7. **Network Flow II**

- bipartite matching
- disjoint paths
- extensions to max flow
- survey design
- airline scheduling
- image segmentation
- project selection
- baseball elimination

---

**Minimum cut application (RAND 1950s)**

“Free world” goal. Cut supplies (if Cold War turns into real war).

---

**Maximum flow application (Tolstoĭ 1930s)**

Soviet Union goal. Maximize flow of supplies to Eastern Europe.

---

**Max-flow and min-cut applications**

Max-flow and min-cut problems are widely applicable model.

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

- bipartite matching
- disjoint paths
- extensions to max flow
- survey design
- airline scheduling
- image segmentation
- project selection
- baseball elimination

### 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 $L$ with a node in $R$.

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

### 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 unit-capacity edges from $s$ to each node in $L$.
- Add unit-capacity edges from each node in $R$ to $t$.

---

**Matching**

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

**Max matching.** Given a graph $G$, find a max-cardinality matching.
Max-flow formulation: proof of correctness

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

**Pf. ≤**
- Given a max matching $M$ of cardinality $k$.
- Consider flow $f$ that sends 1 unit on each of the $k$ corresponding paths.
- $f$ is a flow of value $k$.

**Pf. ≥**
- Let $f$ be a max flow in $G'$ and let $k$ denote its value.
- Integality 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 $M$
  - $|M| = k$: apply flow-value lemma to cut $(L \cup \{s\}, R \cup \{t\})$.

Perfect matchings in bipartite graphs

**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 $|L| = |R|$.
- Which other conditions are necessary?
- Which other conditions are sufficient?
**Perfect matchings in bipartite graphs**

**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 $N(S)$.

$S = \{2, 4, 5\}$

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

**Hall’s marriage theorem**

**Theorem.** [Frobenius 1917, Hall 1935] Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$. Then, graph $G$ has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

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

$S = \{2, 4, 5\}$

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

**Bipartite matching**

**Problem.** Given a bipartite graph, find a max-cardinality matching.
Network flow II: quiz 2

Which of the following are properties of the graph $G = (V, E)$?

A. $G$ has a perfect matching.
B. Hall's condition is satisfied: $|\mathcal{N}(S)| \geq |S|$ for all subsets $S \subseteq V$.
C. Both A and B.
D. Neither A nor B.

---

Historical significance (Jack Edmonds 1965)

2. Digression. An explanation is due on the use of the words “efficient algorithm.” First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or “code.”

For practical purposes computational details are vital. However, my purpose is only to show as attractively as I can that there is an efficient algorithm. According to the dictionary, “efficient” means “adequate in operation or performance.” This is roughly the meaning I want—in the sense that it is conceivable for maximum matching to have no efficient algorithm. Perhaps a better word is “good.”

I am claiming, as a mathematical result, the existence of a good algorithm for finding a maximum cardinality matching in a graph.

There is an obvious finite algorithm, but that algorithm increases in difficulty exponentially with the size of the graph. It is by no means obvious whether or not there exists an algorithm whose difficulty increases only algebraically with the size of the graph.

---

Nonbipartite matching

Problem. Given an undirected graph, find a max-cardinality matching.

- Structure of nonbipartite graphs is more complicated.
- But well understood.
- Blossom algorithm: $O(n^5)$.
- Best known: $O(m n^{1/2})$.


---

Hackathon problem.

- Hackathon attended by $n$ Harvard students and $n$ Princeton students.
- Each Harvard student is friends with exactly $k > 0$ Princeton students; each Princeton student is friends with exactly $k$ Harvard students.
- Is it possible to arrange the hackathon so that each Princeton student pair programs with a different friend from Harvard?

Mathematical reformulation. Does every $k$-regular bipartite graph have a perfect matching?

Ex. Boolean hypercube.

---

2-regular bipartite graph
Section 7.6

7. **Network Flow II**

- bipartite matching
- disjoint paths
- extensions to max flow
- survey design
- airline scheduling
- image segmentation
- project selection
- baseball elimination

### Edge-disjoint paths

**Def.** Two paths are **edge-disjoint** if they have no edge in common.

**Edge-disjoint paths problem.** Given a digraph $G = (V, E)$ and two nodes $s$ and $t$, find the max number of edge-disjoint $s \rightarrow t$ paths.

**Ex.** Communication networks.

**Max-flow formulation.** Assign unit capacity to every edge.

**Theorem.** Max number of edge-disjoint $s \rightarrow t$ paths = value of max flow.

**Pf.**
- Suppose there are $k$ edge-disjoint $s \rightarrow t$ paths $P_1, \ldots, P_k$.
- Set $f(e) = 1$ if $e$ participates in some path $P_j$; else set $f(e) = 0$.
- Since paths are edge-disjoint, $f$ is a flow of value $k$.  

![Diagram](edge-disjoint-paths.png)
**Edge-disjoint paths**

**Max-flow formulation.** Assign unit capacity to every edge.

**Theorem.** Max number of edge-disjoint $s \rightarrow t$ paths = value of max flow.

**Pf.**
- Suppose max flow value is $k$.
- **Integral theorem** $\Rightarrow$ there exists 0–1 flow $f$ of value $k$.
- Consider edge $(s, u)$ with $f(s, u) = 1$.
  - by flow conservation, there exists an edge $(u, v)$ with $f(u, v) = 1$
  - continue until reach $t$, always choosing a new edge
- Produces $k$ (not necessarily simple) edge-disjoint paths. $\blacksquare$

---

**Network connectivity**

**Def.** A set of edges $F \subseteq E$ disconnects $t$ from $s$ if every $s \rightarrow t$ path uses at least one edge in $F$.

**Network connectivity.** Given a digraph $G = (V, E)$ and two nodes $s$ and $t$, find min number of edges whose removal disconnects $t$ from $s$.

---

**Menger’s theorem**

**Theorem.** [Menger 1927] The max number of edge-disjoint $s \rightarrow t$ paths equals the min number of edges whose removal disconnects $t$ from $s$.

**Pf.**
- Suppose the removal of $F \subseteq E$ disconnects $t$ from $s$, and $|F| = k$.
- Every $s \rightarrow t$ path uses at least one edge in $F$.
- Hence, the number of edge-disjoint paths is $\leq k$. $\blacksquare$

---

**Menger’s theorem**

**Theorem.** [Menger 1927] The max number of edge-disjoint $s \rightarrow t$ paths equals the min number of edges whose removal disconnects $t$ from $s$.

**Pf.**
- Suppose max number of edge-disjoint paths is $k$.
- Then value of max flow = $k$.
- Max-flow min-cut theorem $\Rightarrow$ there exists a cut $(A, B)$ of capacity $k$.
- Let $F$ be set of edges going from $A$ to $B$.
- $|F| = k$ and disconnects $t$ from $s$. $\blacksquare$
Network flow II: quiz 3

How to find the max number of edge-disjoint paths in an undirected graph?

A. Solve the edge-disjoint paths problem in a digraph (by replacing each undirected edge with two antiparallel edges).

B. Solve a max flow problem in an undirected graph.

C. Both A and B.

D. Neither A nor B.

Edge-disjoint paths in undirected graphs

Def. Two paths are edge-disjoint if they have no edge in common.

Edge-disjoint paths problem in undirected graphs. Given a graph $G = (V, E)$ and two nodes $s$ and $t$, find the max number of edge-disjoint $s$–$t$ paths.
**Edge-disjoint paths in undirected graphs**

**Max-flow formulation.** Replace each edge with two antiparallel edges and assign unit capacity to every edge.

**Observation.** Two paths \( P_1 \) and \( P_2 \) may be edge-disjoint in the digraph but not edge-disjoint in the undirected graph.

If \( P_1 \) uses edge \((u, v)\) and \( P_2 \) uses its antiparallel edge \((v, u)\)

**Lemma.** In any flow network, there exists a maximum flow \( f \) in which for each pair of antiparallel edges \( e \) and \( e' \): either \( f(e) = 0 \) or \( f(e') = 0 \) or both. Moreover, integrality theorem still holds.

**Pf.** [by induction on number of such pairs]

- Suppose \( f(e) > 0 \) and \( f(e') > 0 \) for a pair of antiparallel edges \( e \) and \( e' \).
- Set \( f(e) = f(e) - \delta \) and \( f(e') = f(e') - \delta \), where \( \delta = \min \{ f(e), f(e') \} \).
- \( f \) is still a flow of the same value but has one fewer such pair.

**More Menger theorems**

**Theorem.** Given an undirected graph and two nodes \( s \) and \( t \), the max number of edge-disjoint \( s \rightarrow t \) paths equals the min number of edges whose removal disconnects \( s \) and \( t \).

**Theorem.** Given an undirected graph and two nonadjacent nodes \( s \) and \( t \), the max number of internally node-disjoint \( s \rightarrow t \) paths equals the min number of internal nodes whose removal disconnects \( s \) and \( t \).

**Theorem.** Given a directed graph with two nonadjacent nodes \( s \) and \( t \), the max number of internally node-disjoint \( s \rightarrow t \) paths equals the min number of internal nodes whose removal disconnects \( t \) from \( s \).
7. Network Flow II

- bipartite matching
- disjoint paths
- extensions to max flow
- survey design
- airline scheduling
- image segmentation
- project selection
- baseball elimination

SECTION 7.7

Network flow II: quiz 4

Which extensions to max flow can be easily modeled?

A. Multiple sources and multiple sinks.
B. Undirected graphs.
C. Lower bounds on edge flows.
D. All of the above.

Multiple sources and sinks

Def. Given a digraph $G = (V, E)$ with edge capacities $c(e) \geq 0$ and multiple source nodes and multiple sink nodes, find max flow that can be sent from the source nodes to the sink nodes.

Multiple sources and sinks: max-flow formulation

- Add a new source node $s$ and sink node $t$.
- For each original source node $s_i$, add edge $(s, s_i)$ with capacity $\infty$.
- For each original sink node $t_j$, add edge $(t_j, t)$ with capacity $\infty$.

Claim. 1–1 correspondence between flows in $G$ and $G'$.
Circulation with supplies and demands

**Def.** Given a digraph \( G = (V, E) \) with edge capacities \( c(e) \geq 0 \) and node demands \( d(v) \), a circulation is a function \( f(e) \) that satisfies:

- For each \( e \in E \): \( 0 \leq f(e) \leq c(e) \) (capacity)
- For each \( v \in V \): \( \sum_{e \text{ in to } v} f(e) - \sum_{e \text{ out of } v} f(e) = d(v) \) (flow conservation)

**Claim.** \( G \) has circulation iff \( G' \) has max flow of value \( D = \sum_{v : d(v) > 0} d(v) = \sum_{v : d(v) < 0} -d(v) \)

**Integrality theorem.** If all capacities and demands are integers, and there exists a circulation, then there exists one that is integer-valued.

**Pf.** Follows from max-flow formulation + integrality theorem for max flow.

**Theorem.** Given \( (V, E, c, d) \), there does not exist a circulation iff there exists a node partition \( (A, B) \) such that \( \sum_{v \in B} d(v) > \text{cap}(A, B) \).

**Pf sketch.** Look at min cut in \( G' \).

Circulation with supplies and demands: max-flow formulation

- Add new source \( s \) and sink \( t \).
- For each \( v \) with \( d(v) < 0 \), add edge \((s, v)\) with capacity \(-d(v)\).
- For each \( v \) with \( d(v) > 0 \), add edge \((v, t)\) with capacity \( d(v)\).

Circulation problem with lower bounds. Given \( (V, E, \ell, c, d) \), does there exist a feasible circulation?
Circulation with supplies, demands, and lower bounds

Max-flow formulation. Model lower bounds as circulation with demands.  
- Send \(l(e)\) units of flow along edge \(e\).
- Update demands of both endpoints.

\[
\begin{array}{c}
\text{lower bound} & \text{upper bound} \\
\text{flow network } G & \text{flow network } G' \\
\text{vs} & \text{ds} & \text{ds} & \text{vs} \\
(\text{v}) & (\text{d}(v)) & (\text{d}(w)) & (\text{d}(v)) + 2 \\
(\text{w}) & (\text{d}(w)) & \text{d}(w) - 2 & \\
\end{array}
\]

Theorem. There exists a circulation in \(G\) iff there exists a circulation in \(G'\). Moreover, if all demands, capacities, and lower bounds in \(G\) are integers, then there exists a circulation in \(G\) that is integer-valued.

Pf sketch. \(f(e)\) is a circulation in \(G\) iff \(f'(e) = f(e) - l(e)\) is a circulation in \(G'\).

Survey design

- Design survey asking \(n_1\) consumers about \(n_2\) products.
- Can survey consumer \(i\) about product \(j\) only if they own it.
- Ask consumer \(i\) between \(c_j\) and \(c_j'\) questions.
- Ask between \(p_j\) and \(p_j'\) consumers about product \(j\).

Goal. Design a survey that meets these specs, if possible.

Bipartite perfect matching. Special case when \(c_i = c_i' = p_j = p_j' = 1\).

Survey design

Max-flow formulation. Model as a circulation problem with lower bounds.  
- Add edge \((i, j)\) if consumer \(j\) owns product \(i\).
- Add edge from \(s\) to consumer \(j\).
- Add edge from product \(i\) to \(t\).
- Add edge from \(t\) to \(s\).
- All demands \(= 0\).
- Integer circulation \(\iff\) feasible survey design.
### 7. NETWORK FLOW II

- bipartite matching
- disjoint paths
- extensions to max flow
- survey design
- airline scheduling
- image segmentation
- project selection
- baseball elimination

---

**Airline scheduling**

**Airline scheduling.**

- Complex computational problem faced by airline carriers.
- Must produce schedules that are efficient in terms of equipment usage, crew allocation, and customer satisfaction.
- 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 $d_i$ at time $f_i$.
- Minimize number of flight crews.

---

**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.

---

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

**Pf.**

- $k = \text{number of flights}$.
- $c = \text{number of crews (unknown)}$.
- $O(k)$ nodes, $O(k^2)$ edges.
- At most $k$ crews needed.
  - solve $\log k$ circulation problems.
  - binary search for optimal value $c^*$
- Value of the flow is between 0 and $k$.
  - 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.

Image segmentation

Image segmentation.
- Divide image into coherent regions.
- Central problem in image processing.

Ex. Separate human and robot from background scene.

Image segmentation

Foreground / background segmentation.
- Label each pixel in picture as belonging to foreground or background.
- \( V = \) set of pixels, \( E = \) pairs of neighboring pixels.
- \( a_i \geq 0 \) is likelihood pixel \( i \) in foreground.
- \( b_j \geq 0 \) is likelihood pixel \( i \) in background.
- \( p_{ij} \geq 0 \) is separation penalty for labeling one of \( i \) and \( j \) as foreground, and the other as background.

Goals.
- Accuracy: if \( a_i > b_j \) in isolation, prefer to label \( i \) in foreground.
- Smoothness: if many neighbors of \( i \) are labeled foreground, we should be inclined to label \( i \) as foreground.
- Find partition \((A, B)\) that maximizes:

\[
\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E} p_{ij}
\]
**Image segmentation**

Formulate as min-cut problem.

- Maximization.
- No source or sink.
- Undirected graph.

Turn into minimization problem.

- Maximizing \( \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E} \rho_{ij} \)

  is equivalent to minimizing

  \[ \left( \sum_{i \in V} a_i + \sum_{j \in V} b_j \right) - \sum_{i \in A} a_i - \sum_{j \in B} b_j + \sum_{(i,j) \in E} \rho_{ij} \]

  or alternatively

  \[ \sum_{j \in B} a_j + \sum_{i \in A} b_i + \sum_{(i,j) \in E} \rho_{ij} \]

**Image segmentation**

Consider min cut \((A, B)\) in \(G'\).

- \(A\) = foreground.

  \[ \text{cap}(A, B) = \sum_{j \in B} a_j + \sum_{i \in A} b_i + \sum_{(i,j) \in E} \rho_{ij} \]

  if \(i\) and \(j\) on different sides, \(\rho_{ij}\) counted exactly once

- Precisely the quantity we want to minimize.

**Image segmentation**

Formulate as min-cut problem \(G' = (V', E')\).

- Include node for each pixel.
- Use two antiparallel edges instead of undirected edge.
- Add source \(s\) to correspond to foreground.
- Add sink \(t\) to correspond to background.

Grabcut image segmentation

**Grabcut.** [Rother–Kolmogorov–Blake 2004]

"GrabCut" — Interactive Foreground Extraction using Iterated Graph Cuts

Carsten Rother\(^1\)  
Vladimir Kolmogorov\(^2\)  
Andrew Blake\(^3\)

Microsoft Research Cambridge, UK

Figure 1: Three examples of GrabCut. The user drags a rectangle loosely around an object. The object is then extracted automatically.
7. **Network Flow II**

- bipartite matching
- disjoint paths
- extensions to max flow
- survey design
- airline scheduling
- image segmentation
- project selection
- baseball elimination

### Section 7.11

**Project selection (maximum weight closure problem)**

**Projects with prerequisites.**

- Set of possible projects $P$: project $v$ has associated revenue $p_v$.
- Set of prerequisites $E$: $(v, w) \in E$ is a prerequisite for $v$.
- A subset of projects $A \subseteq P$ is feasible if the prerequisite of every project in $A$ also belongs to $A$.

**Project selection problem.** Given a set of projects $P$ and prerequisites $E$, choose a feasible subset of projects to maximize revenue.

---

**Project selection: prerequisite graph**

**Prerequisite graph.** Add edge $(v, w)$ if $w$ is a prerequisite for $v$.

---

**Project selection: min-cut formulation**

**Min-cut formulation.**

- Assign a capacity of $\infty$ to each prerequisite edge.
- Add edge $(s, v)$ with capacity $p_v$ if $p_v > 0$.
- Add edge $(v, t)$ with capacity $-p_v$ if $p_v < 0$.
- For notational convenience, define $p_s = p_t = 0$. 
Project selection: min-cut formulation

Claim. \((A, B)\) is min cut \(A - \{ s \}\) is an optimal set of projects.

- Infinite capacity edges ensure \(A - \{ s \}\) is feasible.
- Max revenue because: \(\text{cap}(A, B) = \sum_{v \in B : p_v > 0} p_v + \sum_{v \in A : p_v < 0} (-p_v)

\[= \sum_{v : p_v > 0} p_v - \sum_{v \in A} p_v \]

minimizing this is equivalent to maximizing revenue

Open-pit mining

Open-pit mining. [studied since early 1960s]

- Blocks of earth are extracted from surface to retrieve ore.
- Each block \(v\) has net value \(p_v = \text{value of ore} - \text{processing cost}.
- Can’t remove block \(v\) until both blocks \(w\) and \(x\) are removed.

Baseball elimination

7. NETWORK FLOW II

- bipartite matching
- disjoint paths
- extensions to max flow
- survey design
- airline scheduling
- image segmentation
- project selection
- baseball elimination
Baseball elimination problem

Q. Which teams have a chance of finishing the season with the most wins?

<table>
<thead>
<tr>
<th>i</th>
<th>team</th>
<th>wins</th>
<th>losses</th>
<th>to play</th>
<th>ATL</th>
<th>PHI</th>
<th>NYM</th>
<th>MON</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Atlanta</td>
<td>83</td>
<td>71</td>
<td>8</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Philly</td>
<td>80</td>
<td>79</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>New York</td>
<td>78</td>
<td>78</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Montreal</td>
<td>77</td>
<td>82</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Montreal is mathematically eliminated.
- Montreal finishes with $\leq 80$ wins.
- Atlanta already has 83 wins.

Remark. This is the only reason sports writers appear to be aware of — conditions are sufficient but not necessary!

Baseball elimination problem

Current standings.
- Set of teams $S$.
- Distinguished team $z \in S$.
- Team $x$ has won $w_x$ games already.
- Teams $x$ and $y$ play each other $r_{xy}$ additional times.

Baseball elimination problem. Given the current standings, is there any outcome of the remaining games in which team $z$ finishes with the most (or tied for the most) wins?

Baseball elimination problem: max-flow formulation

Can team 4 finish with most wins?
- Assume team 4 wins all remaining games $\implies w_4 + r_4$ wins.
- Divvy remaining games so that all teams have $\leq w_4 + r_4$ wins.
Baseball elimination problem: max-flow formulation

**Theorem.** Team 4 not eliminated iff max flow saturates all edges leaving s.

**Pf.**
- Integrality theorem ⇒ each remaining game between \( x \) and \( y \) added to number of wins for team \( x \) or team \( y \).
- Capacity on \( (x, i) \) edges ensure no team wins too many games.

Baseball elimination: explanation for sports writers

**Certificate of elimination.**

\[
    T \subseteq S, \quad w(T) = \sum_{i \in T} w_i, \quad g(T) = \sum_{(x, y) \subseteq T} g_{xy},
\]

\[
    \sum_{i \in T} w_i = \sum_{(x, y) \subseteq T} g_{xy}.
\]

**Theorem.** [Hoffman–Rivlin 1967] Team \( z \) is eliminated iff there exists a subset \( T^* \) such that

\[
    w_z + g_z < \frac{w(T^*) + g(T^*)}{|T^*|}.
\]

**Pf.**
- Suppose there exists \( T^* \subseteq S \) such that \( w_z + g_z < \frac{w(T^*) + g(T^*)}{|T^*|} \).
- Then, the teams in \( T^* \) win at least \( (w(T^*) + g(T^*)) / |T^*| \) games on average.
- This exceeds the maximum number that team \( z \) can win.

Detroit is mathematically eliminated.

- Detroit finishes with ≤ 76 wins.
- Wins for \( R = \{ \text{NYY, BAL, BOS, TOR} \} \) = 278.
- Remaining games among \( \{ \text{NYY, BAL, BOS, TOR} \} \) = 3 + 8 + 7 + 7 = 27.
- Average team in \( R \) wins 305/4 = 76.25 games.
Baseball elimination: explanation for sports writers

Pf. ⇒

• Use max-flow formulation, and consider min cut \((A, B)\).
• Let \(T^* = \text{team nodes on source side } A \text{ of min cut.}\)
• Observe that game node \(x-y \in A \text { iff both } x \in T^* \text{ and } y \in T^*\).
• Since team \(z\) is eliminated, by max-flow min-cut theorem,

\[
g(S \setminus \{z\}) > \text{cap}(A, B)
\]

\[
= g(S \setminus \{z\}) - g(T^*) + \sum_{x \in T^*} (w_x + g_z - w_z)
\]

\[
= g(S \setminus \{z\}) - g(T^*) - w(T^*) + |T^*|(w_z + g_z)
\]

• Rearranging terms:

\[
w_z + g_z < \frac{w(T^*) + g(T^*)}{|T^*|}
\]

•