COL751 - Lecture 9

Let $G = (V, E)$ be a flow graph with unit capacities, with a source vertex s and a sink vertex t. Recall that any (s,t)-cut is a partition (A, B) such that $s \in A$ and $t \in B$. Note that on removal of the edges in cut, i.e. the set $E \cap (A \times B)$, there is no s to t path in G.

Equivalently, an (s,t)-cut is a set $\mathcal{E}$ of edges such that there is no s to t path in $G - \mathcal{E}$. These two definitions of (s,t)-cut are equivalent as in this case we can define $A_{\mathcal{E}}$ as vertices reachable from s in $G - \mathcal{E}$, and $B_{\mathcal{E}}$ as $V \setminus A_{\mathcal{E}}$. The size of an (s,t)-cut $\mathcal{E} \subseteq E$ is equal to number of edge in set $\mathcal{E}$.

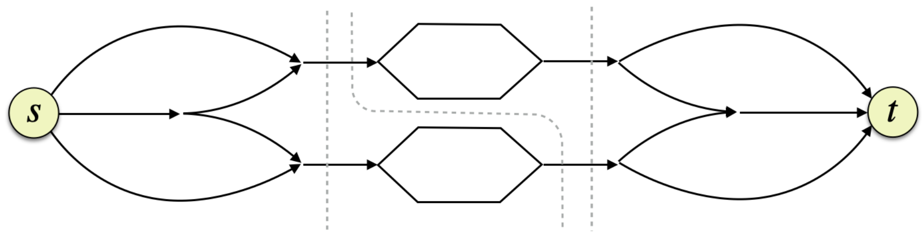

Figure 1 – Depiction of a graph with (s,t)-max-flow value as two.

Consider the graph above. There are four possible (s,t)-min-cuts of size 2. In general, there can be exponentially many (s,t)-min-cuts, and enumerating/storing them is not feasible. So, we ask the following related question:

> *Can we compute a compact data-structure that given any query set $\mathcal{E} \subseteq E$, answers if $\mathcal{E}$ is an (s,t)-min-cut in G in $O\big(\ poly(|\mathcal{E}|)\ \big)$ time?*

We will affirmatively answer this using the structural properties of (s,t)-min-cuts.

1 Structural properties of (s,t) Min-Cuts

Pre-processing phase We first remove from G all vertices $v \in V$ that do not lie on any simple (s,t)-path in G.

Let f be an (s,t)-max-flow in an unweighted undirected graph G, and G_f be corresponding residual graph. We denote by G_f^{scc} the graph obtained by merging SCCs of G_f into super-nodes or clusters, and let $\overline{G_f^{scc}}$ be the reverse graph.

Definition 1 *We say an edge (x,y) in G is:*

- *inter-cluster if $SCC(x) \neq SCC(y)$.*
- *intra-cluster if $SCC(x) = SCC(y)$.*
- *cut-edge if deletion of (x,y) from G reduces (s,t)-max-flow by one.*

Definition 2 *We say a set of edges $\mathcal{E}$ in a DAG is a* **chain** *if there is a simple path containing $\mathcal{E}$. A set $\mathcal{E}$ is an* **anti-chain** *if no subset of $\mathcal{E}$ of size two or more is a chain.*

Property 3 (Corollary of Max-Flow Min-Cut Theorem) *Any (s,t)-cut (A,B) is a minimum-cut if and only if all edges across (A,B) in a residual graph G_f (with respect to a max-flow f) are directed from B to A.*

Proof: Homework.

(Hint: Use Lemma 3 and Theorem 7 from Lecture 8). $\square$

We now prove some properties of cuts in G.

Lemma 4 *Each cut edge in G is an inter-cluster edge.*

Proof: We will prove that if $e = (x,y) \in E$ is intra-cluster, then e is not cut-edge.

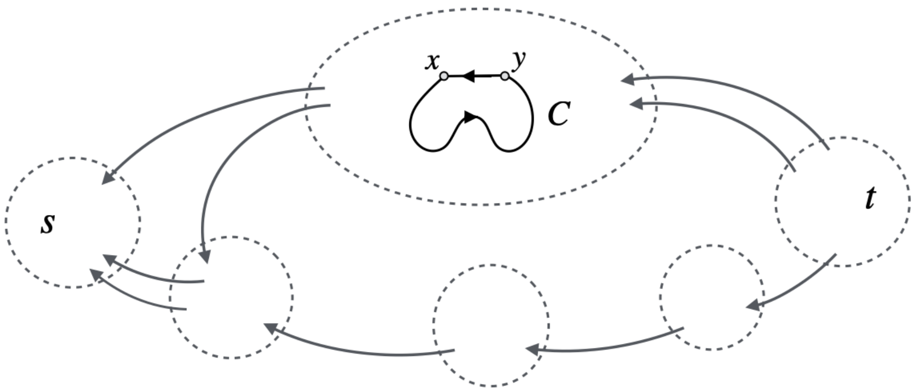

Figure 2 – Depiction of SCCs in graph G_f along with cycle C containing edge (y,x).

If $f(e) = 0$, then proof is immediate. Suppose $f(x,y) \neq 0$, then graph G_f will contain back edge (y,x). Consider a cycle C in G_f containing (y,x). We just pass one unit flow in this cycle. This cancels flow through e as flow is re-routed to path $P = C \setminus e$ in G_f. With respect to this new max-flow, say f', we have $f'(e) = 0$. This proves that e is not a cut edge as on its deletion the maximum flow is unaffected. $\square$

Lemma 5 *For any (s,t)-min-cut (A,B), and any cluster W in G_f, either $W \subseteq A$ or $W \subseteq B$.*

Proof: Note that all edges within W are intra-cluster and thus by Lemma 4 cannot be cut edges. Therefore, W cannot be subdivided by cut (A,B) as otherwise it will contain a cut edge.

Alternative proof: By Property 3, for any (s,t)-min-cut (A,B) all edges across (A,B) in G_f are directed from B to A. So it cannot be the case that both $W \cap A$ and $W \cap B$ are non-empty as otherwise G_f will contain an edge from B to A which is not possible. $\square$

Remark An immediate corollary of Lemma 5 is that on merging SCCs into super-nodes *none* of the (s,t)-min-cuts of G is destroyed.

Lemma 6 *For any (s,t)-min-cut (A,B), edges going from B to A in G_f corresponds to an anti-chain in G_f^{scc}.*

Proof: Consider an (s,t)-min-cut (A,B), and let $\mathcal{E}$ be set of edges in G_f going from B to A. By Property 3 there is no path in graph G_f going from A to B, so no two edges in $\mathcal{E}$ can lie on same path. This proves that $\mathcal{E}$ is an anti-chain. □

Lemma 7 (Reverse of Lemma 6) *Any maximal anti-chain of inter-cluster edges, say $\mathcal{E}$, corresponds to an (s,t)-min-cut.*

Proof: Let $e_1 = (y_1, x_1), \ldots, e_k = (y_k, x_k)$ be edges in $\mathcal{E}$. Let

$$\begin{aligned} A &= \cup_{i=1}^{k} Reach(x_i, G_f), \\ B &= V \setminus A. \end{aligned}$$

1. As $\mathcal{E}$ is an anti-chain, $y_1, \ldots, y_k \notin A$.
2. There is no edge in G_f in set $(B \times A) \setminus \mathcal{E}$ because the set $\mathcal{E}$ is a maximal anti-chain.
3. By definition of set A, there is no edge in G_f lying in set $A \times B$ (why?). Thus (A,B) is a cut. Moreover, it is an (s,t)-min-cut as by Property 3 any cut (A,B) is minimum cut if all edges across cut (A,B) are directed from B to A in G_f.

This also proves that size of each maximal anti-chain is same as size of (s,t)-min-cut. □

Lemma 8 (Reverse of Lemma 4) *Each inter-cluster edge in G_f corresponds to a cut-edge in G.*

Proof: Let $e = (y_1, x_1)$ be an inter-cluster edge in G_f, and $\mathcal{E}$ be any maximal anti-chain containing e. By Lemma 7, edges in $\mathcal{E}$ corresponds to an (s,t)-min-cut. This proves that e is a cut edge. □

As a corollary of Lemma 4-8, we obtain the following theorem.

Theorem 9 *Let G be an unweighted undirected graph satisfying that each $v \in V$ lies on a simple (s,t)-path in G, and let f be an (s,t)-max-flow in G. Then there is $1-1$ correspondence between:*

1. *Inter-cluster an cut edges.*
2. *(s,t)-min-cuts and maximal anti-chains in G_f^{scc}.*

2 Data-Structure for verifying (s,t)-min-cuts

Let G be an unweighted undirected graph with (s,t)-max-flow λ. We consider the problem of designing an oracle that answers if a query set $\mathcal{E}$ of edges is an (s,t)-min-cut quickly. Henceforth, for any vertex $x \in V$ we use notation $\boldsymbol{x}$ to denote cluster of x in graph G_f^{scc}.

Consider a family $\mathcal{P} = (P_1, \dots, P_\lambda)$ of λ edge-disjoint paths from $\boldsymbol{s}$ to $\boldsymbol{t}$ in $\overline{G_f^{scc}}$. For each cluster $\boldsymbol{x}$ and each path $P \in \mathcal{P}$, let $\textsc{first}(\boldsymbol{x}, P)$ denote the first SCC in path P that is reachable from $\boldsymbol{x}$ in $\overline{G_f^{scc}}$.

Lemma 10 *Consider two inter-cluster edges $e_1 = (\boldsymbol{x}_1, \boldsymbol{y}_1)$ and $e_2 = (\boldsymbol{x}_2, \boldsymbol{y}_2)$ in $\overline{G_f^{scc}}$, and let $P \in \mathcal{P}$ be the path containing e_2. Then $e_1 \leqslant e_2$ (i.e. there is a path in which e_1 precedes e_2) if and only if $\textsc{first}(\boldsymbol{y}_1, P)$ is either identical to or a predecessor of $\boldsymbol{x}_2$ in P.*

In our data-structure we store:

1. A hash function H storing all cut-edges. This takes $O(n\lambda)$ space, and given an edge e it answers whether or not e is a cut-edge in constant time.
2. A function F that maps a vertex $x \in V$ to its cluster $\boldsymbol{x}$ in $\overline{G_f^{scc}}$.
3. The node $\textsc{first}(\boldsymbol{x}, P)$, for each cluster $\boldsymbol{x}$ and each path $P \in \mathcal{P}$. This again takes $O(n\lambda)$ space.
4. Mapping M from cut edges to paths in $\mathcal{P}$ that contain them. So given any edge $(\boldsymbol{x}, \boldsymbol{y})$ we can retrieve index i that satisfies $(\boldsymbol{x}, \boldsymbol{y}) \in P_i$ in constant time.
5. Topological ordering of vertices in $\overline{G_f^{scc}}$.

Next we present the query oracle.

for $i = 1$ *to* λ **do**
  if (x_i, y_i) *is not in hash-table* H **then** return false;
end
foreach $i \neq j \in [1, \lambda]$ **do**
  Use mapping M to determine path P containing edge $(\boldsymbol{x}_j, \boldsymbol{y}_j)$;
  Next compute $\textsc{first}(\boldsymbol{y}_i, P)$;
  if (*Topological ordering of* $\textsc{first}(\boldsymbol{y}_i, P)$ *is less than equal to that of* $\boldsymbol{x}_j$) **then**
    return false;
  end
end
Return true;

Algorithm 1: Query oracle to verify if set $\{(x_i, y_i)\}_{i=1}^{\lambda}$ is an (s,t)-min-cut.

The size of the data structure is $O(n\lambda)$. Also it is easy to verify that running time of algorithm 1 is $O(|\mathcal{E}|^2)$. Therefore, we have the following result.

Theorem 11 *For any n vertex undirected graph G with (s,t)-max-flow λ, we can compute in polynomial time an $O(n\lambda)$ sized data-structure that given any query set $\mathcal{E}$ of λ edges, reports whether or not it is an (s,t)-min-cut, in $O(|\mathcal{E}|^2)$ time.*

3 Certificate for k-edge connectivity

An undirected graph $G = (V, E)$ is said to be *k-edge connected* if for each pair $(x, y) \in V \times V$ of distinct vertices, there are k-edge disjoint paths between x and y in G.

Problem Let G be a k-edge-connected graph with n vertices and m edges. Our goal is to compute in $O(mk)$ time a sparse subgraph H of G with $O(nk)$ edges such that H is also k-edge-connected.

Algorithm Our algorithm runs in k rounds. In the i^{th} round we compute a spanning forest T_i of graph $G - \big(E(T_1) \cup \cdots \cup E(T_{i-1})\big)$. Finally, we set $H = (V, E_H)$ where $E_H = \cup_{i \leqslant k} E(T_i)$ is the union of the edges of k forests.

Lemma 12 *The subgraph H is a certificate for k-edge connectivity.*

Proof: We need to prove that H is k-edge-connected. Observe that due to Max-Flow Min-Cut theorem the number of edges in G across any cut (A, B) is at least k. In order to prove our claim it suffices to argue that the number of edges across any cut (A, B) in graph H as well is at least k.

For $i = 1$ to k, let $H_i = (V, E_i)$ be graph obtained by taking union of the edges of first i forests, $T_1, \ldots, T_i$. Consider a cut (A, B). We will use induction to argue $|E_i \cap (A \times B)| \geqslant i$. The base condition trivially holds. Consider an index $i \in [2, k]$.

- Case 1: $|E_{i-1} \cap (A \times B)| \geqslant i$:
  In this case we trivially have $|E_i \cap (A \times B)| \geqslant i$ as $E_{i-1} \subseteq E_i$.

- Case 2: $|E_{i-1} \cap (A \times B)| = i - 1$:
  In this case we have sets A and B are connected by at least one edge in graph $G - \big(E(T_1) \cup \cdots \cup E(T_{i-1})\big)$. This edge must be included in spanning forest T_i. So, we have $|E_i \cap (A \times B)| \geqslant i$.

This proves that the number of edges across any cut (A, B) in graph H is at least k. Therefore by Max-Flow Min-Cut theorem, for any pair $(x, y) \in V \times V$, there are k-edge disjoint paths between x and y in H. $\square$