, z_i \in V(P_x)$ and $z_{i+1}, \dots, z_k, y \in V(P_y)$.

Since by assumption P_i is shorter that P_x, we get that the path $P_i \cup P_y$ is a path from $x \to y$ that is shorter than P, which is a contradiction. Thus, we conclude that for each node $z_1, \dots, z_k \in V(P)$ it holds that $P_i \subseteq P$.

□

Proposition 3: Disappearing and appearing edges

Given a network $N = (G, s, t, c)$ and the computation $\mathsf{EdmondsKarp}(N)$, let:

- $f_0, \dots, f_k$ be the series of flows computed, where f_0 is the trivial flow
- $R_0, \dots, R_k$ be the series of residue graphs computed, where R_0 is obtained with flow f_i
- $P_0, \dots, P_k$ be the series of shortest paths from $s \to t$ computed on R_i

Then, $\forall u \in V(G)$ and $\forall i \in [1, k]$ it holds that:

$$(u, v) \in E(R_i), (u, v) \notin E(R_{i+1}) \implies (u, v) \in E(P_i)$$

and that:

$$(u, v) \notin E(R_i), (u, v) \in E(R_{i+1}) \implies (v, u) \in E(P_i)$$

Proof.

If $(x, y) \in E(R_i)$ but $(x, y) \notin E(R_{i+1})$, the edge got removed by the flow-increase of path P_i. This can only happen only if $c(x, y) = f_{i+1}(x, y) = f_i(x, y) + \alpha_i$, where α_i is the flow increase obtained by P_i, meaning that $(x, y) \in P_i$.

Instead, if $(x, y) \notin E(R_i)$ but $(x, y) \in E(R_{i+1})$, the edge got added by the flow-increase of path P_i. This can happen only if $f_i(x, y) \geq f_{i+1}(y, x)$, which in turn can happen only if the flow of the opposite edge (x, y) got increased, meaning that $(y, x) \in E(P_i)$.

□

Lemma 4: Monotone increasing distance in Edmonds-Keep

Given a network $N = (G, s, t, c)$ and the computation $\mathsf{EdmondsKarp}(N)$, let:

- $f_0, \ldots, f_k$ be the series of flows computed, where f_0 is the trivial flow
- $R_0, \ldots, R_k$ be the series of residue graphs computed, where R_0 is obtained with flow f_i
- $P_i, \ldots, P_k$ be the series of shortest paths from $s \to t$ computed, where $P_i \subseteq R_i$

Then, $\forall u \in V(G)$ and $\forall i \in [1, k]$ it holds that:

$$\text{dist}_{R_i}(s, u) \leq \text{dist}_{R_{i+1}}(s, u)$$

Proof.

By way of contradiction, we assume that there exist some vertices for which the statement doesnt hold, meaning that $\exists u_1, \ldots, u_k \in V(G)$ such that $\text{dist}_{R_i}(s, u_j) > \text{dist}_{R_{i+1}}(s, u_j)$.

Let v be the vertex picked from $u_1, \ldots, u_k$ with minimal distance from s in R_i, meaning that:

$$\text{dist}_{R_i}(s, v) = \min_{j=1}^{k} \text{dist}_{R_i}(s, u_j)$$

Consider the shortest path P in R_{i+1} from $s \to v$, that being the path with length $\text{dist}_{R_{i+1}}(s, v)$. Let $u \in V(P)$ be the vertex that precedes v in P, meaning that $(u, v) \in E(P)$.

Since v is the vertex from $u_1, \ldots, u_k$ with the shortest distance and since $\text{dist}_{R_{i+1}}(s, u) = \text{dist}_{R_{i+1}}(s, v) - 1$ due to it being the previous vertex of v in P, it must hold that $u \notin \{u_1, \ldots, u_k\}$ because otherwise u would be the one with the shortest distance.

Thus, we get that $\text{dist}_{R_i}(s, u) \leq \text{dist}_{R_{i+1}}(s, u)$, which implies that:

$$\text{dist}_{R_i}(s, u) \leq \text{dist}_{R_{i+1}}(s, u) = \text{dist}_{R_{i+1}}(s, v) - 1$$

Suppose now that $(u, v) \in E(R_i)$. Given the shortest path P' in $R_i i$ from $s \to u$ can be extended with (u, v), we get that $\text{dist}_{R_i i}(s, v) \leq \text{dist}_{R_i i}(s, u) + 1$. However, this would imply that:

$$\text{dist}_{R_i}(s, v) \leq \text{dist}_{R_i}(s, u) + 1 \leq \text{dist}_{R_{i+1}}(s, u) + 1 \leq= \text{dist}_{R_{i+1}}(s, v)$$

which contradicts the initial assumption. Thus, it can't be that $(u, v) \in E(R_i)$.

Moreover, since $(u, v) \in E(P) \subseteq E(R_{i+1})$ and $(u, v) \notin E(R_i)$, by Proposition 3 we know that:

$$(u, v) \notin E(R_i), (u, v) \in E(R_{i+1}) \implies (v, u) \in E(P_i)$$

Since P_i is the shortest path $s \to t$ in R_i and $u, v \in V(P_i)$, by Observation 3 we know that P_i also contains the shortest paths from $s \to u$ and $s \to v$ in R_i. In particular, the path from $s \to u$ also contains the path $s \to v$, so $\text{dist}_{R_i}(s, v) = \text{dist}_{R_i}(s, u) - 1$.

Combining this with the previous results and the initial assumption, we get that:

$$\text{dist}_{R_i}(s, v) = \text{dist}_{R_i}(s, u) - 1 \leq \text{dist}_{R_{i+1}}(s, u) - 1 = \text{dist}_{R_{i+1}}(s, v) - 2 < \text{dist}_{R_i}(s, v) - 2$$

implying that $0 < -2$, which is impossible, meaning that such vertices $u_1, \ldots, u_k$ can't exists.

□

Theorem 2: Total iterations of Edmonds-Karp

The total number of iterations done by the Edmonds-Karp algorithm is $O(mn)$

Proof.

Given a network $N = (G, s, t, c)$ and the computation $\mathsf{EdmondsKarp}(N)$, let:

- $f_0, \ldots, f_k$ be the series of flows computed, where f_0 is the trivial flow
- $R_0, \ldots, R_k$ be the series of residue graphs computed, where R_0 is obtained with flow f_i
- $P_i, \ldots, P_k$ be the series of shortest paths from $s \to t$ computed, where $P_i \subseteq R_i$

In the following steps, we define an edge (u, v) as *critical* for R_i if $(u, v) \in E(R_i)$ but $(u, v) \notin E(R_{i+1})$. In particular, by Proposition 3 we know that if an edge is critical for R_i then $(u, v) \in P_i$. Moreover, for each shortest path P_i we know that there is at least one critical edge inside it, that being the edge which defines the flow augmentation value α_i for P_i.

Given an edge $(u, v) \in E(G)$, let $\pi(1) < \pi(2) < \ldots < \pi(\ell) \in [1, k]$ be the indices such that (u, v) is critical for $\pi(i)$. By Observation 3, we know that $\forall i \in [1, k]$ it holds that:

$$(u, v) \in E(P_{\pi(i)}) \implies \text{dist}_{R_{\pi(i)}}(s, v) = \text{dist}_{R_{\pi(i)+1}}(s, u) + 1$$

meaning that $(u, v) \in E(R_{\pi(i)+1})$.

If (u, v) is critical for both $R_{\pi(i)}$ and $R_{\pi(j)}$ (where $i < j$, meaning that (u, v) disappears both times), then there must be an index h such that $\pi(i) < h < \pi(j)$ where (u, v) reappears, meaning that $(u, v) \notin E(G_h)$ and $h \in E(G_{h+1})$.

Again, by Proposition 3, we know that:

$$(u, v) \notin E(G_h), h \in E(G_{h+1}) \implies (v, u) \in E(P_h) \implies \text{dist}_{R_h}(s, u) = \text{dist}_{R_h}(s, v) + 1$$

Then, since $\pi(i) < h < \pi(j)$, by the Monotone increasing distance in Edmonds-Keep we know that:

$$\text{dist}_{R_{\pi(j)}}(s, u) \geq \text{dist}_{R_h}(s, u) = \text{dist}_{R_h}(s, v) + 1 \geq \text{dist}_{R_{\pi(i)}}(s, u) + 2$$

Thus, between $R_{\pi(i)}$ and $R_{\pi(j)}$ the vertex u's distance increases by at least 2. Thus, since u's final distance can be at most $n-1$, meaning that $\text{dist}_{R_k}(s,u) \leq n$, starting from 0 the distance can be increased by 2 at most $\frac{n}{2}$ times, meaning that $\ell = \frac{n}{2}$.

Since each flow-augmenting shortest path computed by the algorithm implies the existence of a a critical path and since since the number of times each edge can be critical is at most $\ell = \frac{n}{2}$, the number of paths computable is at most $m \times \frac{n}{2}$, which is in $O(mn)$.

□