Network Flow IV - Applications II

Sariel Har-Peled
4/7/05

1 Previous Lecture

• Maximum bipartite matching.
• Multiple sources and sinks.
• Edge disjoint paths: directed/undirected.
• Circulations with demands.
• Circulations with demands and lower bounds.
• Application - survey design.

2 Airline Scheduling

Problem 2.1 Given information about possible flights, generate a profitable schedule.

Input Example:

• Boston (depart 6 A.M.) - Washington DC (arrive 7 A.M.).
• Urbana (depart 7 A.M.) - Champaign (arrive 8 A.M.)
• Washington (depart 8 A.M.) - Los Angeles (arrive 11 A.M.)
• Urbana (depart 11 A.M.) - San Francisco (arrive 2 P.M.)
• San Francisco (depart 2:15 P.M.) - Seattle (arrive 3:15 P.M.)
• Las Vegas (depart 5 P.M.) - Seattle (arrive 6 P.M.)

Using same airplane: Same airplane for segments $i$ and $j$ if:

• Destination $i$ = origin of $j$.
• Enough time in between for maintenance.
• Can add flight from dest(i) to origin(j).

**Example 2.2**

• Boston (depart 6 A.M.) - Washington DC (arrive 7 A.M.).
  • Washington (depart 8 A.M.) - Los Angeles (arrive 11 A.M.)
  • **Add:** Los Angeles (depart 12 noon) - Las Vegas (1 P.M.)
  • Las Vegas (depart 5 P.M.) - Seattle (arrive 6 P.M.)

Model the feasibility constraints by a graph.

**Problem 2.3**

• $G$ - directed graph.
  • $i \rightarrow j$: if same airplane can serve both $i$ and $j$.
  • $G$ - acyclic. Indeed, since we can have an edge $i \rightarrow j$ only if the flight $j$ comes after the flight $i$ (in time), it follows that we can not have cycles.
  • Can serve all those flights using $k$ airplanes?

**Reduction to circulation**

Construct a graph $H$ as follows:

• **Each flight on the list must be served:**
  \[ \forall i: e_i = u_i \rightarrow v_i. \]
  Lower bound: $\ell(e_i) = 1$ and capacity: $c(e_i) = 1$.

• **The same plane can perform flight $i$ and $j$:**
  If ($i \rightarrow j \in E(G)$) then
  Add edge ($v_i \rightarrow u_j$) with capacity 1 (no lower bound)

• **Any plane can start the day with flight $i$:**
  \[ \forall i: \text{Add edge } (s \rightarrow u_i) \text{ with capacity 1} \]

• **Any plane can end the day with flight $j$:**
  \[ \forall j: \text{Add edge } (v_j \rightarrow t) \text{ with capacity 1} \]

• **If we have extra planes, we don not have to use them:**
  \[ (s \rightarrow t) \text{ with capacity } k. \]

Q: Is there a valid circulation in $H$?

**Lemma 2.4** There is a way to perform all flights using at most $k$ planes if and only if there is a feasible circulation in the network $H$. 
Proof: Assume there is a way to perform the flights using $k' \leq k$ flights. Every airplane defines a path $\pi$ in the network $H$ that starts at $s$ and ends at $t$, and we send one unit of flow using of flow on each such path. We also send $k - k'$ units of flow on the edge $s \rightarrow t$. Clearly, this is a circulation that satisfies all the constraints.

As for the other direction, consider a feasible circulation in $H$. This is an integer valued circulation. Suppose that $k'$ units of flow are sent between $s$ and $t$ not on the edge $s \rightarrow t$. All edges have capacity 1, and as such the circulation on all other edges is either zero or one. We again convert this into $k'$ paths by repeatedly traversing from the vertex $s$ to the destination $t$, removing the edges we are using in each such path after extracting it. Since we never use an edge twice, and $H$ is acyclic, it follows that we would extract $k'$ paths. Each of those paths correspond to one airplane, and the overall schedule for the airplanes is valid, since all required segments are satisfied.

There are a lot of other considerations that we ignored in the above problem. Like:

- Long term maintenance.
- Crew allocation?
- Scheduling for several days?
- Not hard constraint - maximize revenue.

Conclusion: While network flow is used in practice. However, real world problems are complicated, and network flow can capture only a few aspects.

3 Image Segmentation

The input is an image. Partition into background and foreground.
The input is a bitmap on a grid:

⇒ G:

Input:

- A bitmap on a $N \times N$
- Associated directed graph: $G = (V, E)$.
- $f_i$ - estimate of the likelihood of pixel $i$ to be in foreground.
- $b_i$ - estimate of the likelihood of pixel $i$ to be in background.
- $p_{ij}$ - separation penalty - the price of separating $i$ from $j$
  Only adjacent pixels in the grid pay this.

Problem 3.1 Given input $G = (V, E)$, partition $S$ into $F$ and $B$, s.t.

$$q(F, B) = \sum_{i \in F} f_i + \sum_{i \in B} b_i - \sum_{(i,j) \in E, |F \cap \{i,j\}| = 1} p_{ij}.$$ is maximized.

We can rewrite $q(F, B)$ as:

$$q(F, B) = \sum_{i \in F} f_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E, |F \cap \{i,j\}| = 1} p_{ij}$$

$$= \sum_{i \in F} (f_i + B_I) - \sum_{i \in B} f_i - \sum_{j \in F} b_j - \sum_{(i,j) \in E, |F \cap \{i,j\}| = 1} p_{ij}$$

Max. $q(F, B)$ is equivalent to min $u(F, B)$:

$$u(F, B) = \sum_{i \in B} f_i + \sum_{j \in F} b_j + \sum_{(i,j) \in E, |F \cap \{i,j\}| = 1} p_{ij}$$

Q: How to compute this?

Idea... Minimizing $u(F, B)$ = computing a min-cut in the right graph.

- $G = (V, E)$: “directed” grid.
- Add $s$: source
t: Target.
\[
\forall i \in V \text{ add edge } e_i = s \rightarrow i \\
c(e_i) = a_i
\]

\[
\forall j \in V \text{ add edge } e'_j = j \rightarrow t \\
c(e'_j) = b_j
\]

\( H \) - resulting graph.

**Lemma 3.2** A minimum cut \((F, B)\) in \(H\) minimizes \(u(F, B)\).

Using the min-cut max flow theorem, we have:

**Theorem 3.3** One can solve the segmentation problem by computing the max flow in the graph \(H\).

### 4 Project Selection

- Small company - decide which projects to do.
- \(P\) - possible projects.
- Each project \(i \in P\) has a revenue \(p_i\).
  - \(p_i > 0\) - profit. \(p_i < 0\) - loss.
- Projects have dependencies.
  (i.e., must complete project \(a\), so that you can use it in project \(b\)).
- \(G = (P, E)\). \((i \rightarrow j) \in E\) iff \(j\) is a prerequisite for \(i\).

**Definition 4.1** A set \(A \subset P\) is feasible if for all \(i \in A\), all the prerequisites of \(i\) are also in \(A\). Formally, for all \(i \in A\), and edges \((i, j) \in E\), we have \(j \in A\).

**Profit**: \(\text{profit}(A) = \sum_{i \in A} p_i\).

**Problem 4.2** Select a feasible set of projects maximizing profit.

**Idea...** Reduce the problem to a min-cut in a graph. In a similar fashion to what we did in the image segmentation problem.

**Reduction** Add \(s\) and \(t\) to \(G\):

- \(\forall i \in P, \text{ s.t. } p_i > 0, \text{ add } e_i = (s, i), \text{ set } c(e_i) = p_i\).
- \(\forall j \in P, \text{ s.t. } p_j < 0, \text{ add } e'_j = (j, t), \text{ set } c(e'_j) = -p_j\).
- Bound on max flow: \(C = \sum_{i \in P, p_i > 0} p_i\).
- Set capacity of all other edges to \(C\).
- \(H\) resulting graph.
\begin{itemize}
  \item $A$ - a set of feasible projects.
  \item $A' = A \cup \{s\}$ and $B' = (P \setminus A) \cup \{t\}$.
  \item Consider the $s$-$t$ cut $(A', B')$.
  \item No edge in $E(G)$ is in $(A', B')$ (because it is feasible).
\end{itemize}

\textbf{Lemma 4.3} The capacity of the cut $(A', B')$, as defined by a feasible project set $A$, is $c(A', B') = C - \sum_{i \in A} p_i$.

Proof: The edges of $H$ are either: (i) Edges of $G$, (ii) edges emanating from $s$, and (iii) edges entering $t$. Since $A$ is feasible, it follows that no edges of type (i) contribute to the cut. The edges entering $t$ contribute to the cut the value

$$X = \sum_{i \in A \text{ and } p_i < 0} -p_i.$$ 

The edges leaving the source $s$ contribute

$$Y = \sum_{i \in A \text{ and } p_i > 0} p_i.$$ 

By the definition of $C$, we have

$$Y = \sum_{i \in P, p_i > 0} p_i - \sum_{i \in A \text{ and } p_i > 0} p_i = C - \sum_{i \in A \text{ and } p_i > 0} p_i.$$ 

The capacity of the cut $(A', B')$ is

$$X + Y = \sum_{i \in A \text{ and } p_i < 0} (-p_i) + C - \sum_{i \in A \text{ and } p_i > 0} p_i = C - \sum_{i \in A} p_i,$$

as claimed. \hfill \blacksquare

\textbf{Lemma 4.4} If $(A', B')$ is a cut with capacity at most $C$, then the set $A = A' \setminus \{s\}$ is a feasible set of projects.

\textbf{Conclusion} Cuts $(A', B')$ of capacity $\leq C$ corresponds one-to-one to feasible sets.

\begin{itemize}
  \item $A$ minimizes $\sum_{i \in A} p_i$
  \item Then $A$ maximizes $C - \sum_{i \in A} p_i$
  \item By lemma $\Rightarrow$ cost of min cut in $H$.
\end{itemize}

A set $A$ that maximizes $\sum_{i \in A} p_i$, is a set that minimizes $C - \sum_{i \in A} p_i$. But the cut that minimizes this quantity is the min-cut in $H$.

\textbf{Theorem 4.5} If $(A', B')$ is a minimum cut in $H$ then $A = A' \setminus \{s\}$ is an optimum solution to the project selection problem.
5 Baseball elimination

The teams have the following number of wins: New York: 92. Baltimore: 91, Toronto: 91, Boston: 90
There are 5 games remaining (all pairs except New York and Boston.)

Question: Can Boston win the season? Namely, can Boston finish the season with as many point as anybody else? Boston can get at most 92 wins.

If New York wins then it is over.
So Baltimore must win against New York
So Toronto must win against New York
Both have 92 points.
They play against each other, and one of them would get 93 wins. Boston is eliminated!

The teams have the following number of wins: New York: 92. Baltimore: 91, Toronto: 91, Boston: 90
There are 5 games remaining (all pairs except New York and Boston.)

Question: Can Boston win the season? Namely, can Boston finish the season with as many point as anybody else? Boston can get at most 92 wins.

All three other teams gets $X = 92 + 91 + 91 + (5 - 2)$ points together.
One of other team # wins $\geq \lceil X/3 \rceil = 93$

Boston is eliminated!

Input:

- $S$ - a set of teams.
- $w_x$ - # of wins of $x \in S$
- $g_{xy}$ - # games remaining between $x$ and $y$
- $z$ - the team to check if it is eliminated or not.

Problem 5.1 Is the team $z$ eliminated?
Complement: Is there away such $z$ would get as many wins as anybody else?

Idea...

- Assume $z$ wins all its remaining games.
- $m$ - wins overall for $z$.
- Build a network flow - check if any other team must have more than $m$ wins.
- $s$ - source of wins.
• For every remaining game, a flow of one would go from $s$ to one of the teams playing it.

• Every team can have at most $m - w_x$ flow from it to the target.
  If max flow smaller than $\sum_{x,y \neq z} g_{xy}$ then $z$ is eliminated.

Construction

• $G$ - flow network.
• $S' = S \setminus \{z\}$.
• $\beta = \sum_{x,y \in S'} g_{xy}$.
• Add $v_x$ a node for team $x$
• Add nodes $s$ and $t$
• $u_{xy}$ for each pair $x, y \in S'$ s.t. $g_{xy} > 0$.
• Add edge $(s, u_{xy})$ with capacity $g_{xy}$.
• Add edges $(u_{xy}, v_x)$ and $(u_{xy}, v_y)$ with infinite capacity.
• Add edge $(v_x, t)$ with capacity $m - w_x$.

• If there is a flow of value $\beta$ in $G$ then, there is a way s.t. all teams get at most $m$ wins.
• Similarly, if $\exists$ scenario s.t. $z$ ties or gets first place...
• Then $\exists$ flow of value $\beta$.

Theorem 5.2 Team $z$ has been eliminated iff the maximum flow in $G$ has value strictly smaller than $\beta$. Thus, we can test in polynomial time if $z$ has been eliminated.

Theorem 5.3 Suppose that team $z$ has been eliminated. Then there exists a “proof” of this fact of the following form:

• $z$ can finish with at most $m$ wins.
• There is a set of teams $T \subset S$ so that
  $$\sum_{s \in T} w_x + \sum_{x, y \in T} g_{xy} > m \cdot |T|.$$

(And hence one of the teams in $T$ must end with strictly more than $m$ wins.)

proof
• If \( z \) eliminated, then max flow in \( G \) \( \alpha < \beta \).

• \( \exists \) min-cut \((A, B)\) in \( G \) of capacity \( \alpha \)  
  (Min-cut max-flow theorem)

\[
T = \left\{ x \mid v_x \in A \right\}
\]

• Consider \( u_{xy} \) in \( A \) and \( x \) or \( y \) not in \( T \) 
  Then \((u_{xy}, v_x)\) or \((u_{xy}, v_y)\) in cut \((A, B)\).  
  \( \Rightarrow \) cut has infinite capacity. Contradiction.

• If \( x \) or \( y \) not in \( A \) then \( u_{xy} \in B \)

• If \( x, y \in T \) and \( u_{xy} \in B \).  
  Move \( u_{xy} \) to \( A \). New cut \((A', B')\)

\[
c(A', B') = c(A, B) - c((s, u_{xy}))
\]

Contradiction!

**Proof continued...**

• Proved: \( u_{xy} \in A \) if and only if \( x, y \in T \).

• Edges in cut \((A, B)\) are:
  
  - \((v_x, t)\), for \( x \in T \)
  
  - \((s, u_{xy})\) where at least one of \( x \) or \( y \) not in \( T \).

• \( c(A, B) = \sum_{x \in T}(m - w_x) + \sum_{\{x,y\} \in T} g_{xy} = m \cdot |T| - \sum_{x \in T} w_x + \left( \beta - \sum_{x,y \in T} g_{xy} \right) \).

• Since \( c(A, B) = \alpha < \beta \): \( m \cdot |T| - \sum_{x \in T} w_x - \sum_{x,y \in T} g_{xy} < 0 \).

• Namely, \( \sum_{x \in T} w_x + \sum_{x,y \in T} g_{xy} > m \cdot |T| \).