1 Prelims

1.1 Multiway Cut (Edge version)

Given graph G and terminals $T \subseteq V(G)$, $T = \{t_1, t_2, \ldots, t_l\}$, delete minimum number of edges $E' \subseteq E(G)$ such that $G \setminus E'$ has no t_i, t_j path for all i, j.

For $l = 2$, the problem is in P but is NP-hard even for $l = 3$. The problem is FPT for parameter k, the optimal edge cardinality.

Theorem 26.1. *Multiway Cut has a $O^*(4^k)$ algorithm.*

1.2 Submodularity

A set funtion $f : 2^U \to \mathbb{R}$ is called *submodular* if, for all X, Y

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$$

This corresponds to decreasing marginal returns, for all $X \subseteq Y, z \notin Y$

$$f(X \cup \{z\}) - f(X) \geq f(Y \cup \{z\}) - f(Y)$$

1.3 s-t mincuts

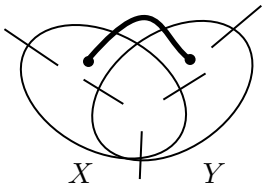

Figure 26.1: Submodularity of cut function

If ∂X denotes the set of edges with exact one end in X, $f(X) := |\partial X|$ is submodular. Indeed, if $\partial X, \partial Y$ are some cuts, then $\partial X \cup Y$ and $\partial X \cap Y$ have the same edges except the bold ones in Figure 26.1.

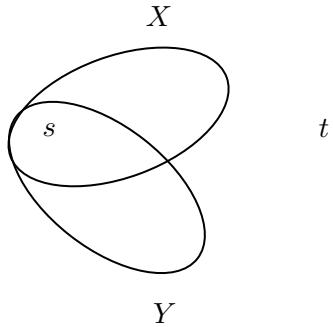

Figure 26.2: $X \cup Y$ and $X \cap Y$ are s-t cuts

Further if $\partial X, \partial Y$ are s-t mincuts then so are $X \cup Y$ and $X \cap Y$. It's easy to see $X \cup Y$ and $X \cap Y$ are both s-t cuts (Figure 26.2). Let $f(X) = f(Y) = \lambda$. By submodularity

$$2\lambda = f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y)$$

But since X, Y are mincuts $f(X \cup Y) \geq \lambda$ and $f(X \cap Y) \geq \lambda$. So the above inequality holds tightly with $f(X \cup Y) = f(X \cap Y) = \lambda$.

Also, there is a nice structure that s-t mincuts satisfy. $\exists R_{\max}, R_{\min}$ such that for any s-t cut C the set of vertices reachable from X (say R_C) satisfy $R_{\min} \subseteq R_C \subseteq R_{\max}$ (Figure 26.3).

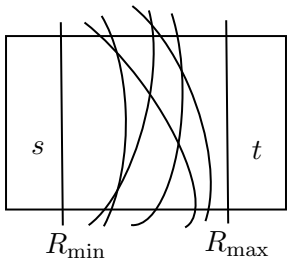

Figure 26.3: Cut structure for s-t mincuts

2 Multiway Cut algorithm

2.1 Important cuts

$S \subseteq E$ be an (X, Y)-cut (set of edges that separate X, Y). S is 'important' if

1. S is inclusion wise minimal i.e. $S - e$ is not a cut for any e.
2. $\nexists$ (X, Y)-cut S' s.t. $|S'| \leq |S|$ and $\text{Reach}(S) \subsetneq \text{Reach}(S')$ ($\text{Reach}(S)$ is the set of vertices reachable from X in $G \setminus S$). Intuitively, S is the 'rightmost' cut of its size (Figure 26.4).

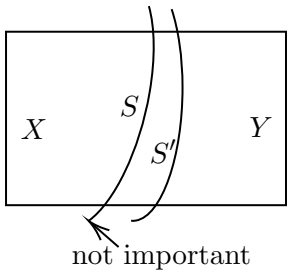

Figure 26.4: Important cuts

We note a couple of facts about important cuts.

Theorem 26.2. *#(important cuts of size $\leq k$) $\leq 4^k$.*

Theorem 26.3. *Let l be the set of important cuts in G. Then $\sum_{c \in l} 4^{-|c|} \leq 1$.*

Note that Theorem 26.3 implies Theorem 26.2. Proof of Theorem 26.3 appears later (Section 2.3), we will first motivate ourselves for this result (and important cuts) and see how to use this to get Theorem 26.1.

2.2 A Branching Algorithm

Lemma 26.4 (Pushing Lemma)**.** $\exists$ *an OPT multiway cut containing edges of some important* $(t_1, T \setminus t_1)$ *cut.*

Proof. Let S^* be minimal subset of OPT that separates $t, T \setminus t$. We claim S^* is not important. If not, we can push S^* to S' an important cut. Note $(\text{OPT} - S^*) \cup S'$ (Figure 26.5) is also multiway cut (t is separated, for other pairs any path uncut by removal S^* will still be cut by S').

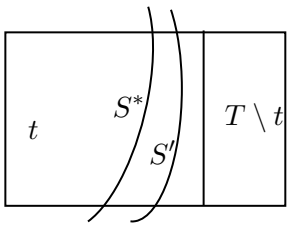

Figure 26.5: Proof of pushing lemma

□

Now the algorithm: We branch on important cuts, and get a recursion tree with branching at most 4^k (Theorem 26.2) and depth at most k. This gives 4^{k^2} leaves and a $O^*(4^{k^2})$ algorithm. We show a tighter analysis using Theorem 26.3 which gives the $O^*(4^k)$ bound.
If S_k is the set of size-k important cuts, by Theorem 26.3 $\sum_{c \in S_k} 4^{-|c|} \leq 1$. By induction,

$$\text{size of tree} \leq \sum_{c \in S_k} 4^{k-|c|} \leq 4^k$$

To branch, we need to efficiently determine how to check if a cut is important.

Lemma 26.5. *Given an* (X, Y)*-cut* S*, can check if* S *is important efficiently.*

Proof. Let R be reachable set for X (Figure 26.6). Then S is important iff $S = \partial R$ and ∂R is the unique minimum (R, Y)-cut. Can be checked in $O(|\partial R|(m+n))$ time.

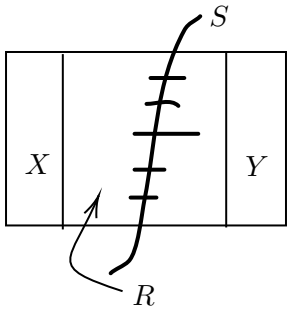

Figure 26.6: Checking if S is important

□

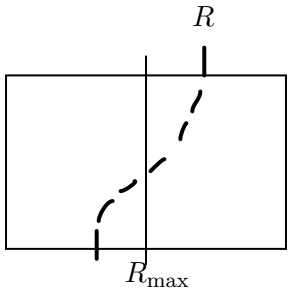

Figure 26.7: $R_{\max} \subseteq R$ for important cut ∂R.

2.3 Proof of Theorem 26.3

Lemma 26.6. *If ∂R is important then $R_{\max} \subseteq R$*

Proof. Suppose not (Figure 26.7).

$$\begin{aligned} |\partial R_{\max}| + |\partial R| &\geq |\partial R_{\max} \cap R| + |\partial R_{\max} \cup R| \\ = \lambda \qquad\qquad &\geq \lambda \\ \implies |\partial R_{\max} \cup R| &\leq |\partial R| \end{aligned}$$

Thus $\partial R_{\max} \cup R$ (which is to the right of ∂R) contradicts that ∂R is important. □

Let's look at edge (u, v) across $R_{\max}$ (Figure 26.8). An important cut C cuts (u, v) or not.

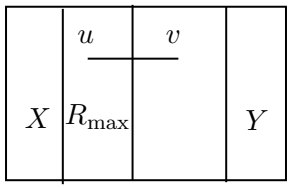

Figure 26.8: Edge across $R_{\max}$

If cuts: Find $k-1$ size important (X, Y) cut in $G - (u, v)$. $(\lambda \to \lambda - 1)$

Else: Contract (u, v) and find $(R_{\max} \cup \{v\}, Y)$ important cut in contracted G. $(\lambda \to \lambda + 1)$

We can now define $2k - \lambda$ as a potential to conclude $2^{2k-\lambda} = 4^k/2^\lambda$ leaf nodes in recursion. A more careful analysis uses an inductive claim $\sum_{c \in l} 4^{-|c|} \leq 2^{-\lambda}$, and uses induction hypothesis to show

If case (S_1): $\sum_{c \in S_1} 4^{-|c|} = \frac{1}{4} \sum_{c' \in S_1} 4^{-|c'|} \leq \frac{1}{4} 2^{-(\lambda-1)} = \frac{1}{2} 2^{-\lambda}$, where c' has (u, v) removed.

Else (S_2): $\sum_{c \in S_2} 4^{-|c|} \leq 2^{-(\lambda+1)} = \frac{1}{2} 2^{-\lambda}$, by IH.

3 Extension to Directed Graphs

A lot extends to directed cuts (definition of important cuts, bound on # cuts), but unfortunately the pushing lemma fails.

3.1 Pushing Lemma fails

Pushing lemma does not hold for directed multiway cuts. Consider the example in Figure 26.9.

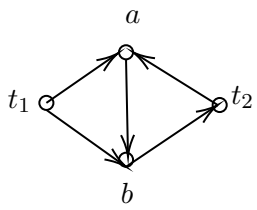

Figure 26.9: Pushing lemma counterexample for directed multiway cut

Here $k = 1$, and (a, b) is the minimum multicut but is not important (and gets pushed to (b, t_1) or (b, t_2)). Where does the lemma fail though? Intuitively, we may allow $T \setminus \{t_1\} \to t_1$ paths while pushing, even though we still cut all $t_1 \to T \setminus \{t_1\}$ paths (Figure 26.10).

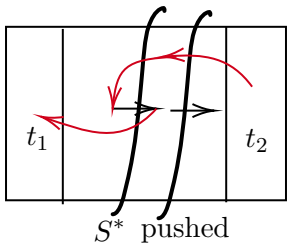

Figure 26.10: We may have a red uncut path while pushing

3.2 All is not lost - Directed Skew Multicut

This captures exactly the situation where pushing lemma works for directed graphs.

In the Directed Skew Multicut problem, we are given $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$. For each j delete edges so that s_j cannot reach $t_j, t_{j+1} \ldots, t_k$ (Figure 26.11).

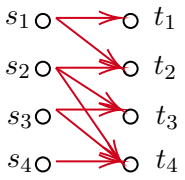

Figure 26.11: Directed Skew Multicut - red edges denote separation

3.3 Application: Directed Feedback Vertex Set

Recall that the Feedback Vertex Set problem considers removal of vertices to make the graph acyclic. Note that in a directed graph the edge (arc) version is equivalent to the node version with same parameter (Figure 26.12). We can use iterative compression - given G and vertex solution of size $k + 1$, want to find edge solution (feedback arc set) of size k.

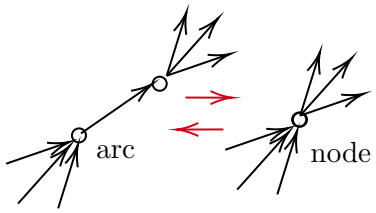

Figure 26.12: Arc and node versions are equivalent for directed graphs

Say $W = \{w_1, \ldots, w_{k+1}\}$. If $E^* \subseteq E$ is the optimal solution, $G \setminus E^*$ has no directed cycles, and therefore has a topological ordering $v_1, \ldots, v_n$. Let's say this ordering restricted to W is $w_1, \ldots, w_{k+1}$. Any directed cycle has to go 'back' $w_j \to w_i$ for $i \leq j$ ($i = j$ corresponds to empty transition). We could guess it to get FPT (at most $(k+1)!$ choices).
We show this is equivalent to the Directed Skew Multicut problem by splitting each w_i into $t_i \to s_i$ (Figure 26.13).

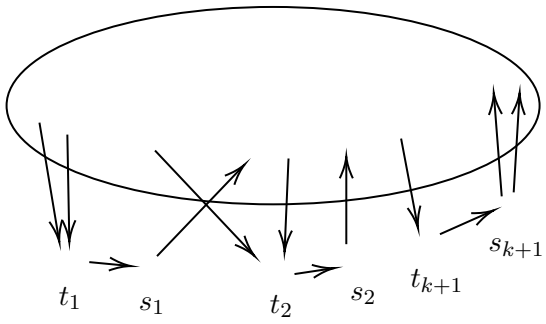

Figure 26.13: Skew multicut with feedback vertex set

Claim 26.7. *$G \setminus E^*$ has acyclic ordering with $w_1, \ldots, w_{k+1}$ $\implies$ E^* hits each $s_j \to t_i$ path for $j \geq i$ $\iff$ E^* is a skew multicut on this graph $\implies$ $G \setminus E^*$ is acyclic.*