CSE 421

Edge Disjoint Path / Image Segmentation / Project Selection

Shayan Oveis Gharan
Marriage Theorem

**Pf.** \( \exists S \subseteq X \text{ s.t., } |N(S)| < |S| \iff G \text{ does not a perfect matching} \)

Formulate as a max-flow and let \((A, B)\) be the min s-t cut

If \(G\) has no perfect matching \(\Rightarrow v(f^*) < |X|\). So, \(\text{cap}(A, B) < |X|\)

Define \(X_A = X \cap A, X_B = X \cap B, Y_A = Y \cap A\)

Then, \(\text{cap}(A, B) = |X_B| + |Y_A|\)

Since min-cut does not use \(\infty\) edges, \(N(X_A) \subseteq Y_A\)

\(|N(X_A)| \leq |Y_A| = \text{cap}(A, B) - |X_B| = \text{cap}(A, B) - |X| + |X_A| < |X_A|\)
Bipartite Matching Running Time

Which max flow algorithm to use for bipartite matching?

Generic augmenting path: \( O(m \text{ val}(f^*) ) = O(mn) \).
Capacity scaling: \( O(m^2 \log C ) = O(m^2) \).
Shortest augmenting path: \( O(m n^{1/2}) \).

Non-bipartite matching.

Structure of non-bipartite graphs is more complicated, but well-understood. [Tutte-Berge, Edmonds-Galai]

Blossom algorithm: \( O(n^4) \). [Edmonds 1965]
Best known: \( O(m n^{1/2}) \). [Micali-Vazirani 1980]
Edge Disjoint Paths
Edge Disjoint Paths Problem

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

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

**Ex:** communication networks.
Max Flow Formulation

Assign a unit capacitory to every edge. Find Max flow from s to t.

Thm. Max number edge-disjoint s-t paths equals max flow value.
Pf. ≤
Suppose there are k edge-disjoint paths $P_1, ..., P_k$.
Set $f(e) = 1$ if e participates in some path $P_i$; else set $f(e) = 0$.
Since paths are edge-disjoint, f is a flow of value k. □
Max Flow Formulation

**Thm.** Max number edge-disjoint s-t paths equals max flow value.

**Pf.** ≥ Suppose max flow value is \( k \)

Integrality theorem \( \Rightarrow \) there exists 0-1 flow \( f \) of value \( k \).

Consider edge \((s, u)\) with \( f(s, u) = 1 \).

- by *conservation*, there exists an edge \((u, v)\) with \( f(u, v) = 1 \)
- continue until reach \( t \), always choosing a new edge

This produces \( k \) (not necessarily simple) edge-disjoint paths.

We can return to \( u \) so we can have cycles. But we can eliminate cycles if desired
Network Connectivity
Network Connectivity

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

**Def.** A set of edges $F \subseteq E$ disconnects $t$ from $s$ if all $s$-t paths uses at least one edge in $F$.

**Ex:** In testing network reliability

![Diagram of a network graph with nodes $s$, $t$, $2$, $3$, $4$, $5$, $6$, and $7$. Edges between nodes are shown, with $s$ pointing to $3$, $3$ pointing to $2$, $3$ pointing to $4$, $4$ pointing to $7$, $7$ pointing to $6$, $6$ pointing to $5$, and $5$ pointing to $t$. The edge between $2$ and $5$ is highlighted in red, indicating a critical path.]
Network Connectivity using Min Cut

Thm. [Menger 1927] The max number of edge-disjoint s-t paths is equal to the min number of edges whose removal disconnects t from s.

Pf.

i) We show that max number edge disjoint s-t paths = max flow.

ii) Max-flow Min-cut theorem => min s-t cut = max-flow

iii) For a s-t cut (A,B), cap(A,B) is equal to the number of edges out of A. In other words, every s-t cut (A,B) corresponds to cap(A,B) edges whose removal disconnects s from t.

So, max number of edge disjoint s-t paths = min number of edges to disconnect s from t.
Image Segmentation
Image Segmentation

Given an image we want to separate foreground from background

- Central problem in image processing.
- Divide image into coherent regions.
Foreground / background segmentation

Label each pixel as foreground/background.

- $V =$ set of pixels, $E =$ pairs of neighboring pixels.
- $a_i \geq 0$ is likelihood pixel $i$ in foreground.
- $b_i \geq 0$ is likelihood pixel $i$ in background.
- $p_{i,j} \geq 0$ is separation penalty for labeling one of $i$ and $j$ as foreground, and the other as background.

**Goals.**

**Accuracy:** if $a_i > b_i$ 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:

$\sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E} p_{i,j}$
Difficulties:
• Maximization (as opposed to minimization)
• No source or sink
• Undirected graph

Step 1: Turn into Minimization

Maximizing
\[ \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E \atop i \in A, j \in B} p_{i,j} \]

Equivalent to minimizing
\[ + \sum_{i \in V} a_i + \sum_{j \in V} b_j - \sum_{i \in A} a_i - \sum_{j \in B} b_j + \sum_{(i,j) \in E \atop i \in A, j \in B} p_{i,j} \]

Equivalent to minimizing
\[ + \sum_{j \in B} a_j + \sum_{i \in A} b_i + \sum_{(i,j) \in E \atop i \in A, j \in B} p_{i,j} \]
Min cut Formulation (cont’d)

$G' = (V', E')$.
Add $s$ to correspond to foreground;
Add $t$ to correspond to background
Use two anti-parallel edges
instead of undirected edge.
Consider min cut \((A, B)\) in \(G'\). (\(A = \) foreground.)

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

Precisely the quantity we want to minimize.

Min cut Formulation (cont’d)
Project Selection
Projects with prerequisites.

- Set $P$ of possible projects. Project $v$ has associated revenue $p_v$.
  - some projects generate money: create interactive e-commerce interface, redesign web page
  - others cost money: upgrade computers, get site license
- Set of prerequisites $E$. If $(v, w) \in E$, can't do project $v$ and unless also do project $w$.
- A subset of projects $A \subseteq P$ is **feasible** if the prerequisite of every project in $A$ also belongs to $A$.

Project selection. Choose a feasible subset of projects to maximize revenue.
Prerequisite graph.

- Include an edge from \( v \) to \( w \) if can't do \( v \) without also doing \( w \).
- \( \{v, w, x\} \) is feasible subset of projects.
- \( \{v, x\} \) is infeasible subset of projects.
Min cut formulation.

- Assign capacity $\infty$ to all 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 $p_s = p_t = 0$. 

![Diagram of the project selection model with nodes and edges labeled with project values and capacities.](image-url)
Claim. \((A, B)\) is min cut iff \(A - \{s\}\) is optimal set of projects.

- Infinite capacity edges ensure \(A - \{s\}\) is feasible.
- Max revenue because:

\[
cap(A, B) = \sum_{v \in B: p_v > 0} p_v + \sum_{v \in A: p_v < 0} (-p_v)
\]

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

constant