Lecture 21:
More Applications of Network Flow

Nov 30, 2018
Image Segmentation
Image Segmentation

- Separate image into foreground and background
- We have some idea of:
  - whether pixel i is in the foreground or background
  - whether pair (i,j) are likely to go together
Image Segmentation

• Input:
  • a directed graph $G = (V, E)$; $V =$ “pixels”, $E =$ “pairs”
  • likelihoods $a_i, b_i \geq 0$ for every $i \in V$
  • separation penalty $p_{ij} \geq 0$ for every $(i, j) \in E$

• Output:
  • a partition of $V$ into $(A, B)$ that maximizes

$$q(A, B) = \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i, j) \in E} p_{ij}$$
Reduction to MinCut

- Differences between SEG and MINCUT:
  - SEG asks us to maximize, MINCUT asks us to minimize

\[
\max_{A,B} \sum_{i \in A} a_i + \sum_{j \in B} b_j - \sum_{(i,j) \in E} p_{ij} \quad \leftrightarrow \quad \min_{A,B} \sum_{i \in A} b_i + \sum_{j \in B} a_j + \sum_{(i,j) \in E} p_{ij}
\]

- SEG allows any partition, MINCUT requires \( s \in A, t \in B \)
Reduction to MinCut

- How should the reduction work?
  - capacity of the cut should correspond to the function we’re trying to minimize

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

Choose a flow network and capacities s.t.

\[
\min_{A,B} \sum_{(i,j) \in E} c(i,j)
\]
Reduction to MinCut

• How should the reduction work?
  • capacity of the cut should correspond to the function we’re trying to minimize

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

Assume \( G \) only has nodes \( i,j \neq 3 \).
Reduction to MinCut

• How should the reduction work?
  • capacity of the cut should correspond to the function we’re trying to minimize

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

from A to B
Step 1: Transform the Input

Input $G, \{a, b, p\}$ for SEG

Input $G'$ for MINCUT

$\text{time: } O(m+n)$
Step 2: Receive the Output

Input $G'$ for MINCUT

Output $(A, B)$ for MINCUT

Solve

time: time to solve min cut in a graph with $n+2$ nodes and $2m+n$ edges

$O(mn)$
Step 3: Transform the Output

Output \((A,B)\) for SEG 

Output \((A,B)\) for MINCUT

\[
\text{time: } O(n)
\]
Reduction to MinCut

• correctness?

\[
\max_{A,B} q(A, B) = \min_{A,B} -q(A, B) = \min_{A,B} \text{cap}_G (A \cup \delta^3, B \cup \delta^3)
\]

• running time?

\[
O(m+n) + O(mn) + O(n) = O(mn)
\]
Densest Subgraph
Image Segmentation

- Want to identify communities in a network
  - “Community”: a set of nodes that have a lot of connections inside and few outside
Densest Subgraph

• Input:
  • an undirected graph \( G = (V, E) \)

• Output:
  • a subset of nodes \( A \subseteq V \) that maximizes \( \frac{2|E(A,A)|}{|A|^2} \)

\[
E(A,B) = \sum_{(i,j) \in E \atop i \in A, j \in B} 1
\]
Reduction to MinCut

- Different objectives
  - find \((A, B)\) to maximize \(\frac{2|E(A,A)|}{|A|}\)
  - find \((A, B)\) to minimize \(|E(A, B)|\)

Suppose \(\frac{2|E(A,A)|}{|A|} \geq \delta\) and see what that implies

\[
\iff 2|E(A,A)| \geq \delta|A|
\]

\[
\iff \sum_{v \in A} \text{deg}(v) - |E(A, B)| \geq \delta|A|
\]

\[
\iff \sum_{v \in V} \text{deg}(v) - \sum_{v \in B} \text{deg}(v) - |E(A, B)| \geq \delta|A|
\]

\[
\iff 2|E| - \sum_{v \in B} \text{deg}(v) - |E(A, B)| \geq \delta|A|
\]

\[
\iff \sum_{v \in B} \text{deg}(v) + \delta|A| + |E(A, B)| \leq 2|E|
\]

\[
\iff \sum_{v \in B} \text{deg}(v) + \sum_{v \in A} \delta + \sum_{(u,v) \in E \text{ for } A \to B} 1 \leq 2|E|
\]
Reduction to MinCut

\[ \sum_{v \in B} \deg(v) + \delta |A| + |E(A, B)| \leq 2|E| \]

\( (\text{mincut in } G) \leq 2|E| \) if and only if \( \exists A \) \( \frac{2|E(A, B)|}{|A|} \geq \delta \)
Reduction to MinCut

\[\sum_{v \in B} \deg(v) + 8|A| + |E(A,B)| \leq 2|E|\]
Edge-Disjoint Paths
(Edge) Disjoint Paths

• **Input:** directed graph $G = (V, E, s, t)$
• **Output:** a largest set of edge-disjoint s-t paths
  • A set of s-t paths $P_1, ..., P_k$ is edge disjoint if the paths do not share any edges

A large set of disjoint paths means we can tolerate edge failures.
Bipartite Matching

• There is a reduction that uses integer maximum s-t flow to solve edge disjoint paths.
Step 1: Transform the Input

Input $G$ for EDP

Input $G'$ for MAXFLOW
Step 2: Receive the Output

Input $G'$ for MAXFLOW

Output $f$ for MAXFLOW

Red arrow means $f(e) = 1$
Black means $f(e) = 0$

Red arrow means $f(e) = 1$
Black means $f(e) = 0$
Step 3: Transform the Output

Output M for MCBM

Output f for MAXFLOW
Correctness

- **Easy Direction:** If there are $k$ edge disjoint paths then there is a flow of value $k$
Correctness

• **Harder Direction:** If there is a flow of value k, then there are k edge disjoint paths
Running Time

• Need to analyze the time for:
  • (1) Producing $G'$ given $G$
  • (2) Finding a maximum flow in $G'$
  • (3) Producing $M$ given $G'$
Summary

Solving maximum s-t flow in a graph with n nodes and m edges and $c(e)=1$ in time $T$

Solving edge disjoint paths in a graph with n nodes and m edges in time $T + O(mn)$

• Can solve edge disjoint paths in time $O(nm)$ using Ford-Fulkerson
(Node) Disjoint Paths

- **Input:** directed graph $G = (V, E, s, t)$
- **Output:** a largest set of node-disjoint s-t paths
  - A set of s-t paths $P_1, ..., P_k$ is node-disjoint if the paths do not share any nodes

A large set of disjoint paths means we can tolerate edge failures.
Step 1: Transform the Input

Input $G$ for NDP

Input $G'$ for EDP