Algorithms based on augmentations: Flows

Peter Hajnal

Bolyai Institute, University of Szeged, Hungary

2024 fall

Basic notions

Definition: Network

Let $\overrightarrow{G}$ be a directed graph, $s, t \in V(G)$ two distinct distinguished vertices (called source and sink), and $c : E(G) \to \mathbb{R}_{++}$ capacity function. The $(\overrightarrow{G}, s, t, c)$ quadruple is called *network*.

Definition: Flow (in network)

The function $f : E(G) \to \mathbb{R}$ is a *flow* in the network H, if

(F1) for each edge e we have $\quad 0 \leq f(e) \leq c(e)$

(F2) for each $v \in V \setminus \{s, t\}$ we have $\sum_{e:e\in E_{in}(v)} f(e) = \sum_{e:e\in E_{out}(v)} f(e)$, where $E_{in}(x)$ is the set of ingoing edges of x, $E_{out}(x)$ is the set of outgoing edges of x.

(F1) is called capacity constrain. (F2) is called flow preservation.

Comparing flows in network

Example

The function $f \equiv 0$ is a flow in any network.

Definition: The value of a flow

$\mathrm{v}(f) = \sum_{e \in E_{in}(t)} f(e) - \sum_{e \in E_{out}(t)} f(e)$.

The flow problem

Definition: The flow problem

Given a network, find a maximum value flow in it.

We said maximum value. Is it correct? YES.

The flow $f : E(G) \to \mathbb{R}_+$ can be considered as a vector $\overrightarrow{f} \in \mathbb{R}^{E(G)}$. The capacity constrain and flow preservation low define a compact subset of $\mathbb{R}^{E(G)}$. The value is a continuous function over this compact domain. The maximum value exists based on your calculus courses from BSc.

The flow problem is a special case of linear programming (LP).

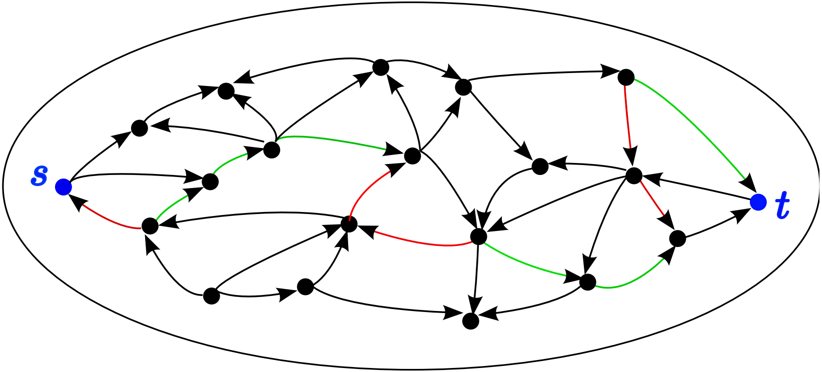
Initial remarks

Let P be an st-path (not necessarily directed) in $\overrightarrow{G}$. I.e. We delete the orientations of the edges, hence obtain an undirected graph G. P is a path in G.

We can classify the edges of P into two categories: As we walk along P from the source to the sink we might follow the orientation of a directed edge of $\overrightarrow{G}$, or we traverse it in the opposite direction. An edge of P can be a forward or a backward edge.

$E(P) = E_{\text{forward}}(P) \bigcup E_{\text{backward}}(P)$.

Example



Augmenting paths

Definition

Let $H = (\overrightarrow{G}, s, t, c)$ be a network. P is *augmenting path* for f (or simply f-augmenting path) if P is a path in G s.t.

(A1) P starts at the source, s.

(A2) P ends at the sink, t.

(A3) For each edge $e \in E_{\text{forward}}(P)$ we have $f(e) < c(e)$, for each $e \in E_{\text{backward}}(P)$ $f(e) > 0$.

Why augmenting?

Lemma

Let f be a flow in the network $H = (\overrightarrow{G}, s, t, c)$, and P be an f-augmenting path. Then the flow f is non-optimal. I.e. there is a flow f^+, that has greater value than f.

Proof

Introduce a few parameter

- $\delta_{\text{forward}} := \min_{e \in E_{\text{forward}}(P)}(c(e) - f(e))$.
- $\delta_{\text{backward}} := \min_{e \in E_{\text{backward}}} f(e)$,
- $\delta := \min\{\delta_{\text{backward}}, \delta_{\text{forward}}\}$.
- If P is an augmenting path, then $\delta > 0$.
- Now we can describe the improved flow:

$$f^+(e) = \begin{cases} f(e), & e \notin E(P), \\ f(e) + \delta, & e \in E_{\text{forward}}(P), \\ f(e) - \delta, & e \in E_{\text{backward}}(P). \end{cases}$$

Proof (cont'd)

We need some observations:

(1) $\delta > 0$.

(2) f^+ obeys the capacity constrain,

(3) f^+ obeys the flow preservation low,

(4) $\mathrm{v}(f^+) = \mathrm{v}(f) + \delta > \mathrm{v}(f)$.

All observations are easy.

Break

The scheme

`Ford-Fulkerson algorithm`

(I) **Initialization:** Pick an initial flow.// For example $f \equiv 0$.

(S) **Search:** Find an f-augmentating path. If we find one, then go to (A); if there is no f-augmentating path, then go to (Stop).

(A) **Augmentation:** Based on the Lemma we "augment" f: $f \leftarrow f^+$. Go back to (S).

(Stop) **Stop:** Stop. Our flow can't be improved by augmenting paths.

Questions

(1) How to search for an augmenting path?

(2) The run of the algorithm is a repetition of a search. Is it possible that we run into an infinite cycle.

(3) What is the relation of optimal flows and those that can't be improved by augmenting path?

(1)

Definition: Partial augmenting path

A P path in $\overrightarrow{G}$ is a partial augmenting path iff it satisfies the constrains (A1) and (A3).

$P_0 : s$ is a partial augmenting path of length 0.

The main idea is that we start with the above example, we extend our partial augmenting paths (we have found so far) until we find a (complete) augmenting path or our search "run out of steam".

The Ford-Fulkerson search for augmenting path

Ford-Fulkerson search for augmenting path

`Initialization of the search`: $S := \{s\}$.
// S is the set of vertices that are reached by partial augmenting path.
`Extension of partial augmenting path`:
// Extension of S.
Let

$$B_{\text{forward}} = \{x \in V - S : \exists y \in S \quad \overrightarrow{yx} \in E \text{ and } f(\overrightarrow{yx}) < c(\overrightarrow{yx})\}$$

and

$$B_{\text{backward}} = \{x \in V - S : \exists y \in S \quad \overrightarrow{xy} \in E \text{ and } f(\overrightarrow{xy}) > 0\}.$$

Find an element x of $B_{\text{forward}} \cup B_{\text{backward}}$.

The search (con't)

Algorithm (cont'd)

(i) `Extension:` If $x \neq t$ then $S \leftarrow S \cup \{x\}$, and go back to `Extension of partial augmenting path`.

(ii) `Success:` If $x = t$ backtrack how we got to t. We find and *st* path, and that path is an augmenting path. We output the augmenting path and STOP.

(iii) `Unsuccessful search:` $B_{\text{forward}} \cup B_{\text{backward}} = \emptyset$. // $t \notin S$, we didn't find an augmenting path.

Backtracking

- It is worth to maintain a tree F on the vertex set S.

- At the beginning $S = \{s\}$, and our tree F is trivial.

- At each extension of S by a vertex e there is a vertex e^- in S such that the connecting edge is "responsible" for the extension. We extend our F by adding e and the edge e^-e. Hence F extends by outgrowth process, it will be a tree.

- This tree contains a unique sx path for each $x \in S$. That unique path will be a partial augmenting path.

What's an unsuccessful search?

In the case of unsuccessful search we will terminate the algorithm with a vertex set S. We know that $T = \overline{S} = V - S$ contains t.

We also know that for all edges $\overrightarrow{xy}$, $x \in S$, $y \in T$ we have $f(\overrightarrow{xy}) = c(\overrightarrow{xy})$, and for all edges $\overrightarrow{xy}$, $x \in T$, $y \in S$ we have $f(\overrightarrow{xy}) = 0$.

Cuts

Definition: Cut

Let $\overrightarrow{G}$ be a directed graph. $\mathcal{V} = \{S, T\}$ is a cut of $\overrightarrow{G}$ iff $V(\overrightarrow{G}) = S \dot\cup T$. S and T are the two parts of the cut. $\mathcal{V}$ is an *st-cut* iff $s \in S$ and $t \in T$.

Definition: Edge set of a cut

$E(\mathcal{V})$ denotes the edge set of $\mathcal{V}$:

$$E(\mathcal{V}) = \{\overrightarrow{xy} \in E(\overrightarrow{G}) : |\{x, y\} \cap S| = 1\}$$

In the case of an *st*-cut the source/sink roles partition $E(\mathcal{V})$ into two categories.

$$E_{\text{forward}}(\mathcal{V}) = \overrightarrow{E}(\mathcal{V}) = \{e = \overrightarrow{xy} : x \in S, y \in T\},$$
$$E_{\text{backward}}(\mathcal{V}) = \overleftarrow{E}(\mathcal{V}) = \{e = \overrightarrow{xy} : x \in T, y \in S\}.$$

Example

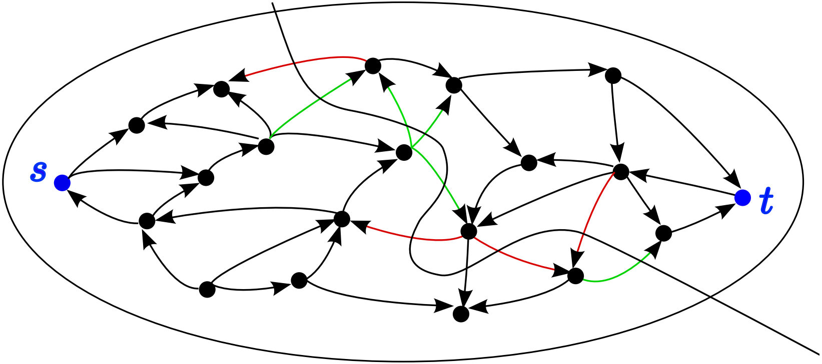


Figure: A cut

The value of a flow: Alternatives

Lemma

Let f be an arbitrary flow.

(i) $\mathrm{v}(f) = \sum_{e: e \in E_{out}(s)} f(e) - \sum_{e: e \in E_{in}(s)} f(e)$

(ii) For arbitrary cut $\mathcal{V} = \{S, T\}$

$$\mathrm{v}(f) = \sum_{e: e \in \overrightarrow{E}(\mathcal{V})} f(e) - \sum_{e: e \in \overleftarrow{E}(\mathcal{V})} f(e).$$

We can express the value of a flow $\mathrm{v}(f)$ using any cut.

Proof of the Lemma

It is enough to prove (ii).

For each vertex v in T we write an equality. In the case of $v \in T - \{t\}$ we write the flow preserving law:

$$\sum_{e: e \in E_{in}(v)} f(e) - \sum_{e: e \in E_{out}(v)} f(e) = 0.$$

In the case of $v = t$ we take the definition of $v(f)$:

$$\sum_{e: e \in E_{in}(v)} f(e) - \sum_{e: e \in E_{out}(v)} f(e) = \mathrm{v}(f).$$

Now we sum these equalities.

Proof of the Lemma (cont'd)

The right hand side will be $\mathrm{v}(f)$.

To see the right hand side of the sum we identify the variable x_e and the edge e. The edges of the network can be classified into four types.

The edges inside S don't show up in the equalities.

Each edge inside T is in two equalities. The two occurrences cancel out during the summation.

Each edge of $e \in \overrightarrow{E}(\mathcal{V})$ is an ingoing edge for one vertex of T. Its contribution to the sum is $+f(e)$.

Each edge of $e \in \overleftarrow{E}(\mathcal{V})$ is an outgoing edge for one vertex of T. Its contribution to the sum is $-f(e)$.

A Corollary of the Lemma

Corollary

f is an arbitrary flow, $\mathcal{V}$ is an arbitrary *st*-cut.

$$\mathrm{v}(f) \leq \sum_{e:e\in\overrightarrow{E}(\mathcal{V})} c(e) =: c(\mathcal{V}),$$

Definition

$c(\mathcal{V})$ is called the capacity of the cut.

Second corollary of the Lemma: (3)

Corollary

If the search of Ford and Fulkerson is unsuccessful then the flow f is optimal. Hence there is no f-augmenting path.

At the end of the unsuccessful search we obtain an st-cut $\mathcal{V}_{\text{exhaust}}$. For this cut on each forward edge the flow is the same as the capacity, and on each backward edge the flow is 0.

The bound of the first Corollary is sharp: $\mathrm{v}(f_{\text{actual}}) = c(\mathcal{V}_{\text{exhaust}})$.

But for an arbitrary flow we have $\mathrm{v}(f) \leq c(\mathcal{V}_{\text{exhaust}})$.

This implies that f_{actual} is an optimal flow.

Break

(2)

We know that if our search is unsuccessful (specially it stops) then the output is correct.

During the sequence of augmentations the value of the flow is strictly increasing and bounded. So it is convergent.

Is cycling (infinite loop of augmentation) possible?

1st answer

• In real life the capacity function is $c : E(\overrightarrow{G}) \to \mathbb{Q}_{++}$, and the initial flow is $f_0 : E(\overrightarrow{G}) \to \mathbb{Q}_{+}$.

• The capacities and the initial flows along the edges give us finite rational numbers. We can assume a common denominator.

• After scaling we can assume that $c : E(\overrightarrow{G}) \to \mathbb{N}$, and the initial flow is $f_0 : E(\overrightarrow{G}) \to \mathbb{N}$.

• Easy observation that when executing the Ford-Fulkerson algorithm we encounter only natural numbers. Specially the amount of the augmentation will be at least 1 after finding an augmenting path.

• Cycling is impossible

Summary of the 1st answer

Theorem

Let $\mathcal{H}$ be a network with rational capacities $c : E(\overrightarrow{G}) \to \mathbb{Q}_{++}$. Start the Ford-Fulkerson algorithm with a rational initial flow $f_0 : E(\overrightarrow{G}) \to \mathbb{Q}_{+}$. Then the algorithm terminates after finitely many augmentation and finds an optimal flow.

The Theorem is not quantitative. It doesn't give a good upper bound on the number of augmentations.

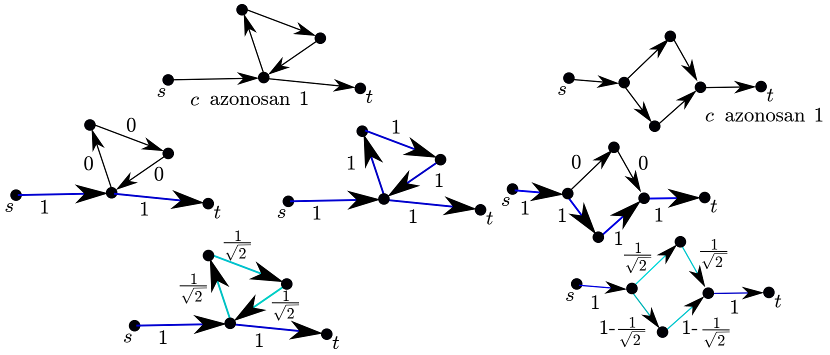
Flow Integrality Theorem

Corollary: Flow Integrality Theorem

Let $(\overrightarrow{G}, s, t, c)$ be with an integral capacity function $c : E(\overrightarrow{G}) \to \mathbb{N}_+$. Then there exists an optimal flow $f_{opt} : E(\overrightarrow{G}) \to \mathbb{N}$.

Flow integrality theorem: Example

We don't claim that any optimal flow is an integral flow. That is not true.



2nd answer

- Assuming exact real arithmetic the above proof doesn't work.

- Actually there are examples when the sequence of augmentations is infinite.

- Even the limit of the values of the computed flow is strictly less than the optimal value.

The fundamental theorem of flows

The fundamental theorem of flows

Let f be a flow in the network $H(\overrightarrow{G}, s, t, c)$. Then the following properties are equivalent:

(i) f is optimal, i.e. f is a maximum value flow,

(ii) for f we have an st-cut $\mathcal{V} = \{S, T\}$ such that $\mathrm{v}(f) = c(\mathcal{V})$,

(iii) There is no f-augmenting path.

The proof of the fundamental theorem

$(i) \Rightarrow (iii)$: We proved that the existence of an augmenting path implies that f is not optimal.

$(ii) \Rightarrow (i)$: For a cut $\mathcal{V}$ from (ii) we know that $c(\mathcal{V})$ is an upper bound of the value of an arbitrary flow. Hence f is optimal.

$(iii) \Rightarrow (ii)$: Run the Ford-Fulkerson algorithm. The search must be unsuccessful. At the end of the algorithm we have an *st*-cut: $\mathcal{V}_{\text{exhaust}}$. Furthermore $\mathrm{v}(f) = c(\mathcal{V}_{\text{exhaust}})$.

Reminder

We know that for any cut the capacity of the cut is an upper bound on the value of an arbitrary flow. The strongest claim is as follows:

$$\max_{f \text{ is a flow}} \mathrm{v}(f) \leq \min_{\mathcal{V} \text{ is an } st \text{ cut}} c(V).$$

From the above analysis we have that

$$\mathrm{v}(f_{\mathrm{opt}}) = c(\mathcal{V}_{\mathrm{exhaust}}).$$

The summary: MFMC Theorem

Max-flow-min-cut Theorem, MFMC Theorem

$$\max_{f \text{ is a flow}} \mathrm{v}(f) = \min_{\mathcal{V} \text{ is an } st\text{-cut}} c(\mathcal{V}).$$

There is a natural extension of the Ford-Fulkerson algorithm: When we output the final (optimal) flow, then we add the computed cut, $\mathcal{V}_{\text{exhaust}}$.

The added information is very useful. Even a non-mathematician will be convinced that the flow is optimal. The correctness of the output is transparent without seeing the the code, without understanding the theory behind.

Thank you for your attention!