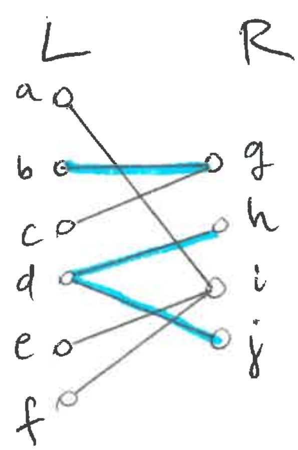
Maximum bipartite matching

- Reference: CLRS, chapter 26

- An undirected graph $G=(V,E)$ is bipartite if the set of vertices can be partitioned into $V = L \cup R$, where L and R are disjoint, and all edges in E connect a vertex in L to a vertex in R.

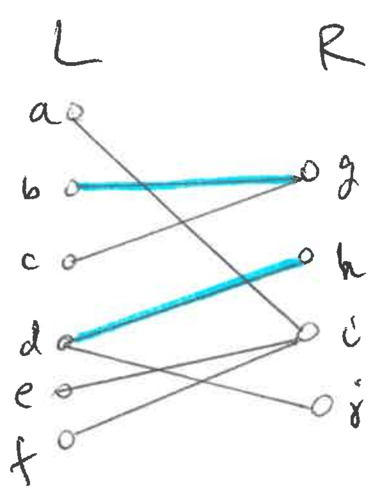
- Given an undirected bipartite graph $G(V,E)$, a matching is a subset of edges $M \subseteq E$ such that for all vertices $v \in V$, at most one edge of M is incident on v.
- Application: find the max number of tasks performed by a set of machines

examples



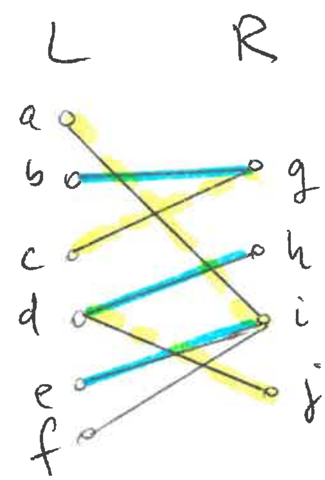
graph G

$\{(b,g),(d,h),(d,j)\}$ is not a valid matching.



graph G

$M=\{(b,g),(d,h)\}$ is a matching of cardinality (or size) 2



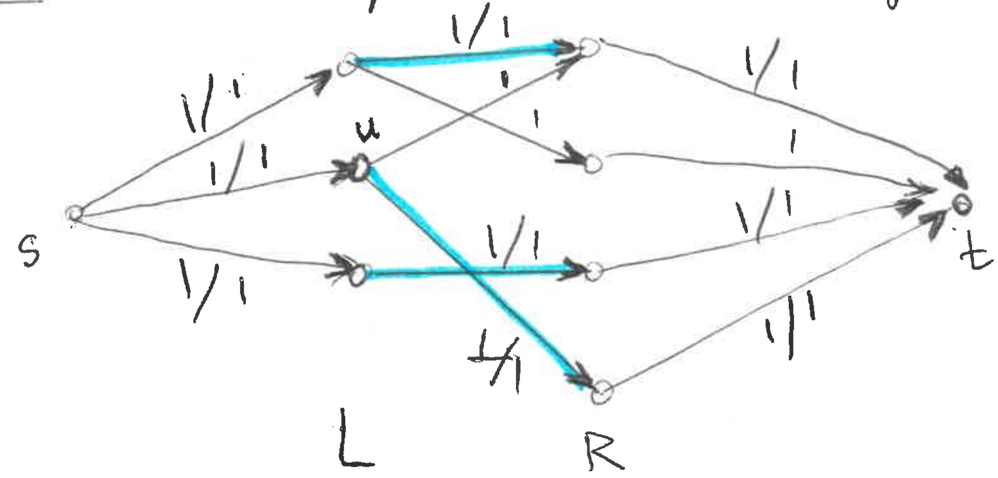
graph G

- $M=\{(b,g),(d,h),(e,i)\}$ is a matching of size 3
- $\{(c,g),(a,i),(d,j)\}$ is another matching of size 3

Problem definition

Given a bipartite graph $G=(V,E)$, find a maximum matching.

Algorithm - model the problem as a max-flow problem



$|f| = 3 \Rightarrow$ matching of cardinality 3

Input
bipartite graph $G(V, E)$
$V = L \cup R$

$\longrightarrow$

Flow network
(standard form)

- build a directed graph $G'(V', E')$
  - set $V' = V \cup \{s, t\}$ where s is the source and t is the sink
  - add the edges in E with direction $L \rightarrow R$
  - add edges connecting s to each vertex in L
  - add edges connecting each vertex in R to t
- add capacity 1 on each edge

$\downarrow$

Max-flow solver

(Ford-Fulkerson or Edmonds-Karp)

$\longleftarrow$ Max-flow f $\longleftarrow$

return the edges in G with flow > 0 as being the edges of the maximum matching M

- G has a maximum matching of size K iff the corresponding flow network G' has a max-flow with value K

RT analysis

Ford-Fulkerson $\Rightarrow RT = O(E' \cdot |f^*|)$

$$|f^*| = O(V)$$

$$|E'| = |E| + |V|$$

- assuming that each vertex in V has at least one edge incident in E $\Rightarrow |E| \geq \frac{|V|}{2} \Rightarrow |V| \leq 2 \cdot |E|$

$$|E'| = |E| + |V| \leq |E| + 2 \cdot |E| = 3 \cdot |E|$$

$$\boxed{RT = O(E \cdot V)}$$

polynomial RT

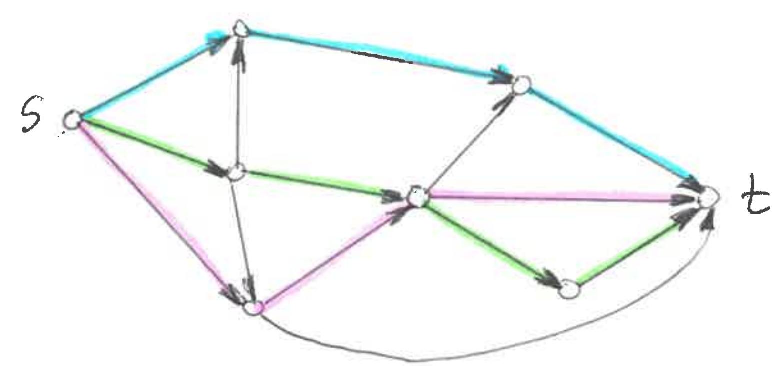
Disjoint paths in directed graphs

- Reference: Algorithm Design, J. Kleinberg and E. Tardos, chapter 7

A set of paths is edge-disjoint if no two paths share an edge, even though multiple paths may go through same vertices.

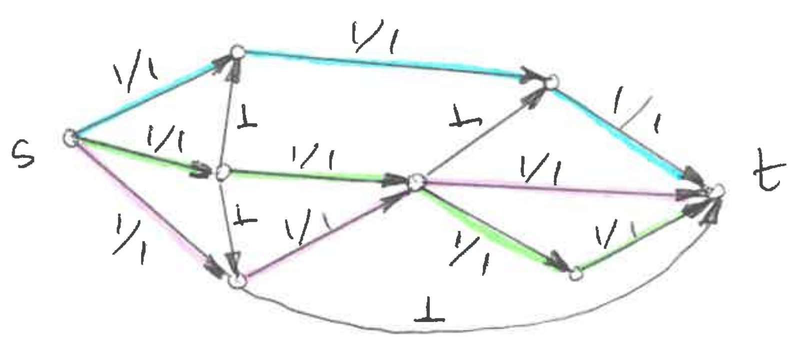
Problem definition

Given a directed graph $G=(V,E)$ and two distinguished vertices $s,t \in V$, find a maximum number of edge-disjoint paths from s to t in G.



Solution

- model the problem as a max-flow problem
- we have all the elements of the flow network, except the capacity of the edges
- idea: assign each edge capacity 1. Based on the integrality theorem, the flow of each edge is an integer $\in \{0,1\}$. A flow of 0 means that the edge is not part of a path, while a flow value 1 means that the edge belongs to an edge-disjoint path.



Input
- graph $G(V,E)$ directed
- vertices s and t

$\longrightarrow$

Flow network
- graph $G(V,E)$ directed
- source s, sink t
- capacity 1 on all edges

$\longrightarrow$ Max-flow solver (Ford-Fulkerson) $\longrightarrow$ Max-flow f $\longrightarrow$ starting from s, follow the flow with value >0 to return the disjoint paths

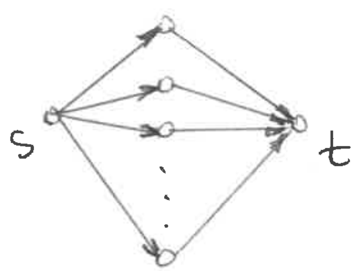
- The maximum number of edge-disjoint paths in a directed graph G from s to t is k if and only if the maximum flow is k.

RT analysis

Ford-Fulkerson takes $O(E \cdot |f^*|)$

$$|f^*| \leq |V| - 2$$

$$|f^*| = O(V)$$



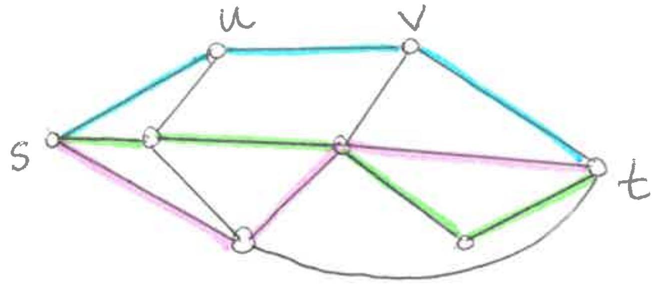
$$\boxed{RT = O(E \cdot V)}$$

polynomial RT

Disjoint paths in undirected graphs

Problem definition

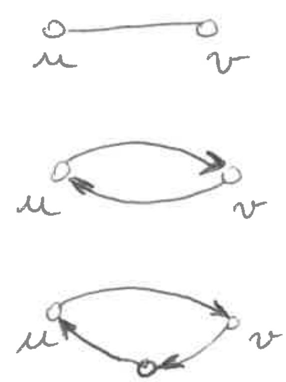
Given an undirected graph $G=(V,E)$ and two distinguished vertices $s,t \in V$, find a maximum number of edge-disjoint paths from s to t in G.



Solution

- model the problem as a max-flow problem
- for a standard flow network, we need a <u>directed</u> graph and edge capacities
- transform G to a directed graph G'
  - an edge $(u,v) \in E$ can mean an edge $u \to v$ or an edge $v \to u$

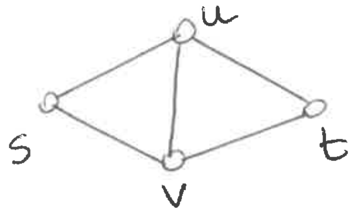
  <u>Steps</u>:
  - for each undirected edge (u,v) in G, add two directed edges (u,v) and (v,u)
  - remove the edges into s and out of t
  - apply the rule for antiparallel edges



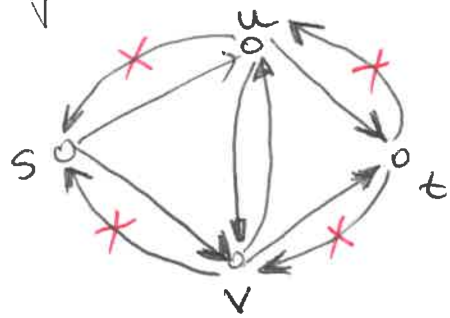
- add capacity 1 on all edges
- strategy: compute a maximum-flow of G', which is mapped to disjoint paths in G', which are mapped to disjoint paths in G.
- Issue: two paths P_1 and P_2 may be edge-disjoint in the directed graph G' and yet share an edge in the undirected graph G (e.g. P_1 uses (u,v) and P_2 uses (v,u)).

example

graph $G(V,E)$

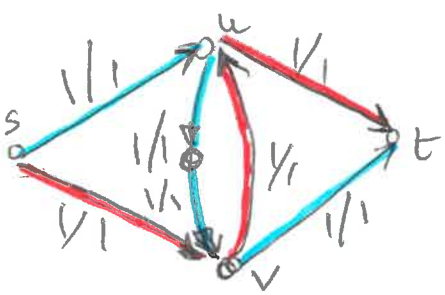


start building the flow network G'



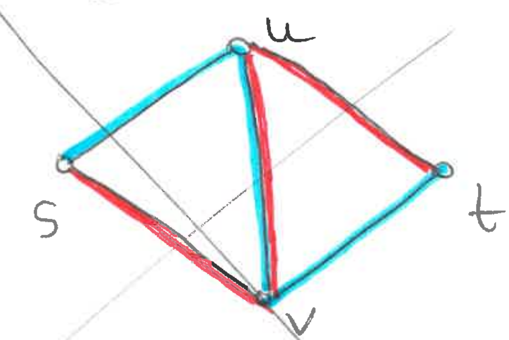
• remove antiparallel edges
· add capacity 1

flow network G'



$|f| = 2$

graph G

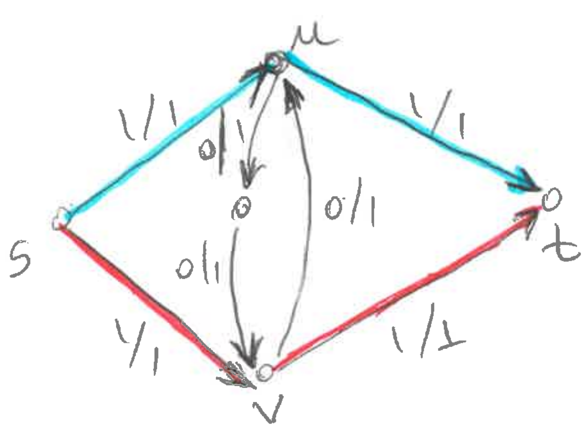


solution is not correct!
Paths cannot share edges!

• adjust (decrease) flow on antiparallel edges such that the flow in one direction becomes 0

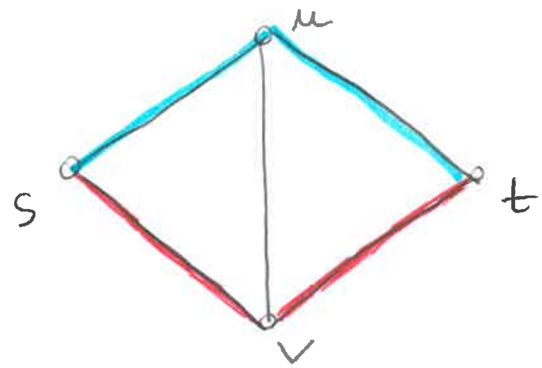
$\sigma = \min\{f(u,v), f(v,u)\} = \min\{1,1\} = 1$

decrease both $f(u,v)$ and $f(v,u)$ by σ



$|f| = 2$

graph G



2 edge-disjoint paths

Algorithm

Input
- graph $G(V,E)$ undirected
- vertices s, t

→

Flow network (standard form)
- build directed graph $G'(V', E')$
  - replace each edge (u,v) by two directed edges

  - remove edges into s and out of t
  - remove antiparallel edges

- source s, sink t
- capacity 1 on all edges

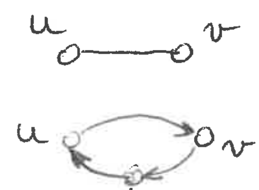
→ Max-flow solver (Ford-Fulkerson) → Max-flow f → adjust flow on antiparallel edges such that flow in one direction is 0 → starting from s, follow the flow with value >0 to return the disjoint paths in G' → map the edge-disjoint paths in G' to the corresponding edge-disjoint paths in G

RT analysis

Ford-Fulkerson takes $O(E' \cdot |f^*|)$

$|f^*| \le |V| - 2 \implies |f^*| = O(V)$

$|E'| \le 3 \cdot |E| \implies |E'| = O(E)$



$\boxed{RT = O(E \cdot V)}$

polynomial RT