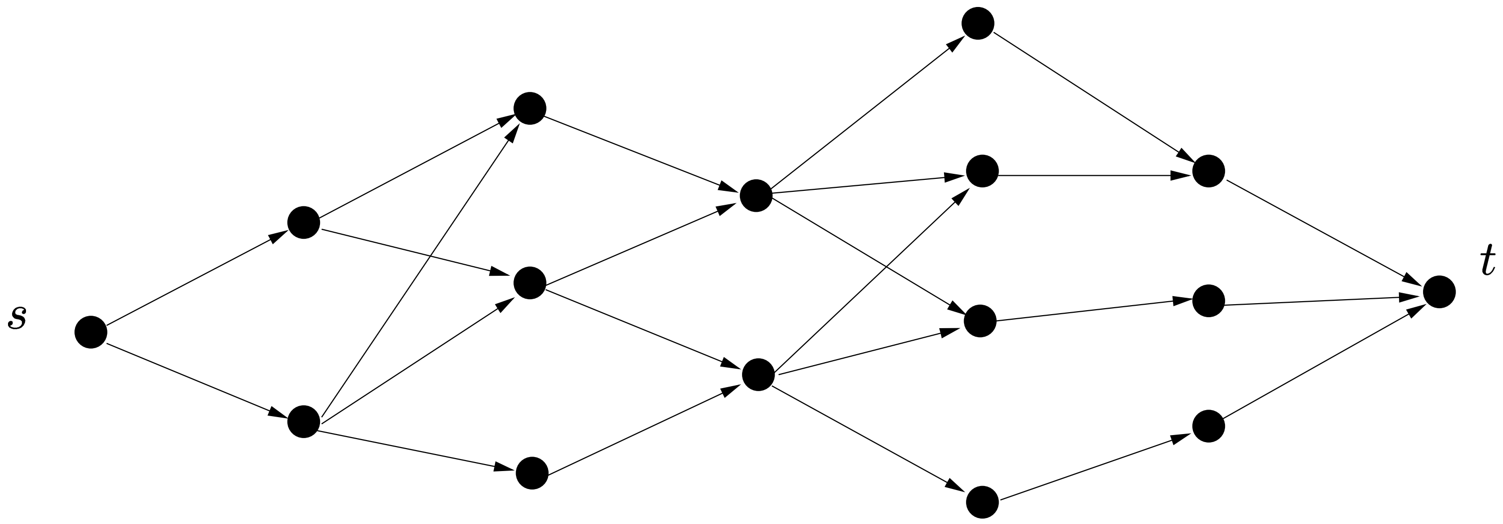
Blocking Flow – DFS Algorithm



Starting at s, always take first outgoing edge out of every node until

(a) t is reached (b) dead-end v is reached

(a) Determine bottleneck capacity ϵ, increase flow along path by ϵ, decrease capacities by ϵ, delete saturated edges.

(b) Back-up one node, delete v and edges leading into v

Analysis

Theorem 3: A blocking flow can be computed in $O(ne)$ time.

Proof: k = depth of the network

A path is constructed in time
$O(k + \#\text{edges found to be ending in dead-ends})$

At most e paths are constructed.

Total time: $O(ke + e) = O(ne)$ $\square$

d-Bounded Networks

Definition: $d \in \mathbb{N}$. $N = (V, E, c)$ is d-bounded if $c(e) \in \{1, 2, \ldots, d\}$ for all $e \in E$. A 1-bounded network is called $(0, 1)$-network.

Applying our flow algorithms to d-bounded networks

$\Longrightarrow$ all intermediate flows are integral, i.e. $f(e) \in \mathbb{N}_0$

$\Longrightarrow$ maximum flow is integral

Theorem 4: A blocking flow in a d-bounded network can be computed in $O(de)$ time. For $d = 1$, we have $O(e)$ time.

Proof: Time to construct a path:

O(#edges on s-t-path + #edges found to be ending in dead ends)

Each edge is contained in at most d paths. $\square$

Improving the Number of Phases

Augmenting networks: Network $N = (V, E, c)$, $s, t \in V$, f legal flow

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

Define

$$AN = (V, E_1 \cup E_2, \overline{c})$$

$$
\begin{aligned}
&\overline{c}(v,w) = c(v,w) - f(v,w) && \text{for } (v,w) \in E_1 \\
&\overline{c}(w,v) = f(v,w) && \text{for } (w,v) \in E_2
\end{aligned}
$$

Constructing layered network w.r.t. N and f

is the same as

constructing layered network w.r.t. AN and $f \equiv 0$

Augmenting networks

Lemma 4: Let N be a network and $f_{\max}$ be value of a maximum (s,t)-flow. Let AN be augmenting network and $\overline{f_{\max}}$ value of maximum (s,t)-flow in AN. Then

$$f_{\max} = \overline{f_{\max}} + val(f).$$

Proof: Let $(S, V - S)$ be any (s,t)-cut.
$c(S, V - S)$ capacity of $(S, V - S)$ w.r.t. N
$\overline{c}(S, V - S)$ capacity of $(S, V - S)$ w.r.t. AN

$$\begin{aligned}
\overline{c}(S, V - S) &= \sum_{v \in S, w \in V - S} \overline{c}(v, w) \\
&= \sum_{v \in S, w \in V - S} ((c(v, w) - f(v, w)) + f(w, v))
\end{aligned}$$

$$
\begin{aligned}
&= c(S, V-S) - \sum_{v\in S, w\in V-S} (f(v,w) - f(w,v)) \\
&= c(S, V-S) - (\sum_{v\in S, w\in V-S} f(v,w) - \sum_{w\in V-S, v\in S} f(w,v)) \\
&= c(S, V-S) - val(f)
\end{aligned}
$$

Thus

$$\overline{c_{\min}} = c_{\min} - val(f),$$

where $c_{\min}, \overline{c_{\min}}$ are the minimum capacities of any (s,t)-cut in N and AN.
$\square$

Improved Running Times

Lemma 5: Let N be a d-bounded network. Then the number of phases is $3 \cdot d^{1/3} n^{2/3}$.

Theorem 5: Let N be a d-bounded network. Then a maximum flow can be computed in time $O(d^{4/3} n^{2/3} e)$.

Definition: A network $N = (V, E, c)$ is simple if $indeg(v) = 1$ or $outdeg(v) = 1$ for all $v \in V$.

Theorem 6: Let $N = (V, E, c)$ be a simple $(0, 1)$-network. Then maximum flow can be computed in $O(n^{1/2} e)$ time.

Proof of Theorem 6

Blocking flow can be computed in $O(e)$ time.

We prove that #phases is $O(n^{1/2})$.

$f_{\max}$ = value of maximum (s,t)-flow.
$f_{\max} < n^{1/2}$: $\surd$

Suppose that $f_{\max} \geq n^{1/2}$
Let phase l be the phase which increases the flow to $\geq f_{\max} - n^{1/2}$

We show that layered network in phase l has depth $\leq n^{1/2}$.

f legal (s,t)-flow just prior to phase l, AN augmenting network w.r.t. f.

Proof of Theorem 6

Claim: AN is simple network.

Proof: Consider $v \in V$ and assume $indeg(v) = 1$ $\quad$ ($outdeg(v) = 1$ similar)

If $f(e) = 0$ for $e \in in(v)$, then $f(e') = 0$ for all $e' \in out(v)$ and v has indegree 1 in AN.

If $f(e) = 1$ for $e \in in(v)$, then $f(e') = 1$ for exactly one $e' \in out(v)$ and v has indegree 1 in AN. $\square$

Proof of Theorem 6

By Lemma 4, AN permits flow $\overline{f}$ with

$$\overline{f_{\max}} \;=\; f_{\max} - val(f) \geq f_{\max} - (f_{\max} - n^{1/2}) = n^{1/2}$$

W.l.o.g. $\overline{f}$ is integral, i.e. $\overline{f}(e) \in \{0, 1\}$.

AN simple and hence one unit of flow can go through each vertex $v \in V - \{s, t\}$.

$\overline{f}$ consists of $n^{1/2}$ vertex-disjoint paths from s to t, one having at most $n^{1/2}$ intermediate vertices. $\square$

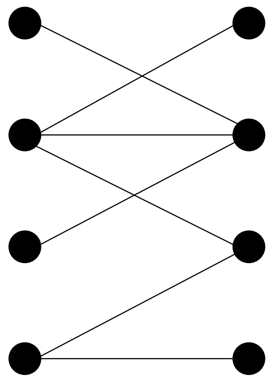
Matching in Bipartite Graphs

$G = (V, E)$ undirected graph.

Matching M is a set of edges $M \subseteq E$ such that no two edges $e_1, e_2 \in M$, $e_1 \neq e_2$ share a common endpoint.

Maximum matching is a matching of maximum cardinality.

$G = (V, E)$ undirected graph is called bipartite if $V = V_1 \cup V_2$ for $V_1, V_2 \subseteq V$ with $V_1 \cap V_2 = \emptyset$.



Matchings in Bipartite Graphs

Theorem 7: Let $G = (V_1 \cup V_2, E)$, $E \subseteq V_1 \times V_2$, be a bipartite graph. Then a maximum matching can be computed in $O(n^{1/2}e)$ time.

Proof: Construct simple network $N = (V_1 \cup V_2 \cup \{s,t\}, \overline{E}, c)$.
All capacities are 1.