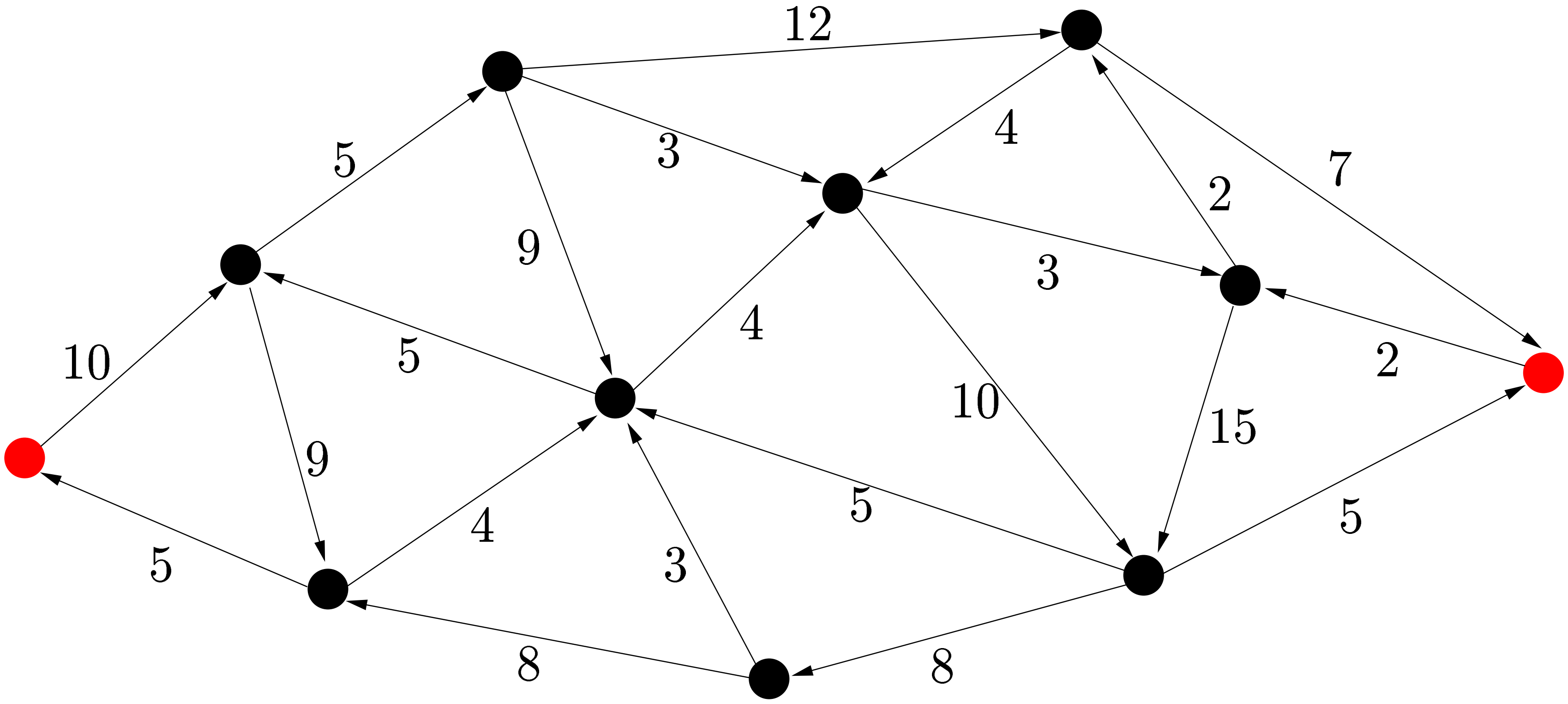
Maximum Network Flow



Networks and Flows

$N = (V, E, c)$ directed network

$G = (V, E)$ directed graph, $c : E \longrightarrow \mathbb{R}^+$ capacity function.

$s, t \in V$, s source, t sink

Legal (s, t)-flow $\qquad f : E \longrightarrow \mathbb{R}$

a) $0 \leq f(e) \leq c(e) \quad \forall\ e \in E$ capacity constraint

b) $\displaystyle\sum_{e \in in(v)} f(e) = \sum_{e \in out(v)} f(e) \quad \forall\ v \in V - \{s, t\}$ conservation laws

$$in(v) = \{\text{edges entering } v\} \qquad out(v) = \{\text{edges leaving } v\}$$

Flow value

If f is a legal flow, the flow value is:

$$val(f) = \sum_{e \in out(s)} f(e) - \sum_{e \in in(s)} f(e)$$

Maximum network flow problem

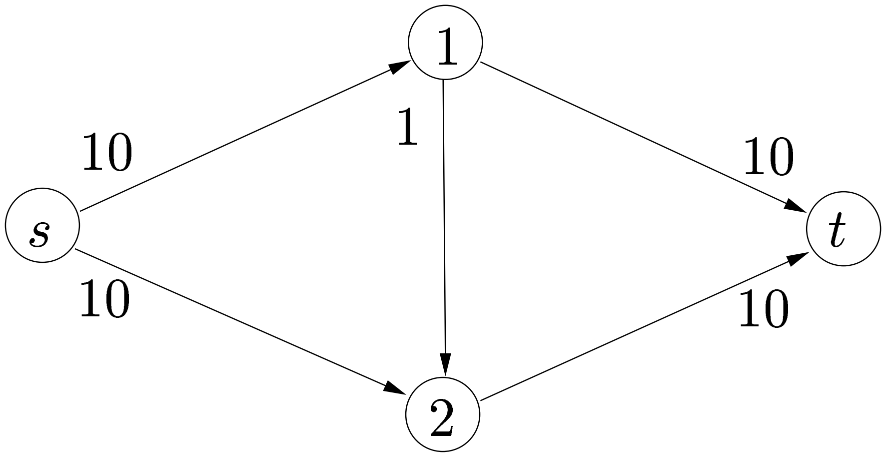
Compute a legal flow with maximum flow value

Cuts

Definition: An (s,t)-cut is a partition S, T of V, i.e. $V = S \cup T$, $S \cap T = \emptyset$, such that $s \in S$, $t \in T$.

Capacity of the cut is

$$c(S,T) = \sum_{e \in E \cap (S \times T)} c(e)$$



Flows and Cuts

Lemma 1: Let f be a legal flow, (S,T) be an (s,t)-cut.

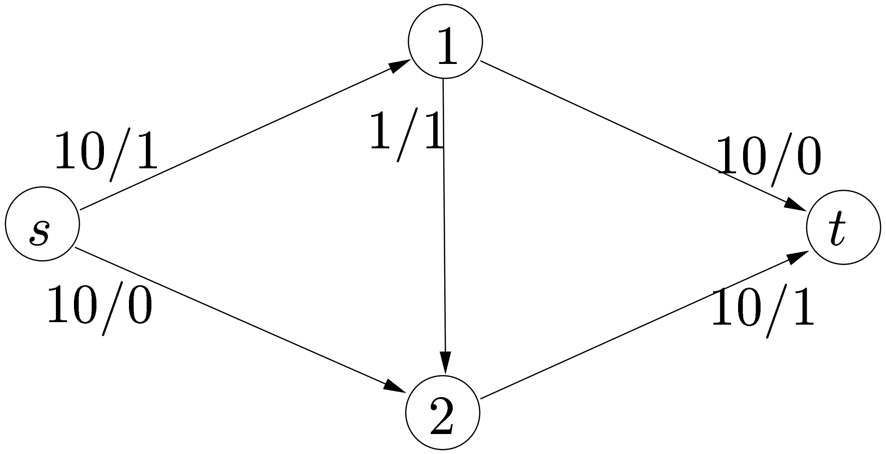
$$val(f) \leq c(S,T)$$

Proof:

$$\begin{aligned}
val(f) &= \sum_{e \in out(s)} f(e) - \sum_{e \in in(s)} f(e) \\
&= \sum_{v \in S} \Big(\sum_{e \in out(v)} f(e) - \sum_{e \in in(v)} f(e) \Big) \\
&= \sum_{e \in E \cap (S \times T)} f(e) - \sum_{e \in E \cap (T \times S)} f(e) \\
&\leq c(S,T) \quad \square
\end{aligned}$$

Algorithm Idea

Augmenting paths: Find paths along which we can increase the flow.



Layered Networks

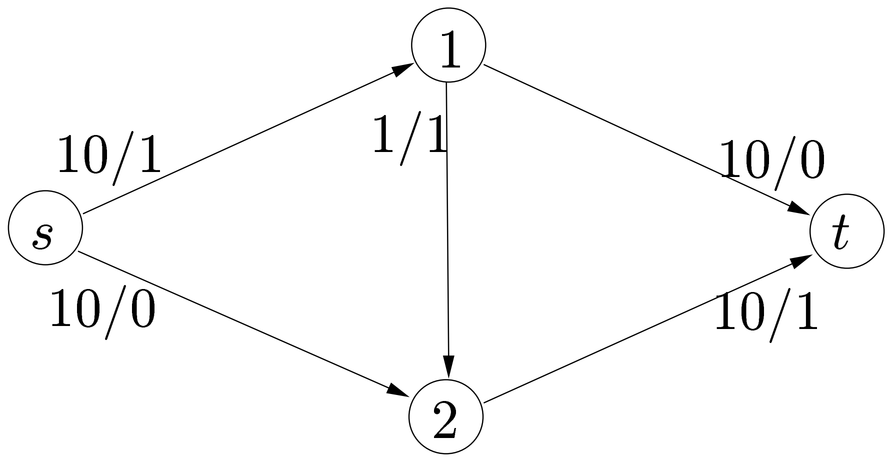
Layered network LN w.r.t. legal flow f

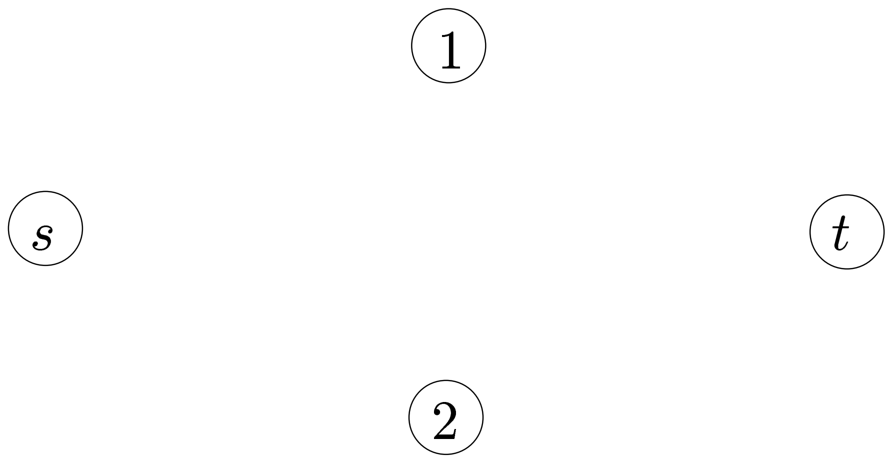
$$
\begin{aligned}
E_1 &= \{(v,w);(v,w)\in E \text{ and } f(e)<c(e)\} \\
E_2 &= \{(w,v);(v,w)\in E \text{ and } f(e)>0\}
\end{aligned}
$$

For $e=(v,w)\in E$ use e_1 to denote $(v,w)\in E_1$ (if it is there)
e_2 to denote $(w,v)\in E_2$ (if it is there)

$\overline{c}: E_1 \cup E_2 \longrightarrow \mathbb{R}^+$

$$
\begin{aligned}
&\overline{c}(e_1)=c(e)-f(e) && \text{for } e_1\in E_1 \\
&\overline{c}(e_2)=f(e) && \text{for } e_2\in E_2
\end{aligned}
$$


1
10/1
1/1
10/0
s
t
10/0
10/1
2

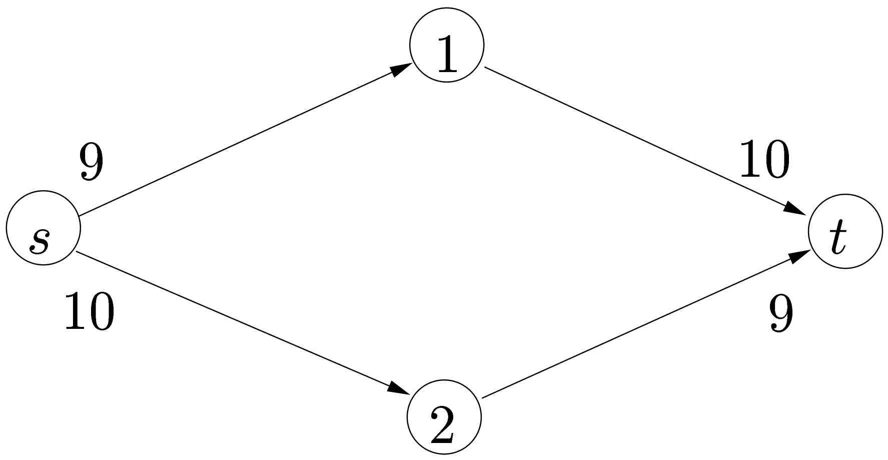

1
s
t
2

$$V_0 = \{s\}$$

$$V_{i+1} = \{w \in V - (V_0 \cup \ldots \cup V_i);\ \exists v \in V_i : (v,w) \in E_1 \cup E_2\} \text{ for } i \geq 1$$

$$\overline{V} = \bigcup_{i \geq 0} V_i$$

$$LN = (\overline{V}, (E_1 \cup E_2) \cap \bigcup_{i \geq 0} (V_i \times V_{i+1}), \overline{c})$$



V_0 V_1 V_2

Lemma 2: f legal (s,t)-flow in N, $LN = (\overline{V}, \overline{E}, \overline{c})$ layered network w.r.t. f.

a) f is maximum flow iff $t \notin \overline{V}$

b) $\overline{f}$ legal (s,t)-flow in LN. Then $f' : E \longrightarrow \mathbb{R}$

$$f'(e) = f(e) + \overline{f}(e_1) - \overline{f}(e_2)$$

is legal flow in N with $val(f') = val(f) + val(\overline{f})$.

Define $\overline{f}(e_i) = 0$ if $e_i \notin \overline{E}$.

Proof: b) Capacity constaints. Let $e \in E$

$$\begin{aligned} 0 &\leq f(e) - \overline{f}(e_2) \\ &\leq f'(e) \\ &\leq f(e) + \overline{f}(e_1) \\ &\leq c(e) \end{aligned}$$

For any node $v \in V$

$$
\begin{aligned}
& \sum_{e \in out(v)} f'(e) - \sum_{e \in in(v)} f'(e) \\
= \quad & \sum_{e \in out(v)} f(e) - \sum_{e \in in(v)} f(e) + \Big(\sum_{e \in out(v)} \overline{f}(e_1) + \sum_{e \in in(v)} \overline{f}(e_2) \Big) \\
& - \Big(\sum_{e \in in(v)} \overline{f}(e_1) + \sum_{e \in out(v)} \overline{f}(e_2) \Big)
\end{aligned}
$$

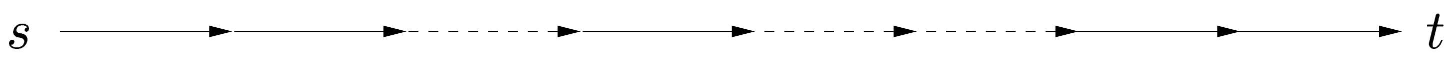
Conservation laws: $v \in V - \{s, t\}$: 0

Value: $v = s$: $val(f') = val(f) + val(\overline{f})$

a) “$\Longrightarrow$”

Assume $t \in \overline{V}$

Then there exists a path P from s to t in LN.



$$\epsilon = \min \ \text{capacity of edges on } P$$

$$\overline{f}(e) = \begin{cases} \epsilon & e \text{ on } P \\ 0 & e \text{ not on } P \end{cases}$$

Thus f not maximal.

“$\Longleftarrow$”

Set $S = \overline{V}$, $\ T = V - S$

$s \in S$, $t \in T$ and (S, T) is an (s, t)-cut.

$(E_1 \cup E_2) \cap (S \times T) = \emptyset$

$$f(e) = c(e) \quad \text{for } e \in S \times T$$

$$f(e) = 0 \qquad \text{for } e \in T \times S$$

$$val(f) = \sum_{e \in E \cap (S \times T)} f(e) - \sum_{e \in E \cap (T \times S)} f(e) = c(S, T)$$

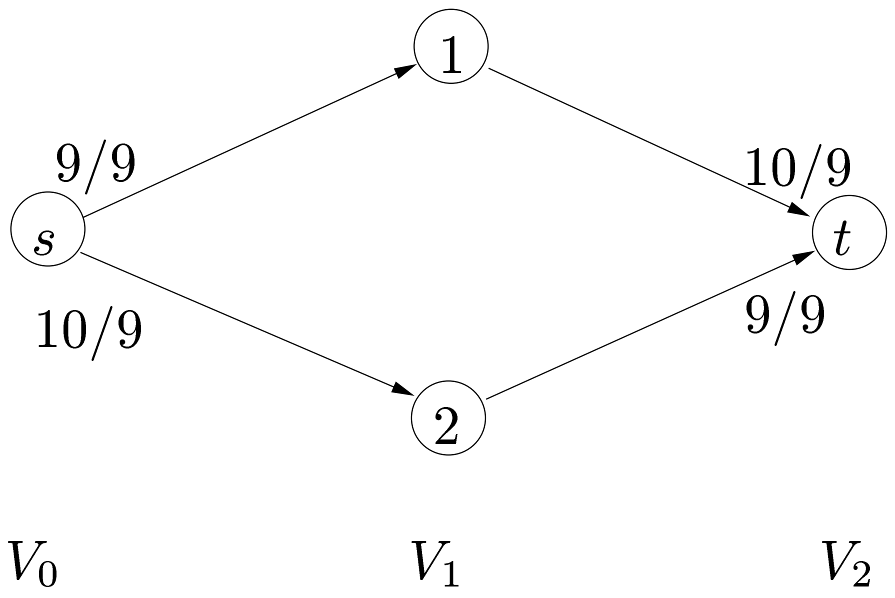
Since $val(g) \leq c(S, T)$ for any legal flow g, f is maximum flow.

Blocking Flow

Definition: Legal flow $\overline{f}$ in layered network LN is blocking if, for every path

$$s = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \xrightarrow{e_3} \quad \cdots \quad \xrightarrow{e_k} v_k = t$$

from s to t, at least one edge is saturated, i.e. $\overline{f}(e_i) = \overline{c}(e_i)$ for at least one i.



Algorithm

1. Let $f(e) \longleftarrow 0$ for all $e \in E$;
2. Construct layered network $LN = (\overline{V}, \overline{E}, \bar{c})$ from f;
3. **while** $t \in \overline{V}$ **do**
4. Find a blocking flow $\overline{f}$ in LN;
5. Update f by $\overline{f}$ as described in Lemma 2b);
6. Construct layered network LN from f
7. **endwhile**.

How can we find a blocking flow?

How many iterations?

Definition: f legal flow in N, LN layered network for f. The depth of LN is the k such that $t \in V_k$.

Lemma 3: Let k_i be the depth of layered network in i-th iteration. Then $k_i > k_{i-1}$, for $i \geq 2$.

Proof: LN_i layered network in i-th iteration.
There exists path P from s to t of length k_i

$$s = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \xrightarrow{e_3} \cdots \xrightarrow{e_{k_i-1}} v_{k_i-1} \xrightarrow{e_{k_i}} v_{k_i} = t$$

$$\begin{aligned} d_j &= \text{layer number of } v_j \text{ in } LN_{i-1},\ 0 \leq j \leq k_i \\ d_j &= \infty \ \text{ if } v_j \text{ not a node in } LN_{i-1} \end{aligned}$$

Claim: For all $i \geq 2$

a) If there is an edge from v_{j-1} to v_j in LN_{i-1}, then $d_j = d_{j-1} + 1$.

b) If there is no edge from v_{j-1} to v_j in LN_{i-1}, then $d_j \leq d_{j-1}$.

c) $k_{i-1} < k_i$

Proof: a) Obvious

b) Assume $d_j \geq d_{j-1} + 1$

f_{i-1} flow that gives rise to LN_{i-1}

f_i flow that gives rise to LN_i

$$\begin{aligned}
\text{If } (v_{j-1}, v_j) \in E \quad &\Longrightarrow \quad f_{i-1}(v_{j-1}, v_j) = c(v_{j-1}, v_j) \\
\text{If } (v_j, v_{j-1}) \in E \quad &\Longrightarrow \quad f_{i-1}(v_j, v_{j-1}) = 0
\end{aligned}$$

Thus $(v_j, v_{j-1}) \in E_{1,i-1} \cup E_{2,i-1}$ and $(v_j, v_{j-1}) \in \overline{E}_{i-1}$.

Hence $d_{j-1} = d_j + 1$ and $d_j = d_{j-1} - 1 < d_{j-1}$.

c) Since $v_0 = s$ and $d_0 = 0$, we have $d_j \leq j$, for $1 \leq j \leq k_i$

Thus $k_{i-1} = d_{k_i} \leq k_i$.

Assume $k_{i-1} = k_i$. Then path P exists in LN_{i-1}.

$$s = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \xrightarrow{e_3} \cdots \xrightarrow{e_{k_i-1}} v_{k_i-1} \xrightarrow{e_{k_i}} v_{k_i} = t$$

Since we compute a blocking flow in LN_{i-1}, there exists a saturated edge

$$v_{j-1} \xrightarrow{e_j} v_j$$

$(v_{j-1}, v_j) \in E \implies f_i(v_{j-1}, v_j) = c(v_{j-1}, v_j)$ and $(v_{j-1}, v_j) \notin \overline{E}_i$

$(v_j, v_{j-1}) \in E \implies f_i(v_j, v_{j-1}) = 0$ and $(v_{j-1}, v_j) \notin \overline{E}_i$ $\square$ $\square$

Corollary: The number of iterations is $\leq n$.

Computing a Blocking Flow

$$LN = (V, E, c), \quad V = \bigcup_{0 \leq i \leq k} V_i, \quad E \subseteq \bigcup_{0 \leq i < k} V_i \times V_{i+1}, \quad V_0 = \{s\}, \quad c : E \longrightarrow \mathbb{R}^+$$

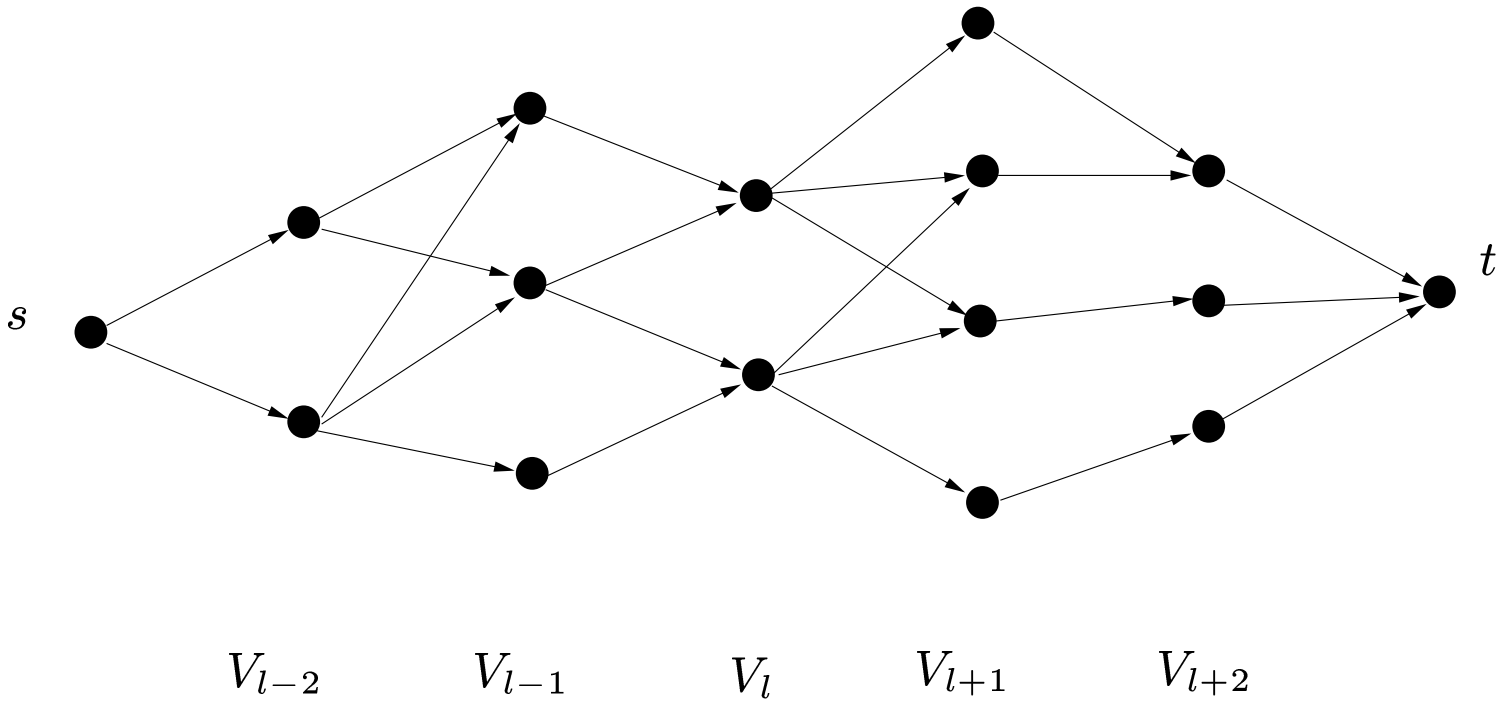
Potential of node v w.r.t. f

$$PO(v) = \min \left\{ \sum_{e \in out(v)} c(e) - f(e), \sum_{e \in in(v)} c(e) - f(e) \right\}$$

$$PO^* = \min\{PO(v) : v \in V\}$$

Choose v with $PO(v) = PO^*$.

Push PO^* units of flow from v to higher layers



Layer V_h: $S_h \subseteq V_h$ that hold PO^* additional units of flow,

$$PO^* = \sum_{x \in S_h} S[x], \quad S[x] = \text{ excess at node } x.$$

Suck PO^* units of flow starting at v from lower layers

Flow increases by PO^* units.

Simplify network by deleting saturated edges and vertices with in- or outdegree 0. (At least one node is deleted.)

Procedure *forward*(x, S, h);

\\\\ x is a node in layer V_h and there are S units of additional flow available in x. These S units are pushed into nodes in layer V_{h+1}.

while $S > 0$ **do**
    Let $e = (x, y)$ be the first edge out of x;
    $\delta \longleftarrow \min(S, c(e) - f(e))$;
    Increase flow along e by δ, add y to S_{h+1} (if it is not already there), increase $S[y]$ by δ, and decrease $c(e)$ by δ;
    $S \longleftarrow S - \delta$;
    if $c(e) = 0$ **then** delete e from the graph **endif**;
endwhile;
Remove x from S_h and set $S[x]$ to zero;
if $(out(x) = \emptyset$ and $x \neq t)$ **or** $(in(x) = \emptyset$ and $x \neq s)$ **then**
    add x to set *del*
endif.

Algorithm Blocking Flow

1. **for all** $x \in V$ **do** $S[x] \longleftarrow 0$ **endfor**;
2. **for all** $l, 0 \leq l \leq k$, **do** $S_l \longleftarrow \emptyset$ **endfor**;
3. $del \longleftarrow \emptyset$;
4. **while** LN is not empty **do**
5. Compute $PO[v]$ for all $v \in V$, let $PO^* = \min\{PO[v]; v \in V\}$ and let $v \in V_l$ be such that $PO^* = PO[v]$;
6. $S[v] \leftarrow PO^*; S_l \longleftarrow \{v\}$;
7. **for** h from l to $k-1$ **do**
8. **for all** $x \in S_h$ **do** $\mathit{forward}(x, S[x], h)$ **endfor**;
9. **endfor**;
10. $S[v] \longleftarrow PO^*; S_l \longleftarrow \{v\}$;
11. **for** h from l downto 1 **do**
12. **for all** $x \in S_h$ **do** $\mathit{suck}(x, S[x], h)$ **endfor**;
13. **endfor**;
14. *simplify(del)*
15. **endwhile**.

Main Result

Theorem 1: A blocking flow in a layered network can be computed in $O(n^2)$ time.

Proof:
1–3: $O(n)$
Loop 4–13: Executed $O(n)$ times, each execution costs $O(n)$ if we ignore *foward, suck, simplify*.
All executions of *forward/suck* take $O(n + e)$.
All executions of *simplify* take $O(n + e)$. $\square$

Theorem 2: A maximum flow can be computed in $O(n^3)$ time.

Proof: A layered network and a blocking flow can be computed in $O(n^2)$ time. $\square$

Max-Flow Min-Cut Theorem

Theorem: Let $N = (V, E, c)$, $\quad s, t \in V$ be network.
f_{max} = max. flow value of any legal (s,t)-flow
c_{min} = min. capacity of all (s,t)-cuts

$$f_{\text{max}} = c_{\text{min}}$$

Proof: f_{max} and c_{min} exist.
By Lemma 1, $f_{\text{max}} \leq c_{\text{min}}$

Let f be flow with $val(f) = f_{\text{max}}$ and $LN = (\overline{V}, \overline{E}, \overline{c})$ be layered network w.r.t. f. Set $S = \overline{V}$ and $T = V - S$.

In the proof of Lemma 2 we showed

$$val(f) = \sum_{e \in E \cap (S \times T)} f(e) - \sum_{e \in E \cap (T \times S)} f(e) = c(S, T) \qquad \square$$