Northwestern University
EECS 336
Fall 2015
Final Lab: Multiway Cut

1 Min Cut with Undirected Edges

Input:

- Undirected graph $G = (V, E)$.
- Terminals s and t.
- Edge weight: w_e for all e.

Output: A set of edges A such that:

- s and t are in different components in $(V, E \setminus A)$.
- $\sum_{e \in A} w_e$ is minimized.

Idea:

- Add forward and back edges.
- Run max flow.
- Use Max Flow = Min Cut

Algorithm:

- For each undirected edge $e = \{u, v\}$, replace e with (u, v) and (v, u), each with weight w_e. [Call this graph G'.]
- Return the min $s - t$ cut in this graph.

Correctness:

- Max flow never uses both a forward edge and a back edge.

  [Proof: remove flow from both edges. Still satisfies constraints, doesn't change value of flow.]
- $\Rightarrow$ min $s - t$ cut in G' has value equal to $\sum_{e \text{ from } A \text{ to } B} w_e$.
- $\Rightarrow$ min $s - t$ cut is also min undirected cut.

2 Multiway Cut

Input:

- Undirected graph $G = (V, E)$.
- Weights w_e.
- Terminals $s_1, \ldots, s_k$.

Output: A collection of edges A such that:

- $s_1, \ldots, s_k$ are mutually disconnected in $(V, E \setminus A)$.
- $\sum_{e \in A} w_e$ is minimized.

Idea:

- Disconnect each of $s_1, \ldots, s_k$ separately, and optimally.
- Remove all of these edges.

2.1 Disconnecting a Single Terminal

Idea: Treat other terminals as a single vertex.

Algorithm: To disconnect s_1,

- Remove $s_2, \ldots, s_k$ from V.
- Replace them with s^*.
- For every edge e between $s_2, \ldots, s_k$ and any other vertex v, add an between v and s^* with the same weight.

  [Note: we have a multigraph now! Call this new graph G'.]
- Find min $s_1 - s^*$ cut in this graph using our undirected min cut algorithm.
- Return the set of edges (or constructed surrogates) returned by this algorithm.

Correctness:

- Consider a cut in G'. We can find a cut in G with the exact same weight by simply cutting the same edges. This disconnects s and all of the other terminals by the fact that it disconnects s and s^*. [There's a simple contradiction argument.]
- Consider a minimum $s_1 - \{s_2 \ldots s_k\}$ cut in G.
  - Note that it won't ever cut an edge between two of $s_2, \ldots, s_k$ - not doing so would also yield a separating cut, but would have smaller cost.
  - Hence for any minimum $s_1 - \{s_2 \ldots s_k\}$ cut in G, all of the edges cut are also present in G'.
  - Cut them. By simple contradiction, this disconnects s_1 and s^*.

2.2 A 2-Approximation for Multiway Cut

[Recall] Idea:

- Disconnect each of $s_1, \ldots, s_k$ separately, and optimally.
- Remove all of these edges.

Algorithm:

- For each s_i, compute C_i, the optimal s_i, $\{s_{-i}\}$ cut.
- Return $C = \bigcup_i C_i$.

Correctness:

- **Step 1: Rewrite [Lowerbound] OPT**
  - Let A^* be the optimal multiway cut.
  - Each terminal s_i is in its own component, V_i
  - For each i, the cut $(V_i, V \setminus V_i)$ is an isolating cut for s_i.
  - Let A_i be the edges cut by $(V_i, V \setminus V_i)$.
  - Note: every edge e in A connects two components, V_i and V_j.
  - $\Rightarrow e \in A_i$ and $e \in A_j$.

    Consequences:
    * $A = \bigcup_i A_i$

    * $\sum_i w(A_i) = 2w(A)$.

- **Step 2: Upperbound ALG**
  - Recall: C_i is optimal isolating cut for s_i.
  - A_i is also an isolating cut for s_i.
  - $\Rightarrow w(A_i) \geq w(C_i)$
  - $\Rightarrow \sum_i w(C_i) \leq \sum_i w(A_i) = 2w(A)$
- $\Rightarrow$ ALG is a 2-approximation!

Runtime: k calls to max flow on a graph the same size as G.