Applications of Network Flow

CS 3AC3

Ryszard Janicki

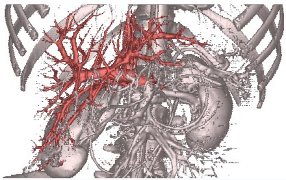
Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada

Acknowledgments: Material based on *Algorithm Design* by Jon Kleinberg and Éva Tardos (Chapter 7)

Max-flow and min-cut applications

Max-flow and min-cut are widely applicable problem-solving model.

- Data mining.
- Open-pit mining.
- Bipartite matching.
- Network reliability.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
- Distributed computing.
- Security of statistical data.
- Egalitarian stable matching.
- Network intrusion detection.
- Multi-camera scene reconstruction.
- Sensor placement for homeland security.
- Many, many, more.

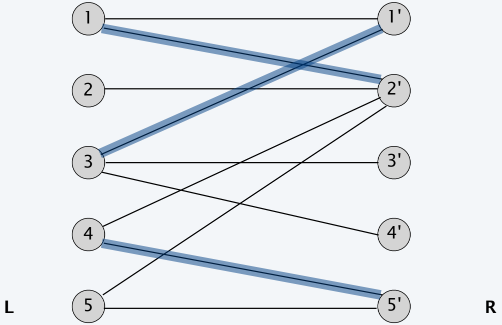


liver and hepatic vascularization segmentation

Bipartite matching

Def. A graph G is bipartite if the nodes can be partitioned into two subsets L and R such that every edge connects a node in L to one in R.

Bipartite matching. Given a bipartite graph $G = (L \cup R, E)$, find a max cardinality matching.

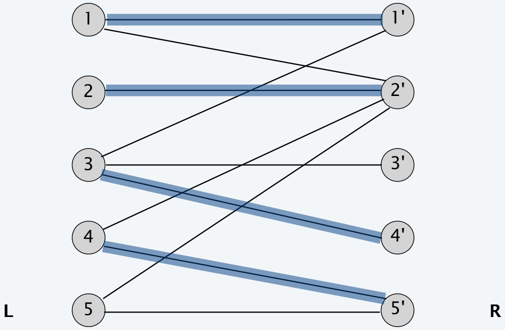


matching: 1–2', 3–1', 4–5'

Bipartite matching

Def. A graph G is bipartite if the nodes can be partitioned into two subsets L and R such that every edge connects a node in L to one in R.

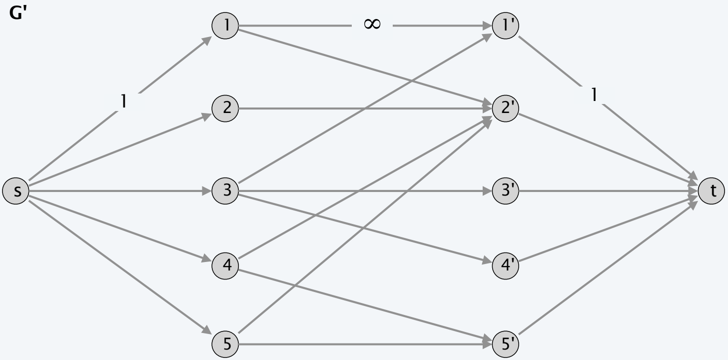
Bipartite matching. Given a bipartite graph $G = (L \cup R, E)$, find a max cardinality matching.



matching: 1–1', 2–2', 3–4', 4–5'

Bipartite matching: max flow formulation

- Create digraph $G' = (L \cup R \cup \{s, t\},\ E')$.
- Direct all edges from L to R, and assign infinite (or unit) capacity.
- Add source s, and unit capacity edges from s to each node in L.
- Add sink t, and unit capacity edges from each node in R to t.

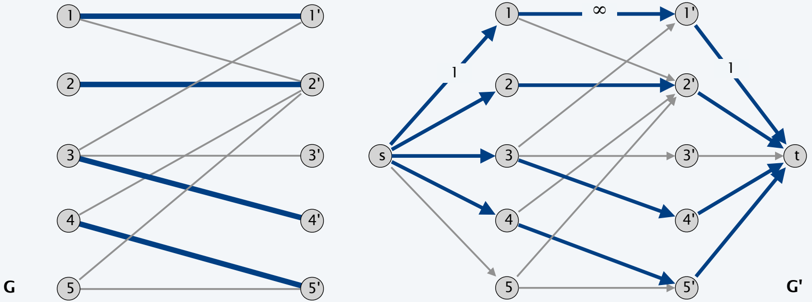


Max flow formulation: proof of correctness

Theorem. Max cardinality of a matching in G = value of max flow in G'.

Pf. $\leq$

- Given a max matching M of cardinality k.
- Consider flow f that sends 1 unit along each of k paths.
- f is a flow, and has value k. ▪

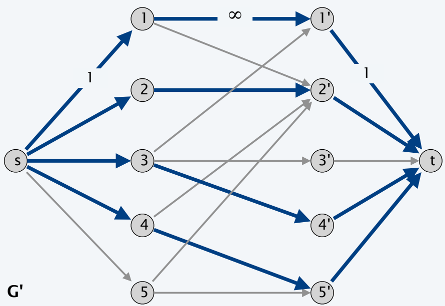


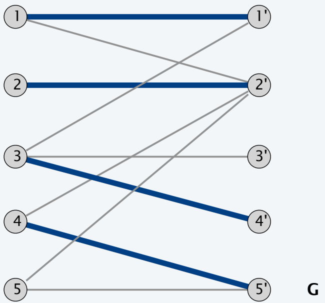
Max flow formulation: proof of correctness

Theorem. Max cardinality of a matching in G = value of max flow in G'.

Pf. $\geq$

- Let f be a max flow in G' of value k.
- Integrality theorem $\Rightarrow$ k is integral and can assume f is 0-1.
- Consider M = set of edges from L to R with $f(e) = 1$.
  - each node in L and R participates in at most one edge in M
  - $|M| = k$: consider cut $(L \cup s, R \cup t)$ ▪





Perfect matching in a bipartite graph

Def. Given a graph $G = (V, E)$ a subset of edges $M \subseteq E$ is a perfect matching if each node appears in exactly one edge in M.

Q. When does a bipartite graph have a perfect matching?

Structure of bipartite graphs with perfect matchings.

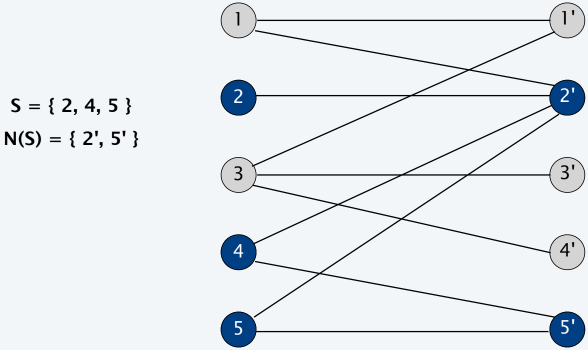
- Clearly we must have $|L| = |R|$.
- What other conditions are necessary?
- What conditions are sufficient?

Perfect matching in a bipartite graph

Notation. Let S be a subset of nodes, and let $N(S)$ be the set of nodes adjacent to nodes in S.

Observation. If a bipartite graph $G = (L \cup R, E)$ has a perfect matching, then $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

Pf. Each node in S has to be matched to a different node in $N(S)$. ▪



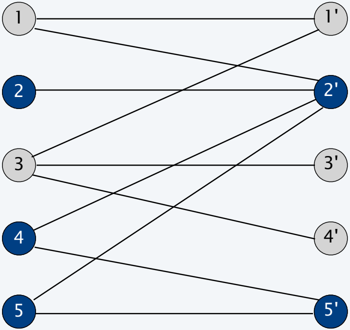
Hall's Theorem

Theorem. Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$.
G has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

Pf. $\Rightarrow$ This was the previous observation.

S = { 2, 4, 5 }

N(S) = { 2', 5' }

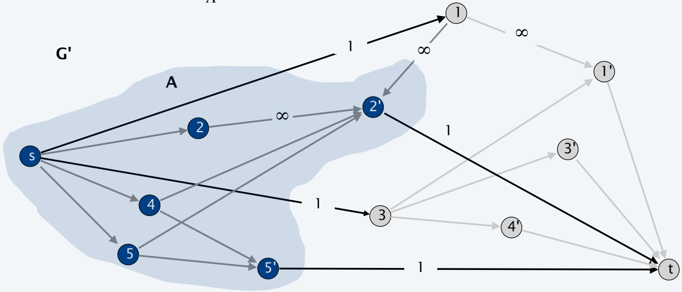


no perfect matching

Proof of Hall's theorem

Pf. $\Leftarrow$ Suppose G does not have a perfect matching.

- Formulate as a max flow problem and let (A, B) be min cut in G'.
- By max-flow min-cut theorem, $cap(A, B) < |L|$.
- Define $L_A = L \cap A$, $L_B = L \cap B$, $R_A = R \cap A$.
- $cap(A, B) = |L_B| + |R_A|$.
- Since min cut can't use ∞ edges: $N(L_A) \subseteq R_A$.
- $|N(L_A)| \leq |R_A| = cap(A, B) - |L_B| < |L| - |L_B| = |L_A|$.
- Choose $S = L_A$. ▪



L_A = {2, 4, 5}
L_B = {1, 3}
R_A = {2', 5'}
$N(L_A)$ = {2', 5'}

Bipartite matching running time

Theorem

The Ford-Fulkerson algorithm solves the bipartite matching problem in $O(mn)$ time, where n is the number of nodes and m is the number of edges.

Airline scheduling

Airline scheduling.

- Complex computational problem faced by nation's airline carriers.
- Produces schedules that are efficient in terms of:
  - equipment usage, crew allocation, customer satisfaction
  - in presence of unpredictable issues like weather, breakdowns
- One of largest consumers of high-powered algorithmic techniques.

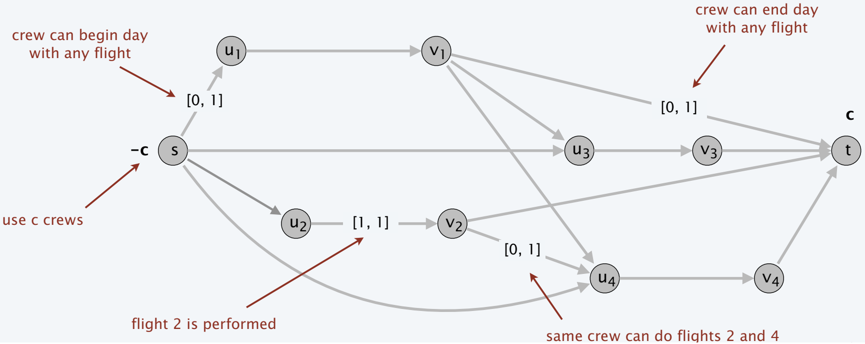
"Toy problem."

- Manage flight crews by reusing them over multiple flights.
- Input: set of k flights for a given day.
- Flight i leaves origin o_i at time s_i and arrives at destination d_i destination at time f_i.
- Minimize number of flight crews.

Airline scheduling

Circulation formulation. [to see if c crews suffice]

- For each flight i, include two nodes u_i and v_i.
- Add source s with demand $-c$, and edges (s, u_i) with capacity 1.
- Add sink t with demand c, and edges (v_i, t) with capacity 1.
- For each i, add edge (u_i, v_i) with lower bound and capacity 1.
- if flight j reachable from i, add edge (v_i, u_j) with capacity 1.



Airline scheduling: running time

Theorem. The airline scheduling problem can be solved in $O(k^3 \log k)$ time.

Pf.

- k = number of flights.
- c = number of crews (unknown).
- $O(k)$ nodes, $O(k^2)$ edges.
- At most k crews needed.

  $\Rightarrow$ solve $\lg k$ circulation problems. ← binary search for optimal value c*
- Value of the flow is between 0 and k.

  $\Rightarrow$ at most k augmentations per circulation problem.
- Overall time = $O(k^3 \log k)$.

Remark. Can solve in $O(k^3)$ time by formulating as minimum flow problem.

Airline scheduling: postmortem

Remark. We solved a toy problem.

Real-world problem models countless other factors:

- Union regulations: e.g., flight crews can only fly certain number of hours in given interval.
- Need optimal schedule over planning horizon, not just one day.
- Deadheading has a cost.
- Flights don't always leave or arrive on schedule.
- Simultaneously optimize both flight schedule and fare structure.

Message.

- Our solution is a generally useful technique for efficient reuse of limited resources but trivializes real airline scheduling problem.
- Flow techniques useful for solving airline scheduling problems (and are widely used in practice).
- Running an airline efficiently is a very difficult problem.