Network Flow

- Maximum Flow: Problem Formulation
- Maximum Flow: Upper Bound
- Maximum Flow: Adding flow along paths
- Residual Network and Augmenting Path
- Ford-Fulkerson Algorithm – Max-Flow-Min-Cut Theorem
- Edmond-Karp Algorithm
- Maximum Flow: Variants and Applications

Imdad ullah Khan

Max Flow: Augmenting Path

An **augmenting path** is a simple $s - t$ path in the residual graph G_f

▷ It is used to augment the flow f

For an augmenting path P in G_f, $bottleneck(P, f) = \min_{e \in P} c'_e$

▷ the minimum residual capacity of any edge on P in G_f

Note c'_e is the residual capacity of the edge e (its capacity in G_f)

Algorithm AUGMENT(P, f) augment flow using a path P in G_f

$b \leftarrow bottleneck(P, f)$
$f' \leftarrow f$
for each edge $e = uv \in P$ **do**
  if e is a forward edge **then**
    $f'_e \leftarrow f_e + b$
  else if e is a backward edge **then**
    $f'_{vu} \leftarrow f_{vu} - b$

Max Flow : The Ford-Fulkerson Algorithm

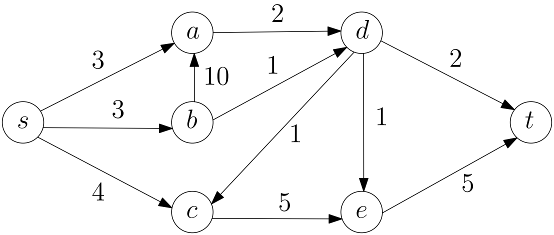
The Ford-Fulkerson algorithm repeatedly augments the current flow until the flow cannot be improved further

Algorithm Ford-Fulkerson Algorithm (G)

$f \leftarrow 0$ ▷ Initialize to a (valid) flow of size 0 (on every edge)
while TRUE **do**
  Compute G_f
  Find an $s - t$ path P in G_f ▷ Using e.g. DFS
  if no such path **then**
    return f
  else
    $f \leftarrow$ AUGMENT(P, f)

Max Flow : The Ford-Fulkerson Algorithm

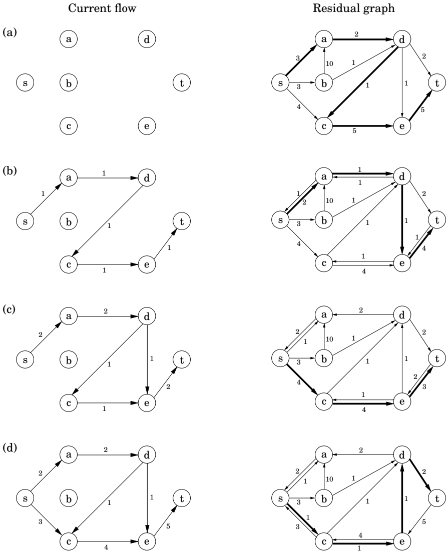
Executing the Ford-Fulkerson Algorithm on the following graph



diagrams taken from the DPV book

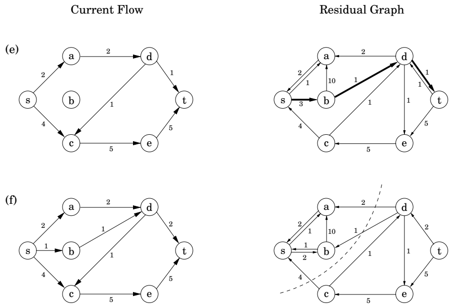
Max Flow : The Ford-Fulkerson Algorithm

Figure 7.6 The max-flow algorithm applied to the network of Figure 7.4. At each iteration, the current flow is shown on the left and the residual network on the right. The paths chosen are shown in bold.



Max Flow : The Ford-Fulkerson Algorithm

Figure 7.6 *Continued*



Max Flow : The Ford-Fulkerson Algorithm

The Ford-Fulkerson algorithm repeatedly augments the current flow

Algorithm Ford-Fulkerson Algorithm (G)

```
f ← 0                          ▷ Initialize to a (valid) flow of size 0 (on every edge)
while TRUE do
    Compute G_f
    Find an s − t path P in G_f                  ▷ Using e.g. DFS
    if no such path then
        return f
    else
        f ← AUGMENT(P, f)
```

- "Correctness" follows from the correctness of the AUGMENT(P, f)
- We need to prove its termination
- We need to discuss its implementation and analyze its running time

The Ford-Fulkerson Algorithm - Analysis

Integrality:

If c_e is integer for every edge e in G, then for every intermediate flow f

- flow on every edge, f_e is integer
- capacity on every edge in G_f, c'_e is integer

Proof : *After iteration i, flow and capacity on all edges are integers*

Basis Step: After iteration 0, $\forall e \in E$, $f_e = 0$ and $c'_e \in \mathbb{Z}$ by construction

Inductive Hypothesis: Before iteration i, $\forall e \in E$, $f_e \in \mathbb{Z}$ and $c'_e \in \mathbb{Z}$

- The capacity b of the bottleneck edge on augmenting path P is integer
- $\forall e \in E \quad f_e^{(i)} = f_e^{(i-1)} \pm b$ $\triangleright$ hence remains $\in \mathbb{Z}$
- Similarly $\forall e \in E$ in $G_{f^{(i)}}$, c'_e (of forward/backward edges) remain $\in \mathbb{Z}$

The Ford-Fulkerson Algorithm - Analysis

Flow is monotonically increasing:

Let f be a flow in G and let P be a $s-t$ path in G_f.
If f' is the flow returned by the $\textsc{Augment}(P, f)$ function, then
$$size(f') = size(f) + b, \quad \text{where } b = bottleneck(P, f)$$

$size(f) \ = \ f^{out}(s)$

- P is $s-t$ path in $G_f \implies$ the first edge e on P is outgoing from s
- This edge sx must be a forward edge in G_f, (because if sx is a backward edge, then $xs \in E(G)$, contradicting $deg^-(s) = 0$ in G
- The AUGMENT procedure will make $f'(e) = f(e) + b$ hence
$$size(f') \ = \ f'^{out}(s) \ = \ f^{out}(s) + b \ = \ size(f) + b$$
- Since $b > 0$, we get that $size(f') > size(f)$

The Ford-Fulkerson Algorithm - Analysis

Termination: We only need to show that max flow is finite (bounded)

The algorithm terminates in at most $C_s = c([\{s\}, \overline{\{s\}}])$ steps

Let f be a flow in G and let $\left[A, \overline{A}\right]$ be any $s - t$ cut in G, then

$$size(f) \;\leq\; c\left(\left[A, \overline{A}\right]\right)$$

$$size(f) \;\leq\; C_s = c([\{s\}, \overline{\{s\}}])$$

In each iteration the flow increases by least an integer $b \geq 1$,

hence there can be at most C_s iterations

The Ford-Fulkerson Algorithm - Analysis

Implementation: $G = (V, E, c), \quad c : E \to \mathbb{Z}^+, \quad |V| = n, \ |E| = m$

The Ford-Fulkerson algorithm can be implemented in $O(mC_s)$ time

Any G_f can have at most $2m$ edges

G_f can be constructed in $O(n + m)$

- We can find $s - t$ path in G_f in $O(n + m)$ ▷ BFS or DFS from s
- AUGMENT(G, f) takes $O(n)$ ▷ incr/dec per edge of P
- At most C_s iterations ▷ flow increased by ≥ 1 in each iteration

This implementation takes $O(mC_s)$ time ▷ assuming $m \geq n$

Max Flow : Upper Bound

Recall the following lemma we proved earlier

Let f be a flow in G and let $[A, \overline{A}]$ be any $s - t$ cut in G, then

$$size(f) \leq c([A, \overline{A}])$$

Tightest upper bound will come from a $s - t$ cut of minimum capacity

$[A^*, \overline{A^*}]$ be an $s - t$ cut with minimum capacity ▷ min-$s - t$-cut

We get the corollary

$$size(f) \leq c([A^*, \overline{A^*}])$$

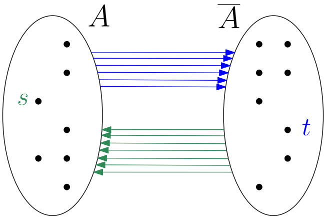
The Ford-Fulkerson Algorithm - Optimality

Let $\left[A^*, \overline{A^*}\right]$ be any $s-t$ cut. By definition we have

$$size(f) = f^{out}(s) = f^{out}(s) + \sum_{s \neq v \in A} \left(f^{out}(v) - f^{in}(v)\right) \qquad \text{just adding 0's}$$

$$= f^{out}(s) + \sum_{s \neq v \in A} f^{out}(v) - \sum_{s \neq v \in A} f^{in}(v)$$

$$= f^{out}(A) - f^{in}(A)$$



$$size(f) = f^{out}(A) - f^{in}(A)$$

$$size(f) = f^{in}(\overline{A}) - f^{out}(\overline{A})$$

The Ford-Fulkerson Algorithm - Optimality

If f is a flow such that there is no $s-t$ path in G_f, then there is a $s-t$ cut $\left[A^*, \overline{A^*}\right]$ in G, such that $size(f) = c\left(\left[A^*, \overline{A^*}\right]\right)$

Construct such a cut. Let $A^* = \mathcal{R}(s)$ in G_f ▷ so $\overline{A^*} = \overline{\mathcal{R}(s)}$

$\left[A^*, \overline{A^*}\right]$ is $s-t$ cut, $s \in A^*$. No $s-t$ path in $G_f \implies t \in \overline{A^*}$

We show that for $e = xy$, $x \in A^*$, $y \in \overline{A^*}$, we have $f_e = c_e$

- If $f_e < c_e$, then $xy \in G_f$ with $c'_e = f_e - c_e > 0$. But then $y \in A^*$

▷ All edges outgoing form A^* are saturated (no capacity left)

Similarly for $e = uv$, $u \in \overline{A^*}$ and $v \in A^*$, we have $f_e = 0$

- If $f_e > 0$, then $vu \in G_f$ with $c'_{vu} = f_e > 0$. But then $u \in A^*$

▷ All edges incoming to A^* are completely unused

$$size(f) = f^{out}(A^*) - f^{in}(A^*) = \sum_{e \text{ outgoing from } A^*} c_e - \sum_{e \text{ incoming to } A^*} 0 = c\left(\left[A^*, \overline{A^*}\right]\right)$$

The Ford-Fulkerson Algorithm: Max-Flow-Min-Cut

Max-Flow-Min-Cut Theorem

If f is a flow with a corresponding cut $\left[A^*, \overline{A^*}\right]$ ($size(f) = c\left(\left[A^*, \overline{A^*}\right]\right)$), then f is a maximum flow and $\left[A^*, \overline{A^*}\right]$ is a minimum cut

Let f be a flow in G and let $\left[A, \overline{A}\right]$ be any $s - t$ cut in G, then

$$size(f) \leq c\left(\left[A, \overline{A}\right]\right)$$

- Immediate corollary to the above Lemma
- If there is flow of larger size than f
- Size of that flow is larger than the cut $c\left(\left[A^*, \overline{A^*}\right]\right)$
- A contradiction to the Lemma
- Similarly a cut of smaller capacity than $\left[A^*, \overline{A^*}\right]$ contradicts the Lemma

The Ford-Fulkerson Algorithm - Optimality

Theorem

The Ford-Fulkerson algorithm returns a maximum flow

Proof:

- It returns a flow f such that G_f has no $s - t$ path
- By the above theorem there is a cut with capacity equal to $size(f)$
- Hence by the Theorem f is optimal

The Ford-Fulkerson Algorithm - Min Cut

Lemma

Given a maximum flow f in a network G, we can compute a minimum $s-t$ cut in G in $O(m)$ steps

Proof:

This minimum cut is given as a bonus

1. Run a DFS or BFS in G_f to find $\mathcal{R}(s)$
2. $[\mathcal{R}(s), \overline{\mathcal{R}(s)}]$ is a min-cut
3. G_f can be computed in $O(n+m) = O(m)$

A BFS or DFS in G_f takes at most $O(n+m) = O(m)$ time