CS 341: ALGORITHMS

Lecture 17: max flow

Readings: CLRS 26.2

Trevor Brown

https://student.cs.uwaterloo.ca/~cs341
trevor.brown@uwaterloo.ca
QUICK REVIEW OF LAST TIME
RECALL: MAX-FLOW MIN-CUT THEOREM

- **Theorem 3:** every max $s$-$t$ flow has value equal to the capacity of a min $s$-$t$ cut

- We give an **algorithmic proof** of this theorem
  - (showing that one algorithm solves both max-flow and min-cut at the same time)
FORD-FULKERSON METHOD

Algorithm development
(mixed together with proof of max-flow min-cut theorem)
FORD-FULKERSON METHOD

- Can **undo** previous decisions to improve the flow
- Can effectively “push back” some flow using an **augmenting path** through a **residual graph**

Same Ford as in Bellman-Ford :)

So, what’s the residual graph, how do we find an augmenting path, and how do we improve the flow?
A residual graph $R_f$ is defined for a given flow $f$ and graph $G$. $R_f$ has the same vertices as $G$. For each edge $e = uv$ in $G$,

- If $f(e) < c(e)$, then $R_f$ contains a forward edge $(u, v)$ with the remaining capacity $c(e) - f(e)$.
- If $f(e) > 0$, then $R_f$ contains a backwards edge $(v, u)$ with capacity $f(e)$ representing flow that could be "pushed back".
ANOTHER EXAMPLE RESIDUAL GRAPH

- Recall: for each edge $e = uv$ in $G$,
  - If $f(e) < c(e)$, then $R_f$ contains a **forward** edge $(u, v)$ with the **remaining capacity** $c(e) - f(e)$
  - If $f(e) > 0$, then $R_f$ contains a **backwards** edge $(v, u)$ with **capacity** $f(e)$ representing flow that could be “pushed back”

![Flow f with value 2](image1)

![Residual graph for this flow](image2)
CONTINUING WITH NEW MATERIAL
FORD-FULKERSON METHOD

- Find a **shortest path** $P$ from $s$ to $t$ in the **residual graph**
- If it **improves** the flow, we call it an **augmenting path**
- And use it to **update the flow**

```
Weights:
1 1 1 1 1 1 1

Greedy Flow (f):
0/1 1/1 0/1 0/1 0/1 0/1 0/1

Residual Graph for this Flow:
Forward edge: remaining capacity
Backwards edge: undo some flow
```
FORD-FULKERSON METHOD

- Find a **shortest path** $P$ from $s$ to $t$ in the **residual graph**
- If it **improves** the flow, we call it an **augmenting path**
- And use it to **update the flow**
FORD-FULKERSON METHOD

- Find a **shortest path** $P$ from $s$ to $t$ in the **residual graph**
- If it **improves** the flow, we call it an **augmenting path**
- And use it to **update the flow**

For each **backwards edge** in $P$, decrease existing flow

**Forward edge:** remaining capacity

**Backwards edge:** undo some flow
FORD-FULKERSON METHOD

- Find a **shortest path** $P$ from $s$ to $t$ in the **residual graph**
- If it **improves** the flow, we call it an **augmenting path**
- And use it to **update the flow**

![Diagram showing shortest path and residual graph]

Original greedy path no longer exists

Updated flow

**Forward edge:** remaining capacity

**Backwards edge:** undo some flow
FORD-FULKERSON METHOD

- Find a **shortest path P** from \( s \) to \( t \) in the **residual graph**
- If it **improves** the flow, we call it an **augmenting path**
- And use it to **update the flow**

![Updated flow](image1)

![New residual graph](image2)

No path from \( s \) to \( t \) in residual graph. Done!
IMPROVING A FLOW $f$
GIVEN AN AUGMENTING PATH $P$

- An augmenting path w.r.t a flow $f$ is a **simple** $s$-$t$ path in $R_f$
- Let $P$ be an augmenting path w.r.t $f$
- Let bottleneck($f, P$) be the minimum capacity of an edge in $P$
- We show this subroutine augment($f, P$) always improves the value of flow $f$

```plaintext
augment(f, P)
  let b = bottleneck(f, P)
  for each edge e = (u,v) in P
      if e is a forward edge
          f(e) = f(e) + b
      else if e is a backwards edge
          let e' = (v,u)
          f(e') = f(e') - b
```
LEMMA 4: AUGMENT() IMPROVES FLOW $f$

- Let $f$ be a flow in $G$ with $f^{\text{in}}(s) = 0$, and $P$ be an augmenting path w.r.t. $f$
- Let $f'$ be the resulting flow after running $\text{augment}(f, P)$
- Then $f'$ is a flow with $\text{value}(f') = \text{value}(f) + \text{bottleneck}(f, P)$
- That is, $\text{augment}(f, P)$ increases the flow by $\text{bottleneck}(f, P)$
PROOF

- Claim: \( \text{augment}(f, P) \) increases the flow by \( \text{bottleneck}(f, P) \)
- First check \( f' \) is a flow
  - Prove capacity and conservation constraints, and \( f'^{\text{in}}(s) = 0 \)
- Are capacity constraints satisfied?
  - We add/subtract \( \text{bottleneck}(f, P) \) to/from each edge
  - And \( \text{bottleneck}(f, P) \) is the minimum of the smallest remaining capacity, and the current flow
  - So capacity constraints are satisfied
PROOF

- Claim: augment\((f, P)\) increases the flow by bottleneck\((f, P)\)

**How about conservation of flow?**

- Consider how the flow into and out of each vertex \(u \notin \{s, t\}\) changes as a result of running augment\((f, P)\)

- We show the change in \(f^\text{in}(u)\) is the same as the change in \(f^\text{out}(u)\)

- There are 4 cases, depending on whether the edges entering/leaving \(u\) are **forward** or **backward** edges
**Case 1: forward / forward**

\[
\begin{array}{c}
\text{flow } f \\
\begin{array}{c}
\text{in} \\
\mathbf{u} \\
\mathbf{out}
\end{array}
\end{array}
\]

\[
\begin{align*}
f_{\text{in}}(u) &= 5 \\
f_{\text{out}}(u) &= 5
\end{align*}
\]

\[
\begin{array}{c}
\text{residual graph } R_f \\
\end{array}
\]

\[
\begin{align*}
f'(\text{in}) &= 5 + b \\
f'(\text{out}) &= 5 + b
\end{align*}
\]

\[
\begin{array}{c}
\text{new flow } f' \\
\text{(after augmenting)}
\end{array}
\]

**Case 2: backwards / backwards**

Both \( f_{\text{in}}(u) \) and \( f_{\text{out}}(u) \) are decreased by \( b \)
Case 3: forward / backwards

Let \( \text{bottleneck}(f, P) = b \)

Added and subtracted \( b \) terms cancel out

Case 4: backwards / forwards is similar.
SHOWING $f^{in}(s) = 0$

- Last step in showing $f'$ is a flow
  - Prove: $s$ still has no flow into it
- Since $f$ is a flow, $f^{in}(s) = 0$
- To get $f'^{in}(s) > 0$, an augmenting path must include an edge into $s$
- But then an augmenting path starts at $s$, then returns to $s$, forming a cycle -- contradiction!
Finally we argue \( \text{value}(f') = \text{value}(f) + \text{bottleneck}(f, P) \)

\( f \) and \( f' \) are flows, so \( \text{value}(f') = f'_{\text{out}}(s) \) and \( \text{value}(f) = f_{\text{out}}(s) \)

We thus show \( f'_{\text{out}}(s) = f_{\text{out}}(s) + \text{bottleneck}(f, P) \)

The augmenting path \( P \) is a simple path (leaving \( s \) exactly once)

And there is no flow into \( s \), so the edge \( e \in P \) leaving \( s \) is a forward edge

This means \( \text{augment}(f, P) \) adds \( \text{bottleneck}(f, P) \) to \( f(e) \)

So \( f'_{\text{out}}(s) = f_{\text{out}}(s) + \text{bottleneck}(f, P) \)
FORD-FULKERSON METHOD

- By Lemma 4, starting from any flow $f$, if we can find an augmenting path $P$ w.r.t $f$ in $R_f$, then we can use $\text{augment}(f, P)$ to improve our flow.

- Ford-Fulkerson does this repeatedly starting from an empty flow.

```plaintext
FordFulkerson(G=(V,E))
for e in E
  f(e) = 0
while there is a simple s-t path P in Rf do
  augment(f, P)
  and update the residual graph Rf
```
What we have proved so far: **augmenting improves flow.**

We *don’t know yet* if

1. we can actually obtain the max flow, or
2. whether max-flow = min-cut.

**MAX-FLOW MIN-CUT THEOREM PROOF**
PROOF STRATEGY

• Claim: when there is **no augmenting path**, there is a **cut with capacity equal to the value of the current flow**.

• Proving this will simultaneously
  • prove the max-flow min-cut theorem,
  • prove correctness of the Ford-Fulkerson method,
  • solve the max flow problem, and
  • solve the min cut problem
PROVING MAX FLOW = MIN CUT

Two directions:
**max flow ≤ min cut** and **max flow ≥ min cut**

We actually proved the ≤ direction already (Lemma 2 last time) when discussing upper bounds for max flow

It remains to prove the ≥ direction.
PROVING MAX FLOW $\geq$ MIN CUT

- Proposition: if $f$ is an $s$-$t$ flow such that there is no $s$-$t$ path in the residual graph $R_f$, then there is an $s$-$t$ cut $S$ s.t. $\text{value}(f) = c_{\text{out}}(S)$.

Understanding the proposition...

If flow value = 2

then cut exists with capacity 2 = flow value

flow $f$

residual graph $R_f$

containing no $s$-$t$ paths
PROVING THE PROPOSITION

Since there is no $s$-$t$ path in $R_f$, there is a subset $S$ of vertices with $s \in S$, $t \notin S$ such that $S$ has no outgoing edges in $R_f$.

What does this statement look like?

---

flow $f$

want to prove: outgoing edges in $G$ from $S$ carry the flow

No outgoing edges in $R_f$ from $S$

residual graph $R_f$
Since there is no $s$-$t$ path in $R_f$, there is a subset $S$ of vertices with $s \in S$, $t \notin S$ such that $S$ has **no outgoing edges** in $R_f$

Claim: $c^{out}(S) = \text{value}(f)$

Consider two types of edges. Type 1:

- $uv$ **exiting** $S$ in $G$ ($uv \in \delta^{out}(S)$ in $G$, $u \in S$, $v \notin S$)
- Since $S$ has no outgoing edge in $R_f$, we know $uv \notin R_f$
- This implies $f(uv) = c(uv)$, as otherwise $uv$ would be a forward edge in $R_f$
PROVING THE PROPOSITION

- **Claim**: $c^{out}(S) = \text{value}(f)$
- Consider two types of edges. Type 2:
  - $uv$ entering $S$ in $G$
    - $(uv \in \delta^{in}(S))$ in $G$, $u \notin S, v \in S$
  - Since $S$ has no outgoing edge in $R_f$, we know there is no edge $vu \notin R_f$
    - (note $vu$ would be directed out of $S$)
  - This implies $f(uv) = 0$, as otherwise $vu$ would be a backwards edge in $R_f$
PROVING THE PROPOSITION

- We just showed
  - For edge $uv$ directed out of $S$, $f(uv) = c(uv)$
  - For edge $uv$ directed into $S$, $f(uv) = 0$
  - So $f^{out}(S) - f^{in}(S) = c^{out}(S) - 0 = c^{out}(S)$
  - This proves the proposition. I.e., given flow $f$, if there are no $s$-$t$ paths in $R_f$, then there is a cut matching the flow

Note this was the last thing remaining to prove the min-cut max-flow theorem, and the correctness of Ford-Fulkerson
TIME COMPLEXITY

of the Ford-Fulkerson method
RUNTIME OF FORD-FULKERSON

- Depends on the implementation

```
1 FordFulkerson(G=(V,E))
2     for e in E
3         f(e) = 0
4
5     while there is a simple s-t path P in Rf do
6         augment(f, P)
7         and update the residual graph Rf
```

- How do we find an augmenting path?

- How many times do we need to augment before we terminate?
RUNTIME OF FORD-FULKERSON

- Assume we find any arbitrary augmenting path $P$, using any technique, in $O(n + m)$ time
- Then every time augment$(f, P)$ is run, we know only that the flow increases
- If capacities are integers, the increase is at least 1
- In this case, if max flow is $k$ then runtime is $O(k(n + m))$
  - For max flow we assume a connected graph, so this is $O(km)$
  - Very bad if $k$ is large

If capacities are reals (and in particular some are irrational), this may never terminate!
Figure 26.7  (a) A flow network for which FORD-FULKERSON can take $\Theta(E |f^*|)$ time, where $f^*$ is a maximum flow, shown here with $|f^*| = 2,000,000$. The shaded path is an augmenting path with residual capacity 1. (b) The resulting residual network, with another augmenting path whose residual capacity is 1. (c) The resulting residual network.
EDMONDS-KARP APPROACH

- **Use BFS** to find a shortest path (in terms of number of edges) and use that as an augmenting path
- It turns out this always **terminates after** $O(nm)$ augmenting paths
  - (even with real capacities)
- BFS takes $O(n + m)$ time; $O(m)$ since the graph is connected
- **So total runtime is** $O(nm^2)$

There are more sophisticated algorithms with $O(V^2E)$ and even $O(V^3)$ runtimes

(***optional**: CLRS 26.4, 26.5)

In 2022, researchers found an almost linear time algorithm, which leverages techniques from convex optimization and sophisticated data structures