Network Flow Algorithm: Complexity, Improvements, Applications

Algorithm Design & Analysis

Fall 2018
Outline
Recap

Let $G$ be a flow network, $f$ a flow on $G$ and $G_f$ the residual flow graph of $G$ and $f$.
Recap

Let $G$ be a flow network, $f$ a flow on $G$ and $G_f$ the residual flow graph of $G$ and $f$

- If $G_f$ contains no flow-augmenting path, then there is a cut $[A, B]$ with $v(f) = \text{cap}(A, B)$.
Recap

Let $G$ be a flow network, $f$ a flow on $G$ and $G_f$ the residual flow graph of $G$ and $f$

- If $G_f$ contains no flow-augmenting path, then there is a cut $[A, B]$ with $v(f) = \text{cap}(A, B)$.
- If there is a cut $[A, B]$ with $v(f) = \text{cap}(A, B)$ then $f$ is a maximum flow (and $[A, B]$ is a minimum capacity cut).
Recap

Let $G$ be a flow network, $f$ a flow on $G$ and $G_f$ the residual flow graph of $G$ and $f$

- If $G_f$ contains no flow-augmenting path, then there is a cut $[A, B]$ with $v(f) = \text{cap}(A, B)$.
- If there is a cut $[A, B]$ with $v(f) = \text{cap}(A, B)$ then $f$ is a maximum flow (and $[A, B]$ is a minimum capacity cut)
- If $f$ is a maximum flow, then $G_f$ has no flow-augmenting path.
Recap

Let $G$ be a flow network, $f$ a flow on $G$ and $G_f$ the residual flow graph of $G$ and $f$

- If $G_f$ contains no flow-augmenting path, then there is a cut $[A, B]$ with $v(f) = \text{cap}(A, B)$.
- If there is a cut $[A, B]$ with $v(f) = \text{cap}(A, B)$ then $f$ is a maximum flow (and $[A, B]$ is a minimum capacity cut)
- If $f$ is a maximum flow, then $G_f$ has no flow-augmenting path.

Thus $f$ is a maximum flow if and only if $G_f$ has no flow-augmenting path if and only if $v(f) = \text{cap}[A, B]$ for some cut $[A, B]$. 
Some observations

- If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued.
- Thus there is always an integer-valued maximum flow.
- \( v(f) \) will increase by at least 1 with every augmentation.
- If \( C = \max(s,v) \in E \), then there are at most \( nC \) augmentations.
- Thus runtime is \( O(mnC) \).
- \( G_f \) can be constructed in \( O(m+n) \) time.
- A flow-augmenting path can be found in \( O(m+n) \) time.
- \( G_f \) can be updated in \( O(n) \) time.
- \( n \in O(m) \) if \( G \) is connected.
- Total space needed is \( O(m+n) \).
Complexity Analysis

Some observations

• If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued.
Some observations

- If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued
- Thus there is always an *integer-valued* maximum flow
Some observations

- If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued.
- Thus there is always an integer-valued maximum flow.
- Also $\nu(f)$ will increase by at least 1 with every augmentation.
Some observations

• If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued
• Thus there is always an integer-valued maximum flow
• Also $v(f)$ will increase by at least 1 with every augmentation
• If $C = \max_{(s,v) \in E} c(s, v)$, then there are at most $nC$ augmentations
Complexity Analysis

Some observations

- If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued
- Thus there is always an *integer-valued* maximum flow
- Also $v(f)$ will increase by at least 1 with every augmentation
- If $C = \max_{(s,v) \in E} c(s, v)$, then there are at most $nC$ augmentations
- Thus runtime is $O(mnC)$
Complexity Analysis

Some observations

- If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued.
- Thus there is always an *integer-valued* maximum flow.
- Also $\nu(f)$ will increase by at least 1 with every augmentation.
- If $C = \max_{(s,v) \in E} c(s,v)$, then there are at most $nC$ augmentations.
- Thus runtime is $O(mnC)$.
  - $G_f$ can be constructed in $O(m + n)$ time.

• $n \in O(m)$ if $G$ is connected.

Total space needed is $O(m + n)$. 
Complexity Analysis

Some observations

- If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued.
- Thus there is always an *integer-valued* maximum flow.
- Also $v(f)$ will increase by at least 1 with every augmentation.
- If $C = \max_{(s,v) \in E} c(s,v)$, then there are at most $nC$ augmentations.
- Thus runtime is $O(mnC)$.
  - $G_f$ can be constructed in $O(m + n)$ time.
  - A flow-augmenting path can be found in $O(m + n)$ time.

Total space needed is $O(m + n)$ if $G$ is connected.
Complexity Analysis

Some observations

- If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued
- Thus there is always an *integer-valued* maximum flow
- Also $v(f)$ will increase by at least 1 with every augmentation
- If $C = \max_{(s,v) \in E} c(s, v)$, then there are at most $nC$ augmentations
- Thus runtime is $O(mnC)$
  - $G_f$ can be constructed in $O(m + n)$ time
  - A flow-augmenting path can be found in $O(m + n)$ time
  - $G_f$ can be updated in $O(n)$ time.

$n \in O(m)$ if $G$ is connected

Total space needed is $O(m + n)$
Complexity Analysis

Some observations

- If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued
- Thus there is always an integer-valued maximum flow
- Also $v(f)$ will increase by at least 1 with every augmentation
- If $C = \max_{(s,v) \in E} c(s, v)$, then there are at most $nC$ augmentations
- Thus runtime is $O(mnC)$
  - $G_f$ can be constructed in $O(m + n)$ time
  - A flow-augmenting path can be found in $O(m + n)$ time
  - $G_f$ can be updated in $O(n)$ time.
  - $n \in O(m)$ if $G$ is connected

Total space needed is $O(m + n)$
**Complexity Analysis**

Some observations

- If capacities are integers and initial flow is 0, then all intermediate flows will be integer-valued.
- Thus there is always an *integer-valued* maximum flow.
- Also $v(f)$ will increase by at least 1 with every augmentation.
- If $C = \max_{(s,v) \in E} c(s, v)$, then there are at most $nC$ augmentations.
- Thus runtime is $O(mnC)$.
  - $G_f$ can be constructed in $O(m + n)$ time.
  - A flow-augmenting path can be found in $O(m + n)$ time.
  - $G_f$ can be updated in $O(n)$ time.
  - $n \in O(m)$ if $G$ is connected.
- Total space needed is $O(m + n)$. 
Choosing Good Augmenting Paths

• Ford-Fulkerson is pseudo-polynomial, since $C$ could be exponential in $n$ and $m$.  

\begin{itemize}
  \item Look for path that maximizes bottleneck capacity
  \item Look for path with large bottleneck capacity
  \item Look for path with fewest edges
  \item Do something other than finding augmenting paths
\end{itemize}
Choosing Good Augmenting Paths

- Ford-Fulkerson is *pseudo-polynomial*, since $C$ could be exponential in $n$ and $m$.
- There are simple pathological cases resulting in $O(nC)$ augmentations.
Choosing Good Augmenting Paths

- Ford-Fulkerson is *pseudo-polynomial*, since $C$ could be exponential in $n$ and $m$.
- There are simple pathological cases resulting in $O(nC)$ augmentations
- Strategies for good augmenting paths
Choosing Good Augmenting Paths

- Ford-Fulkerson is *pseudo-polynomial*, since $C$ could be exponential in $n$ and $m$.
- There are simple pathological cases resulting in $O(nC)$ augmentations.
- Strategies for good augmenting paths:
  - Look for path that maximizes bottleneck capacity.
Choosing Good Augmenting Paths

• Ford-Fulkerson is *pseudo-polynomial*, since $C$ could be exponential in $n$ and $m$.

• There are simple pathological cases resulting in $O(nC)$ augmentations

• Strategies for good augmenting paths
  • Look for path that maximizes bottleneck capacity
  • Look for path with large bottleneck capacity
Choosing Good Augmenting Paths

- Ford-Fulkerson is *pseudo-polynomial*, since $C$ could be exponential in $n$ and $m$.
- There are simple pathological cases resulting in $O(nC)$ augmentations.
- Strategies for good augmenting paths
  - Look for path that maximizes bottleneck capacity
  - Look for path with large bottleneck capacity
  - Look for path with fewest edges
Choosing Good Augmenting Paths

• Ford-Fulkerson is \textit{pseudo-polynomial}, since \( C \) could be exponential in \( n \) and \( m \).
• There are simple pathological cases resulting in \( O(nC) \) augmentations
• Strategies for good augmenting paths
  • Look for path that maximizes bottleneck capacity
  • Look for path with large bottleneck capacity
  • Look for path with fewest edges
  • Do something other than finding augmenting paths
Augmenting Path Selection with Scaling

Idea: Consider only edges in $G_f$ with large capacity
Augmenting Path Selection with Scaling

Idea: Consider only edges in $G_f$ with large capacity

• Pick a large value $\Delta$ (say a power of 2)
Augmenting Path Selection with Scaling

Idea: Consider only edges in $G_f$ with large capacity

- Pick a large value $\Delta$ (say a power of 2)
- Let $G_f(\Delta)$ be the subgraph of $G_f$ of edges $e$ with $c(e) \geq \Delta$
Augmenting Path Selection with Scaling

**Idea:** Consider only edges in $G_f$ with large capacity

- Pick a large value $\Delta$ (say a power of 2)
- Let $G_f(\Delta)$ be the subgraph of $G_f$ of edges $e$ with $c(e) \geq \Delta$
- **$\Delta$-scaling phase:** Repeatedly search $G_f(\Delta)$ for flow-augmenting $s, t$-paths, augmenting $f$

Eventually, $\Delta = 1$ and $G_f(\Delta) = G_f$; normal F-F algorithm is in force

Thus a maximum-value flow will be found. That is: The algorithm is correct, now let's show that it's fast!
Augmenting Path Selection with Scaling

Idea: Consider only edges in $G_f$ with large capacity

- Pick a large value $\Delta$ (say a power of 2)
- Let $G_f(\Delta)$ be the subgraph of $G_f$ of edges $e$ with $c(e) \geq \Delta$
- $\Delta$-scaling phase: Repeatedly search $G_f(\Delta)$ for flow-augmenting $s, t$-paths, augmenting $f$
- When no such path is found, set $\Delta \leftarrow \Delta/2$ and repeat
Augmenting Path Selection with Scaling

Idea: Consider only edges in $G_f$ with large capacity

- Pick a large value $\Delta$ (say a power of 2)
- Let $G_f(\Delta)$ be the subgraph of $G_f$ of edges $e$ with $c(e) \geq \Delta$
- $\Delta$-scaling phase: Repeatedly search $G_f(\Delta)$ for flow-augmenting $s, t$-paths, augmenting $f$
- When no such path is found, set $\Delta \leftarrow \Delta/2$ and repeat
- Eventually, $\Delta = 1$ and $G_f(\Delta) = G_f$; normal F-F algorithm is in force
Augmenting Path Selection with Scaling

Idea: Consider only edges in $G_f$ with large capacity

- Pick a large value $\Delta$ (say a power of 2)
- Let $G_f(\Delta)$ be the subgraph of $G_f$ of edges $e$ with $c(e) \geq \Delta$
- $\Delta$-scaling phase: Repeatedly search $G_f(\Delta)$ for flow-augmenting $s, t$-paths, augmenting $f$
- When no such path is found, set $\Delta \leftarrow \Delta/2$ and repeat
- Eventually, $\Delta = 1$ and $G_f(\Delta) = G_f$; normal F-F algorithm is in force
- Thus a maximum-value flow will be found.

That is: The algorithm is correct, now let’s show that it’s fast!
Augmenting Path Selection with Scaling

**Idea:** Consider only edges in $G_f$ with large capacity

- Pick a large value $\Delta$ (say a power of 2)
- Let $G_f(\Delta)$ be the subgraph of $G_f$ of edges $e$ with $c(e) \geq \Delta$
- $\Delta$-scaling phase: Repeatedly search $G_f(\Delta)$ for flow-augmenting $s, t$-paths, augmenting $f$
- When no such path is found, set $\Delta \leftarrow \Delta/2$ and repeat
- Eventually, $\Delta = 1$ and $G_f(\Delta) = G_f$; normal F-F algorithm is in force
- Thus a maximum-value flow will be found.

That is: The algorithm is *correct*, now let’s show that it’s *fast*!
Augmentation with Scaling: Complexity Analysis

- Start with $\Delta$ such that $\Delta \leq C < 2\Delta$ where $C$ is the max. capacity of an edge leaving $s$
Augmentation with Scaling: Complexity Analysis

- Start with $\Delta$ such that $\Delta \leq C < 2\Delta$ where $C$ is the max. capacity of an edge leaving $s$
- Thus, there are at most $\lceil \log C \rceil \Delta$-scaling phases
Augmentation with Scaling: Complexity Analysis

• Start with $\Delta$ such that $\Delta \leq C < 2\Delta$ where $C$ is the max. capacity of an edge leaving $s$
• Thus, there are at most $\lceil \log C \rceil \Delta$-scaling phases
• We’ll show that each $\Delta$-scaling phase entails at most $2m$ augmentations.
Augmentation with Scaling: Complexity Analysis

- Start with $\Delta$ such that $\Delta \leq C < 2\Delta$ where $C$ is the max. capacity of an edge leaving $s$
- Thus, there are at most $\lceil \log C \rceil$ $\Delta$-scaling phases
- We’ll show that each $\Delta$-scaling phase entails at most $2m$ augmentations.
- This guarantees a run-time of $O(m^2 \log C)$
Augmentation with Scaling : Complexity Analysis

- Start with $\Delta$ such that $\Delta \leq C < 2\Delta$ where $C$ is the max. capacity of an edge leaving $s$
- Thus, there are at most $\left\lceil \log C \right\rceil$ $\Delta$-scaling phases
- We’ll show that each $\Delta$-scaling phase entails at most $2m$ augmentations.
- This guarantees a run-time of $O(m^2 \log C)$
- Since $\log C$ is (at most) linear in the problem input size, we have a polynomial run time!
Augmentation with Scaling: Complexity Analysis

- Start with $\Delta$ such that $\Delta \leq C < 2\Delta$ where $C$ is the max. capacity of an edge leaving $s$
- Thus, there are at most $\lceil \log C \rceil \Delta$-scaling phases
- We’ll show that each $\Delta$-scaling phase entails at most $2m$ augmentations.
- This guarantees a run-time of $O(m^2 \log C)$
- Since $\log C$ is (at most) linear in the problem input size, we have a polynomial run time!

Note: Every augmentation in $\Delta$-scaling phase increases flow by $\Delta$
Lemma
If $f^*$ is a maximum flow and $f$ is the flow at the end of the $\Delta$-scaling phase, then $v(f^*) - v(f) \leq m\Delta$.

Proof.
Number of Augmentations per Scaling Phase

Lemma
If $f^*$ is a maximum flow and $f$ is the flow at the end of the $\Delta$-scaling phase, then $v(f^*) - v(f) \leq m\Delta$.

Proof.
Let $A$ be all vertices reachable from $s$ in $G_f(\Delta)$ at end of $\Delta$-scaling phase.
Lemma

If $f^*$ is a maximum flow and $f$ is the flow at the end of the $\Delta$-scaling phase, then $v(f^*) - v(f) \leq m\Delta$.

Proof.

Let $A$ be all vertices reachable from $s$ in $G_f(\Delta)$ at end of $\Delta$-scaling phase.

- $[A, B]$ is an $s, t$-cut of $G$
Lemma

If $f^*$ is a maximum flow and $f$ is the flow at the end of the $\Delta$-scaling phase, then $v(f^*) - v(f) \leq m\Delta$.

Proof.

Let $A$ be all vertices reachable from $s$ in $G_f(\Delta)$ at end of $\Delta$-scaling phase.

- $[A, B]$ is an $s, t$-cut of $G$
- For every $e \in [A, B]$, $c(e) - f(e) < \Delta$, so $c(e) - \Delta < f(e)$
Lemma

If $f^*$ is a maximum flow and $f$ is the flow at the end of the $\Delta$-scaling phase, then $v(f^*) - v(f) \leq m\Delta$.

Proof.

Let $A$ be all vertices reachable from $s$ in $G_f(\Delta)$ at end of $\Delta$-scaling phase.

- $[A, B]$ is an $s, t$-cut of $G$
- For every $e \in [A, B]$, $c(e) - f(e) < \Delta$, so $c(e) - \Delta < f(e)$
- For every $e \in [B, A]$, $f(e) < \Delta$
Lemma
If \( f^* \) is a maximum flow and \( f \) is the flow at the end of the \( \Delta \)-scaling phase, then \( v(f^*) - v(f) \leq m\Delta \).

Proof.
Let \( A \) be all vertices reachable from \( s \) in \( G_f(\Delta) \) at end of \( \Delta \)-scaling phase.

- \([A, B]\) is an \( s, t \)-cut of \( G \)
- For every \( e \in [A, B] \), \( c(e) - f(e) < \Delta \), so \( c(e) - \Delta < f(e) \)
- For every \( e \in [B, A] \), \( f(e) < \Delta \)
- Now calculate \( v(f) \) ...
Proof of Lemma continued

\[ v(f) = \sum_{e \in [A, B]} f(e) - \sum_{e \in [B, A]} f(e) \] \hspace{1cm} (1)

Thus
\[ v(f^*) - v(f) \leq m \Delta \]; that is:
\[ v(f^*) \leq v(f) + m \Delta \] \hspace{1cm} (5)
Proof of Lemma continued

\[ v(f) = \sum_{e \in [A,B]} f(e) - \sum_{e \in [B,A]} f(e) \]  \hspace{1cm} (1)

\[ > \sum_{e \in [A,B]} (c(e) - \Delta) - \sum_{e \in [B,A]} \Delta \]  \hspace{1cm} (2)

\[ \geq \sum_{e \in [A,B]} c(e) - m \Delta \]  \hspace{1cm} (3)

\[ = \text{cap} \left[ A, B \right] - m \Delta \geq v(f^*) - m \Delta \]  \hspace{1cm} (4)

Thus \[ v(f^*) - v(f) \leq m \Delta; \] that is:

\[ v(f^*) \leq v(f) + m \Delta \]  \hspace{1cm} (5)
Proof of Lemma continued

\[ v(f) = \sum_{e \in [A,B]} f(e) - \sum_{e \in [B,A]} f(e) \quad (1) \]

\[ > \sum_{e \in [A,B]} (c(e) - \Delta) - \sum_{e \in [B,A]} \Delta \quad (2) \]

\[ = \sum_{e \in [A,B]} c(e) - \sum_{e \in [A,B]} \Delta - \sum_{e \in [B,A]} \Delta \quad (3) \]

\[ \geq \sum_{e \in [A,B]} c(e) - m \Delta \quad (4) \]

\[ = \text{cap} [A,B] - m \Delta \geq v(f^*) - m \Delta \quad (5) \]

Thus \( v(f^*) - v(f) \leq m \Delta \); that is:

\[ v(f^*) \leq v(f) + m \Delta \]
Proof of Lemma continued

\[ v(f) = \sum_{e \in [A,B]} f(e) - \sum_{e \in [B,A]} f(e) \]  
\[ > \sum_{e \in [A,B]} (c(e) - \Delta) - \sum_{e \in [B,A]} \Delta \]  
\[ = \sum_{e \in [A,B]} c(e) - \sum_{e \in [A,B]} \Delta - \sum_{e \in [B,A]} \Delta \]  
\[ \geq \sum_{e \in [A,B]} c(e) - m\Delta \]
Proof of Lemma continued

\[ v(f) = \sum_{e \in [A,B]} f(e) - \sum_{e \in [B,A]} f(e) \]  \hspace{1cm} (1)

\[ > \sum_{e \in [A,B]} (c(e) - \Delta) - \sum_{e \in [B,A]} \Delta \]  \hspace{1cm} (2)

\[ = \sum_{e \in [A,B]} c(e) - \sum_{e \in [A,B]} \Delta - \sum_{e \in [B,A]} \Delta \]  \hspace{1cm} (3)

\[ \geq \sum_{e \in [A,B]} c(e) - m\Delta \]  \hspace{1cm} (4)

\[ = \text{cap}[A, B] - m\Delta \geq v(f^*) - m\Delta \]  \hspace{1cm} (5)
Proof of Lemma continued

\[ v(f) = \sum_{e \in [A,B]} f(e) - \sum_{e \in [B,A]} f(e) \]  \hspace{1cm} (1)

\[ > \sum_{e \in [A,B]} (c(e) - \Delta) - \sum_{e \in [B,A]} \Delta \]  \hspace{1cm} (2)

\[ = \sum_{e \in [A,B]} c(e) - \sum_{e \in [B,A]} \Delta - \sum_{e \in [B,A]} \Delta \]  \hspace{1cm} (3)

\[ \geq \sum_{e \in [A,B]} c(e) - m\Delta \]  \hspace{1cm} (4)

\[ = \operatorname{cap}[A, B] - m\Delta \geq v(f^*) - m\Delta \]  \hspace{1cm} (5)

Thus \( v(f^*) - v(f) \leq m\Delta \); that is: \( v(f^*) \leq v(f) + m\Delta \)
Wrapping up the Analysis

Lemma
Each \( \Delta \)-scaling phase performs at most \( 2m \) augmentations.

Proof.


**Wrapping up the Analysis**

**Lemma**

*Each* Δ*-scaling phase performs at most 2m augmentations.*

**Proof.**

- First phase: True: An edge leaving s can be used in at most one augmentation
Wrapping up the Analysis

Lemma
Each $\Delta$-scaling phase performs at most $2m$ augmentations.

Proof.

- First phase: True: An edge leaving $s$ can be used in at most one augmentation
- Consider any other phase $\Delta$, and the previous $\Delta'$-phase ($\Delta' = 2\Delta$); let $f_p$ be the flow value at end of $\Delta'$-phase
Wrapping up the Analysis

Lemma
Each $\Delta$-scaling phase performs at most $2m$ augmentations.

Proof.

• First phase: True: An edge leaving $s$ can be used in at most one augmentation

• Consider any other phase $\Delta$, and the previous $\Delta'$-phase ($\Delta' = 2\Delta$); let $f_p$ be the flow value at end of $\Delta'$-phase
  • At end of $\Delta'$- phase, $v(f^*) \leq v(f_p) + m\Delta'$
Wrapping up the Analysis

Lemma
Each $\Delta$-scaling phase performs at most $2m$ augmentations.

Proof.

- First phase: True: An edge leaving $s$ can be used in at most one augmentation
- Consider any other phase $\Delta$, and the previous $\Delta'$-phase ($\Delta' = 2\Delta$); let $f_p$ be the flow value at end of $\Delta'$-phase
  - At end of $\Delta'$- phase, $\nu(f^*) \leq \nu(f_p) + m\Delta'$
  - So $\nu(f^*) \leq \nu(f_p) + 2m\Delta$ at beginning of $\Delta$ phase
Wrapping up the Analysis

Lemma
Each $\Delta$-scaling phase performs at most $2m$ augmentations.

Proof.

• First phase: True: An edge leaving $s$ can be used in at most one augmentation

• Consider any other phase $\Delta$, and the previous $\Delta'$-phase ($\Delta' = 2\Delta$); let $f_p$ be the flow value at end of $\Delta'$-phase
  • At end of $\Delta'$- phase, $\nu(f^*) \leq \nu(f_p) + m\Delta'$
  • So $\nu(f^*) \leq \nu(f_p) + 2m\Delta$ at beginning of $\Delta$ phase
  • But each augmentation of $\Delta$-phase increases flow by at least $\Delta$
Wrapping up the Analysis

Lemma
Each $\Delta$-scaling phase performs at most $2m$ augmentations.

Proof.

- First phase: True: An edge leaving $s$ can be used in at most one augmentation
- Consider any other phase $\Delta$, and the previous $\Delta'$-phase ($\Delta' = 2\Delta$); let $f_p$ be the flow value at end of $\Delta'$-phase
  - At end of $\Delta'$-phase, $v(f^*) \leq v(f_p) + m\Delta'$
  - So $v(f^*) \leq v(f_p) + 2m\Delta$ at beginning of $\Delta$ phase
  - But each augmentation of $\Delta$-phase increases flow by at least $\Delta$
  - Thus there can be at most $2m$ such augmentations before the flow would exceed $v(f^*)$
Multiple Sources & Multiple Sinks

We can easily adapt our methods to networks with multiple sources and sinks.
Multiple Sources & Multiple Sinks

We can easily adapt our methods to networks with multiple sources and sinks.

An \((S, T)\)-flow network is a directed graph \(G = (V, E)\) with edge capacities \(c(e) \geq 0\) where \(S, T\) are disjoint subsets of \(V\) in which each \(s \in S\) has in-degree 0 and each \(t \in T\) has out-degree 0.
Multiple Sources & Multiple Sinks

We can easily adapt our methods to networks with multiple sources and sinks.

An \((S, T)\)-flow network is a directed graph \(G = (V, E)\) with edge capacities \(c(e) \geq 0\) where \(S, T\) are disjoint subsets of \(V\) in which each \(s \in S\) has in-degree 0 and each \(t \in T\) has out-degree 0.

An \((S, T)\) flow \(f\) on an \((S, T)\)-flow network assigns a flow \(0 \leq f(e) \leq c(e)\) to each edge such that for every \(v \not\in S \cup T\), \(f^{in}(v) = f^{out}(v)\).
We can easily adapt our methods to networks with multiple sources and sinks.

An \((S, T)\)-flow network is a directed graph \(G = (V, E)\) with edge capacities \(c(e) \geq 0\) where \(S, T\) are disjoint subsets of \(V\) in which each \(s \in S\) has in-degree 0 and each \(t \in T\) has out-degree 0.

An \((S, T)\) flow \(f\) on an \((S, T)\)-flow network assigns a flow \(0 \leq f(e) \leq c(e)\) to each edge such that for every \(v \notin S \cup T\), \(f^{in}(v) = f^{out}(v)\).

The value of an \((S, T)\) flow \(f\) is given by \(v(f) = \sum_{s \in S} f^{out}(s)\).
Multiple Sources & Multiple Sinks

An \((S, T)\)-cut on an \((S, T)\)-flow network is a partition of the vertices \((A, B)\) such that \(S \subseteq A\) and \(T \subseteq B\).
Multiple Sources & Multiple Sinks

An $(S, T)$-cut on an $(S, T)$-flow network is a partition of the vertices $(A, B)$ such that $S \subseteq A$ and $T \subseteq B$

All of the facts established for single source/sink flows have obvious analogs for $(S, T)$-flow networks. Here’s the trick to seeing this.
Multiple Sources & Multiple Sinks

An \((S, T)\)-cut on an \((S, T)\)-flow network is a partition of the vertices \((A, B)\) such that \(S \subseteq A\) and \(T \subseteq B\)

All of the facts established for single source/sink flows have obvious analogs for \((S, T)\)-flow networks. Here’s the trick to seeing this.

Given an \((S, T)\)-flow network \(G = (V, E)\), create a new flow network \(G' = (V', E')\) where
Multiple Sources & Multiple Sinks

An \((S, T)\)-cut on an \((S, T)\)-flow network is a partition of the vertices \((A, B)\) such that \(S \subseteq A\) and \(T \subseteq B\)

All of the facts established for single source/sink flows have obvious analogs for \((S, T)\)-flow networks. Here’s the trick to seeing this.

Given an \((S, T)\)-flow network \(G = (V, E)\), create a new flow network \(G' = (V', E')\) where

- \(V' = V \cup \{s', t'\}\)
Multiple Sources & Multiple Sinks

An \((S, T)\)-cut on an \((S, T)\)-flow network is a partition of the vertices \((A, B)\) such that \(S \subseteq A\) and \(T \subseteq B\).

All of the facts established for single source/sink flows have obvious analogs for \((S, T)\)-flow networks. Here’s the trick to seeing this.

Given an \((S, T)\)-flow network \(G = (V, E)\), create a new flow network \(G' = (V', E')\) where

- \(V' = V \cup \{s', t'\}\)
- \(E' = E \cup \{(s', s) : s \in S\} \cup \{(t, t') : t \in T\}\)
Multiple Sources & Multiple Sinks

An \((S, T)\)-cut on an \((S, T)\)-flow network is a partition of the vertices \((A, B)\) such that \(S \subseteq A\) and \(T \subseteq B\).

All of the facts established for single source/sink flows have obvious analogs for \((S, T)\)-flow networks. Here’s the trick to seeing this.

Given an \((S, T)\)-flow network \(G = (V, E)\), create a new flow network \(G' = (V', E')\) where

- \(V' = V \cup \{s', t'\}\)
- \(E' = E \cup \{(s', s) : s \in S\} \cup \{(t, t') : t \in T\}\)
- For each \(s \in S\), \(\text{cap}(s', s) = 1 + \sum_{(s, u) \in E} \text{cap}(s, u)\)
  The "+1" ensures that residual capacity is always positive.
Multiple Sources & Multiple Sinks

An \((S, T)\)-cut on an \((S, T)\)-flow network is a partition of the vertices \((A, B)\) such that \(S \subseteq A\) and \(T \subseteq B\).

All of the facts established for single source/sink flows have obvious analogs for \((S, T)\)-flow networks. Here’s the trick to seeing this.

Given an \((S, T)\)-flow network \(G = (V, E)\), create a new flow network \(G' = (V', E')\) where

1. \(V' = V \cup \{s', t'\}\)
2. \(E' = E \cup \{(s', s) : s \in S\} \cup \{(t, t') : t \in T\}\)
3. For each \(s \in S\), \(\text{cap}(s', s) = 1 + \sum_{(s,u) \in E} \text{cap}(s, u)\)
   The "+1" ensures that residual capacity is always positive.
4. For each \(t \in T\), \(\text{cap}(t, t') = \sum_{(u,t) \in E} \text{cap}(u, t)\)
Lemma

The \((S, T)\)-flow network \(G\) has a flow \(f\) of value \(a\) if and only if the \((s', t')\)-flow network \(G'\) has a flow \(f'\) of value \(a\).
Multiple Sources & Multiple Sinks

Lemma
The \((S, T)\)-flow network \(G\) has a flow \(f\) of value \(a\) if and only if the \((s', t')\)-flow network \(G'\) has a flow \(f'\) of value \(a\).

Lemma
Given an \((S, T)\)-flow network \(G\) and a flow \(f\), if the cut \((A, B)\) induced by \(f'\) in \(G'\) is an \((s', t')\)-cut, then \((A, B)\) is an \((S, T)\)-cut in \(G\) with the same capacity.
Multiple Sources & Multiple Sinks

Lemma
The \((S, T)\)-flow network \(G\) has a flow \(f\) of value \(a\) if and only if the \((s', t')\)-flow network \(G'\) has a flow \(f'\) of value \(a\).

Lemma
Given an \((S, T)\)-flow network \(G\) and a flow \(f\), if the cut \((A, B)\) induced by \(f'\) in \(G'\) is an \((s', t')\)-cut, then \((A, B)\) is an \((S, T)\)-cut in \(G\) with the same capacity.

Proof.
If not, some \(t \in T\) is reachable from \(s'\) in \(G'\). Let \(e' = (t, t')\).
Multiple Sources & Multiple Sinks

Lemma
The \((S, T)\)-flow network \(G\) has a flow \(f\) of value \(a\) if and only if the \((s', t')\)-flow network \(G'\) has a flow \(f'\) of value \(a\).

Lemma
Given an \((S, T)\)-flow network \(G\) and a flow \(f\), if the cut \((A, B)\) induced by \(f'\) in \(G'\) is an \((s', t')\)-cut, then \((A, B)\) is an \((S, T)\)-cut in \(G\) with the same capacity.

Proof.
If not, some \(t \in T\) is reachable from \(s'\) in \(G'\). Let \(e' = (t, t')\)
  - Then \(f'(e') = f^\text{in}(t) < \sum_{e \text{ into } t} \text{cap}(e) = \text{cap}(e')\)
**Multiple Sources & Multiple Sinks**

**Lemma**
The \((S, T)\)-flow network \(G\) has a flow \(f\) of value \(a\) if and only if the \((s', t')\)-flow network \(G'\) has a flow \(f'\) of value \(a\).

**Lemma**
Given an \((S, T)\)-flow network \(G\) and a flow \(f\), if the cut \((A, B)\) induced by \(f'\) in \(G'\) is an \((s', t')\)-cut, then \((A, B)\) is an \((S, T)\)-cut in \(G\) with the same capacity.

**Proof.**

If not, some \(t \in T\) is reachable from \(s'\) in \(G'_{f'}\). Let \(e' = (t, t')\)

- Then \(f'(e') = f^{in}(t) < \sum_{e \text{ into } t} \text{cap}(e) = \text{cap}(e')\)
- So \(t'\) is reachable from \(s'\), so \((A, B)\) isn’t an \((s', t')\)-cut
Multiple Sources & Multiple Sinks

Lemma
The \((S, T)\)-flow network \(G\) has a flow \(f\) of value \(a\) if and only if the \((s', t')\)-flow network \(G'\) has a flow \(f'\) of value \(a\).

Lemma
Given an \((S, T)\)-flow network \(G\) and a flow \(f\), if the cut \((A, B)\) induced by \(f'\) in \(G'\) is an \((s', t')\)-cut, then \((A, B)\) is an \((S, T)\)-cut in \(G\) with the same capacity.

Proof.
If not, some \(t \in T\) is reachable from \(s'\) in \(G'\). Let \(e' = (t, t')\)

- Then \(f'(e') = f^{\text{in}}(t) < \sum_{e \text{ into } t} \text{cap}(e) = \text{cap}(e')\)
- So \(t'\) is reachable from \(s'\), so \((A, B)\) isn’t an \((s', t')\)-cut
- So no \(t \in T\) is reachable from \(s'\), making \((A, B)\) an \((S, T)\)-cut in \(G\).
Theorem
Running Ford-Fulkerson on the \((s', t')\)-flow network \(G'\) will produce a maximum flow, minimum-cut pair which corresponds to a maximum-flow, minimum-cut pair for the \((S, T)\)-flow network \(G\) with the same flow value and cut capacity.