Networks and Flows on Graphs

Maximum Flow Problem

Bashar Dudin

May 16, 2017

EPITA

What is it about?

Maximum Flow Problem

You have a water network together with a unique source. Each water conduct has a maximum water flow capacity. Using the whole network how can you maximize the water flow at a given sink?

Mathematical Modeling

A (transportation) network is a simple digraph $G = (V, A)$ having a single source s, a single sink t, and for each arrow a of G a valued capacity $c(a)$.

A (transportation) network is a simple digraph $G = (V, A)$ having a single source s, a single sink t, and for each arrow a of G a valued capacity $c(a)$.

Definition

A flow on a network G is a map $f : A \to \mathbb{R}_+$ such that:

- For each $a \in A$, $f(a) \leq c(a)$,
- For each vertex v, except for the source and sink, the sum of values f takes on ingoing arrows is equal to the sum taken by f on outgoing arrows. In a more pedantic fashion

$$\forall v \in V \setminus \{s, t\}, \quad \sum_{\{a | a \to s\}} f(a) = \sum_{\{a | s \to a\}} f(a)$$

Definition

The value $\mathrm{val}[f]$ of a flow f on a network $G = (V, A)$ is the sum of values taken by f on arrows going into t.

Definition

The value $\mathrm{val}[f]$ of a flow f on a network $G = (V, A)$ is the sum of values taken by f on arrows going into t.

Maximum Flow Problem

Given a network G find a flow f on G with maximum value flow.

Cuts

Definition of a cut

A cut of a digraph $G = (X, A)$ is a subset S of vertices such that S contains the source s but not the sink t.

Cuts

Definition of a cut

A cut of a digraph $G = (X, A)$ is a subset S of vertices such that S contains the source s but not the sink t.

We write $\overline{S}$ for the complement of S in X, the set of arrows having tail in S and head in $\overline{S}$ is written $\left(S, \overline{S}\right)$.

Remark : The two simplest cuts are respectively given by the cut only containing the source and the one containing every vertex but the sink.

Cuts

Definition of a cut

A cut of a digraph $G = (X, A)$ is a subset S of vertices such that S contains the source s but not the sink t.

We write $\overline{S}$ for the complement of S in X, the set of arrows having tail in S and head in $\overline{S}$ is written $(S, \overline{S})$.

Remark : The two simplest cuts are respectively given by the cut only containing the source and the one containing every vertex but the sink.

Definition

Given a flow f on a network $G = (X, A)$. The value flow $f[S]$ at a cut S of G is the sum $f[S, \overline{S}]$ of values taken by f on elements in $(S, \overline{S})$ minus the sum $f[\overline{S}, S]$ of values taken by f on $(\overline{S}, S)$.

Cuts

Proposition

Given a flow f on a network G the value flow at any cut is independant of the cut.

Cuts

Proposition

Given a flow f on a network G the value flow at any cut is independant of the cut.

This means the value of a given flow can be recovered as the value at any cut. In particular, the value flow is equal to the sum of values taken by the flow at arrows going out of the source.

Cuts

Proposition

Given a flow f on a network G the value flow at any cut is independant of the cut.

Proof : Let S_∞ be the cut of G containing everything but the sink. Let $S \subsetneq S_1 \subsetneq \cdots \subsetneq S_\infty$ be a maximal sequence of cuts (convince yourself that S_{k+1} has exactly one more vertex than S_k). By definition

$$f[S_\infty] = \mathrm{val}[f]$$

Let v_{k+1} be the only vertex in S_{k+1} but not in S_k.

$$f[S_{k+1}] = f[S_k] - \sum_{S_k \to a \to v_{k+1}} f(a) + \sum_{v_{k+1} \to a \to S_k} f(a) + \sum_{v_{k+1} \to a \to \overline{S}_{k+1}} f(a) - \sum_{\overline{S}_{k+1} \to a \to v_{k+1}} f(a)$$

These last four terms are exactly the difference between the values of f at ingoing arrows into v_{k+1} and outgoing ones from v_{k+1}. This difference is zero. ■

Cuts

Definition

Given an s-t cut S of G, we call *capacity* of S the non-negative number

$$c[S] = \sum_{a \in (S,\overline{S})} c(a).$$

Facts

Let f be a flow on a network G and let S be an s-t cut of G.

- $\mathrm{val}[f] \leq c[S]$
- $\mathrm{val}[f] = c[S]$ iff for all $a \in (S,\overline{S})$, $f(a) = c(a)$ and for all $a \in (\overline{S},S)$, $f(a) = 0$.

Thus, if $\mathrm{val}[f] = c[S]$ then f is a flow having maximal flow value and S is an s-t cut having minimal capacity.

Max-Flow Min-Cut Theorem

Theorem

The maximum value over all s-t flows is equal to the minimum capacity over all s-t cuts.

Max-Flow Min-Cut Theorem

Theorem

The maximum value over all s-t flows is equal to the minimum capacity over all s-t cuts.

A more precise statement is to say that, given a network G, there is a flow f and an s-t cut S such that $\text{val}[f] = c[S]$ iff f has maximum value among flows on G and S has minimum capacity among s-t cuts of G.

Max-Flow Min-Cut Theorem

Theorem

The maximum value over all s-t flows is equal to the minimum capacity over all s-t cuts.

A more precise statement is to say that, given a network G, there is a flow f and an s-t cut S such that $\text{val}[f] = c[S]$ iff f has maximum value among flows on G and S has minimum capacity among s-t cuts of G.

Conclusion

Finding the maximum value of a flow is equivalent to finding a minimizing cut.

Ford-Fulkerson Algorithm

A chain C in a network G, joining the source s to the sink t, comes with a natural orientation (a choice of an arrow for each edge) consistantly going from the source to the sink. We write C_+ the set of arrows of G supported on C and matching its orientation. We write C_- the set of arrows of G supported on C and opposite to that orientation.

Ford-Fulkerson Algorithm

A chain C in a network G, joining the source s to the sink t, comes with a natural orientation (a choice of an arrow for each edge) consistantly going from the source to the sink. We write C_+ the set of arrows of G supported on C and matching its orientation. We write C_- the set of arrows of G supported on C and opposite to that orientation.

Definition

Let f be a flow on a network G. An *augmenting chain* in G is a chain C joining the source s to the sink t, such that

$$\alpha = \min\left\{\min_{a\in C_+} c(a) - f(a), \min_{a\in C_-} f(a)\right\} > 0$$

Ford-Fulkerson Algorithm

Let f be a flow on a network G. Given an augmenting chain C in G one can modify f along C (and C alone) by adding α to each arrow in C_+ and substracting α to each arrow in C_-. Looking at the value of this new flow φ at the source one can see that

$$\text{val}[\varphi] = \text{val}[f] + \alpha$$

This shows that f was not a flow of maximim flow value.

Ford-Fulkerson Algorithm

Let f be a flow on a network G. Given an augmenting chain C in G one can modify f along C (and C alone) by adding α to each arrow in C_+ and substracting α to each arrow in C_-. Looking at the value of this new flow φ at the source one can see that

$$\mathrm{val}[\varphi] = \mathrm{val}[f] + \alpha$$

This shows that f was not a flow of maximim flow value. In fact

Proposition

A flow is of maximum flow value on a network G iff there are no augmenting chains in G.

Ford-Fulkerson Algorithm

Let f be a flow on a network G. Given an augmenting chain C in G one can modify f along C (and C alone) by adding α to each arrow in C_+ and substracting α to each arrow in C_-. Looking at the value of this new flow φ at the source one can see that

$$\mathrm{val}[\varphi] = \mathrm{val}[f] + \alpha$$

This shows that f was not a flow of maximim flow value. In fact

Proposition

A flow is of maximum flow value on a network G iff there are no augmenting chains in G.

Remark : The fact a flow on a network that doesn't have any augmenting chains is of maximal flow value is harder to grasp. You can get a proof in *Introduction to Algorithms* by Thomas H. Cormen and al.

Ford-Fulkerson Algorithm

Algorithm 1 Ford-Fulkerson Algorithm

Input: G a network and f a flow on G
Output: a maximal flow on G

1: **procedure** FORD-FULKERSON(G, f)
2: $(C, \alpha) \leftarrow$ AUGMENTINGCHAIN(G, f)
3: **while** $C \neq$ **None do**
4: f is increased or decreased by α on arrows in C depending on orientation
5: $(C, \alpha) \leftarrow$ AUGMENTINGCHAIN(G, f)
6: **end while**
7: **end procedure**

Ford-Fulkerson Algorithm

Algorithm 2 Finding an augmenting chain

Input: G a network and f a flow on G

Output: an augmenting chain on G if any, and α

1: **function** AUGMENTINGCHAIN(G, f)
2: Mark the source by a $+$
3: Mark head h of any unsaturated arrow (t, h), whose tail is marked, by $+t$
4: Mark tail t of any arrow (t, h) having non-zero flow, whose head is marked, by $-h$
5: **if** sink is marked **then**
6: **return** an augmenting chain C by following marks from the sink to the source, and α
7: **end if**
8: **end function**

Termination

Termination

The Ford-Fulkerson algorithm terminates if all capacities and flow values on arrows are rational.

Termination

Termination

The Ford-Fulkerson algorithm terminates if all capacities and flow values on arrows are rational.

This is clear : if flow is not of maximal value, we can find an augmenting chain along which we can increase flow value. At some point we get a maximal flow.

Termination

Termination

The Ford-Fulkerson algorithm terminates if all capacities and flow values on arrows are rational.

This is clear : if flow is not of maximal value, we can find an augmenting chain along which we can increase flow value. At some point we get a maximal flow. As to why we get a flow of maximal value : launch Ford-Fulkerson's algorithm one last time, write S for all marked vertices (this is not the set of all vertices because there is no augmenting chains), then the set of all arrows in $(S, \overline{S})$ are saturated and all those in $(\overline{S}, S)$ get flow value 0. This is enough to say that the capacity of s-t cut S is equal to the value flow we got; i.e. we have a minimal cut and thus a maximal flow.

Remark : If you google the Ford-Fulkerson algorithm, you should be able to find counter-examples in the case of irrational capacities.

Complexity

The complexity of the Ford-Fulkerson algorithm depends on how efficient you are to find augmentig paths, that's why it is sometimes called the Ford-Fulkerson *method* rather than algorithm.

Complexity

The complexity of the Ford-Fulkerson algorithm depends on how efficient you are to find augmentig paths, that's why it is sometimes called the Ford-Fulkerson *method* rather than algorithm. Without specifying how we visit vertices we have that

Complexity

Given a network $G = (V, A)$ with integer capacities and maximal value flow m the Ford-Fulkerson complexity is $O(ma)$, where a is the number of arrows of G.

To increase flow value by 1 you need to find an augmenting path by going through arrows of G, to possibly mark their sources and targets. Starting by the 0 flow you need to do it at worst m times, each time going through all arrows.

Complexity

The complexity of the Ford-Fulkerson algorithm depends on how efficient you are to find augmentig paths, that's why it is sometimes called the Ford-Fulkerson *method* rather than algorithm. Without specifying how we visit vertices we have that

Complexity

Given a network $G = (V, A)$ with integer capacities and maximal value flow m the Ford-Fulkerson complexity is $O(ma)$, where a is the number of arrows of G.

To increase flow value by 1 you need to find an augmenting path by going through arrows of G, to possibly mark their sources and targets. Starting by the 0 flow you need to do it at worst m times, each time going through all arrows.

Question

Can you imagine modifying the given algorithm for a better complexity?

Further readings

We are done with the basics, we'll be going through a couple of exercices about maximal flows and matchings.

Further readings

We are done with the basics, we'll be going through a couple of exercices about maximal flows and matchings. Your are encouraged to read about the following two related topics

- maximal flows at minimal costs
- general preference based matchings (uses what is sometimes called the *hungarian method*).