Advanced Approximation Algorithms **(CMU 15-854B, Spring 2008)**

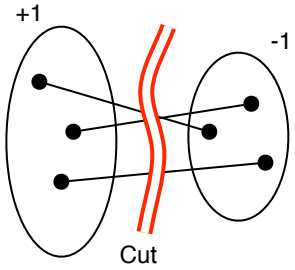
Lecture 18: Multicut

March 20, 2008

Lecturer: Anupam Gupta *Scribe: Kanat Tangwongsan*

1 The Multicut Problem

In this lecture, we will study a different type of graph cuts than the max-cut problem previously discussed. Let us recall that the maximum-cut problem is to find a 2-coloring of the vertices to maximize the total weight of the edges whose endpoints have different colors. We have seen a simple factor-$1/2$ approximation in Homework #1, and an SDP-based factor-0.878 approximation in Lecture 14. We also know that the problem is NP-hard. It is natural to wonder what we can say if the objective is instead to minimize the weight.



Surprisingly, the minimization version turns out to be much easier than max-cut: by a celebrated theorem of Ford and Fulkerson [FF62], the minimum s-t cut problem can be solved efficiently using the duality between max-flow and min-cut. Thus, we can try all possible (s, t) pairs and solve this problem exactly in polynomial time. In fact, we can be a little more careful about which (s, t) pairs to examine and obtain a slightly more efficient algorithm.

Let's look at a generalization of this problem.

Minimum Multicut. The input consists of a weighted, undirected graph $G = (V, E)$ with a non-negative weight c_e for every edge $e \in E$, and a set of terminal pairs $\{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$. A *multicut* is a set of edges whose removal disconnects each of the terminal pairs. Formally, a set $E' \subseteq E$ is a multicut if for all $i = 1, 2, \ldots, k$, there is no path between s_i and t_i in the graph $(V, E \setminus E')$. The cost of a multicut E' is given by $\mathsf{cost}(E') = \sum_{e \in E'} c_e$. Thus, the minimum mulitcut problem (or multicut for short) is to find a multicut E' that minimizes $\mathsf{cost}(E')$.

A special case of $k = 1$ is the min s-t cut problem. As already noted, this problem can be solved efficiently using the duality between max-flow and min-cut.

When $k = 2$, Hu [Hu63] (see also Seymour [Sey79] for a simple proof) showed that the problem interestingly reduces to an single-commodity flow instance, and therefore can be solved in polynomial time. When $k \geq 3$, the problem is known to be NP-hard; however, a factor-k approximation can be easily achieved by taking the union of all (s_i, t_i) cuts.

Hardness of Multicut. Not only is the multicut problem NP-hard but also hard to approximate within 1.3 of the optimal solution (even on trees). If we assume the unique games conjecture (UGC), the problem is $\Omega((\log \log n)^{\varepsilon})$ hard.

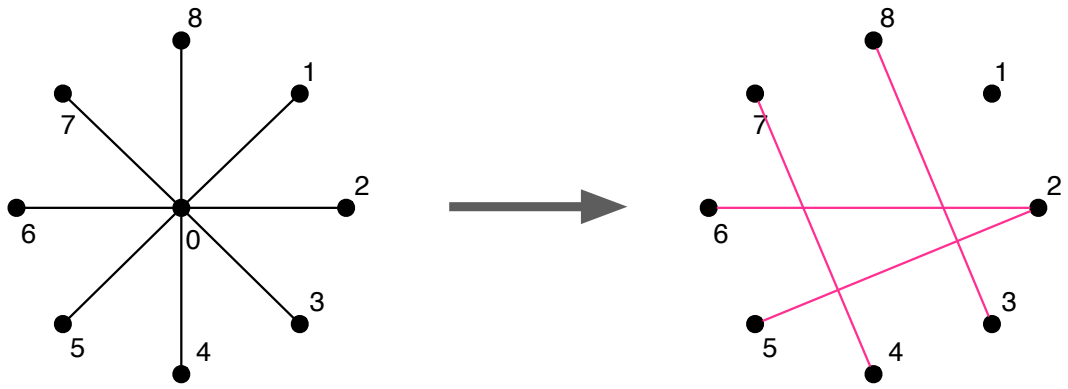


Figure 1: The reduction on S_8 with terminal pairs $\{(2,6),(2,5),(3,8),(4,7)\}$.

We present a reduction from multicut on star graphs to vertex cover. Figure 1 illustrates the reduction. Consider a star graph S_n on $n+1$ nodes. Suppose the vertices are $\{0,1,\ldots,n\}$, where 0 is the center node. Given the terminal pairs $\{(s_1,t_1),\ldots(s_k,t_k)\}$, we will construct a graph G on vertices $\{1,\ldots,n\}$ as follows: join every (s_i,t_i) pair an edge. Note that a multicut in S_n directly corresponds to a vertex cover in G and vice versa, and hence the two problems (vertex cover on general graphs, and multicut on stars) are equivalent. Therefore, the hardness results of vertex cover can be used to show that the multicut problem is hard to approximate to 1.36 (due to Dinur and Safra).

Remark. The unique games conjecture implies that vertex cover is hard to approximate to $2-\varepsilon$ (due to Khot and Regev), and hence the same (conditional) hardness holds for multicut as well.

2 Multicut on Trees

Let us now attempt to develop an approximation algorithm for this problem. We start with trees. As a first attempt, we formulate the following LP relaxation:

$$\begin{aligned} \text{Minimize} \quad & \sum_{e\in E} c_e x_e \\ \text{Subject to} \quad & \sum_{e\in \mathcal{P}_{s_i\to t_i}} x_e \geq 1 \qquad \text{for all } i=1,2,\ldots,k, \end{aligned}$$

where $\mathcal{P}_{s_i\to t_i}$ is the set of edges on the unique path between s_i and t_i. Note that when the constraints $x_e \in \{0,1\}$ are added, the LP becomes an Integer Program (IP), which models precisely the multicut problem; unfortunately, we cannot hope to solve this IP exactly, because the multicut problem is NP-hard even on trees. Hence, we will give a rounding procedure for the LP solution, show that the cost of the solution is at most 2 times the LP value.

Given an optimal solution x of the LP, we define a distance function $d\colon V \times V \to \mathbb{R}_+$ induced by x as follows:

$$d(u,v) := \sum_{e \in \mathcal{P}_{u \to v}} x_e \qquad \text{for all } (u,v) \in V \times V$$

Pick a node $r \in V$ arbitrarily. For the analysis purposes, we will make the tree rooted at r. For $i = 1, \ldots, k$, let lca_i denote the least common ancestor between s_i and t_i. Remember that $d(s_i, t_i) \geq 1$ for all $i = 1, \ldots, k$. This leads to the following observation:

Observation 2.1. *For all $i = 1, \ldots, k$, $\max\{d(s_i, \mathsf{lca}_i), d(t_i, \mathsf{lca}_i)\} \geq 1/2$.*

Definition 2.2. Assuming that $d(r,v) = d(r,u) + x_{uv}$, we say that an edge uv is "cut" by $\alpha \in \mathbb{R}_+ \cup \{0\}$ if and only if $d(r,u) \leq \alpha$ and $d(r,v) > \alpha$.

The Algorithm: We now describe the rounding algorithm. Given an optimal LP solution x, the algorithm performs the following steps:

1. Pick $R \in [0, 1/2]$ uniformly at random.
2. Delete all the edges "cut" by $R, R + \frac{1}{2}, R + 1, R + \frac{3}{2}, \ldots$.

Claim 2.3. *The algorithm returns a feasible solution.*

Proof. We will show that for all $i = 1, 2, \ldots, k$, at least an edge on the path $\mathcal{P}_{s_i \to t_i}$ is "cut." Consider a pair of terminal (s_i, t_i). By Observation 2.1, we know $\max\{d(s_i, \mathsf{lca}_i), d(t_i, \mathsf{lca}_i)\} \geq \frac{1}{2}$, so w.l.o.g, assume $d(s_i, \mathsf{lca}_i) \geq \frac{1}{2}$. Therefore, at least on edge on the unique path between s_i and lca_i is "cut," which completes the proof. □

Claim 2.4. $\mathbf{E}[\textit{cost of the solution}] \leq 2\mathsf{Opt}_{LP}$

Proof. Since $\mathbf{E}[\text{cost of the solution}] = \sum_e c_e \cdot \mathbf{Pr}[e \text{ is cut}]$, it suffices to show that the probability that an edge e is cut is at most $2x_e$. This is clearly the case if $x_e \geq 1/2$. For $x_e \in [0, \frac{1}{2})$, we have that the probability e is "cut" is exactly $\frac{x_e}{1/2} = 2x_e$, proving the claim. □

These two claims translate directly to the following theorem:

Theorem 2.5. *The algorithm is a factor-2 randomized algorithm for the multicut problem on trees.*

We can also derandomize this algorithm: since $R \in [0, \frac{1}{2}]$ can give rise to only a small number of possible cuts, we can try all of them. The claim above shows that one of these cuts is at most 2 times the optimal LP value.

Aside: There are other algorithms for multicut on trees: Garg, Vazirani, and Yannakakis [GVY97] gave a primal-dual algorithm. More recently, Golovin, Nagarjan, and Singh [GNS06] use total-unimodularity to give a slightly different algorithm—this is one step towards solving a generalization of the multicut problem, in which we only need to disconnect at least ℓ out of k terminal pairs.

3 Multicut on General Graphs

We now turn our attention to the general graphs. Like before, we will formulate an LP relaxation for this problem:

$$\begin{aligned} \text{Minimize} \quad & \sum_{e \in E} c_e x_e \\ \text{Subject to} \quad & \sum_{e \in \mathcal{P}} x_e \geq 1 \quad \text{for all path } \mathcal{P} \text{ between } s_i \text{ and } t_i, \forall i = 1, 2, \ldots, k. \end{aligned}$$

This LP can have exponentially many constraints, but we have a simple separation oracle: the Floyd-Warshall algorithm can solve the all-pair shortest path problem in $O(n^3)$, and then we can check if all the terminal pairs are at least unit distance apart. In fact, this problem also has a compact LP formulation, which we leave as an exercise.

We will assume the following properties without loss of generality: 1) the graph is a complete graph and hence we have a variable x_{ij} for all $i \neq j \in V$, and 2) the edge lengths $\{x_{ij}\}$ satisfy the triangle inequality, and thus the distance function d defined over $\{x_{ij}\}$, together with V, forms a metric space. These properties allow us to apply a handy tool, called *low-diameter decomposition*, which we now state and use as a black box. In the next section, we will present a construction for this decomposition.

Low-diameter Decomposition: Given a metric space (V, d) on $|V| = n$ points and a parameter $D \in \mathbb{R}_+$, there is partition of V into $S_1 \uplus S_2 \uplus S_3 \uplus \cdots \uplus S_t$ such that

1. **(Low Diameter)** For each $i = 1, \ldots, t$, the distance $d(u, v) \leq D$ for all $u, v \in S_i$.
2. **(Low Cutting Cost)**
$$\sum_{e \in \text{the cut}} c_e \leq \frac{O(\log n)}{D} \sum_e c_e d_e,$$
where an edge uv belongs to the cut if $u \in S_i$ and $v \in S_j$ with $i \neq j$.

Using the low-diameter decomposition as a black box, we propose the following algorithm:

1. Solve the LP
2. Apply low-diameter decomposition (LDD) with $D = 1 - \varepsilon$.
3. The cut includes all edges going between S_i and S_j where $i \neq j$.

Already, we can see that the cut disconnects all terminal pairs, because $D < 1$ (property 1) and $d(s_i, t_i)$ must be at least 1 in the LP solution. We also know that the cutting cost—i.e., the cost of the multicut—is at most $O(\log n)\frac{1}{1-\varepsilon}\sum_e c_e x_e = O(\log n)\mathsf{Opt}_{\mathsf{LP}}$ (property 2). Hence, we have the following theorem.

Theorem 3.1. *The algorithm gives a factor-$O(\log n)$ approximation.*

4 Construction of Low-diameter Decomposition

The low-diameter decomposition technique has found many applications. Indeed, the technique has been discovered and rediscovered numerous times in various contexts [LS93, GVY04, LR99]. In this section, we present a randomized construction due to Carlinescu, Karloff, and Rabani [CKR01] (see also [FRT04] for a summary and generalizations). Instead of working with the conditions of the previous section, we propose the following alternative:

1'. **(Low Radius)** For all S_i, there exists $c_i \in V$ such that for all $u \in S_i$, $d(c_i, u) \leq r$.

2'. **(Low Cutting Probability)** For each edge e, $\mathbf{Pr}[e \text{ belongs to the cut}] \leq \frac{d_e}{r} O(\log n)$.

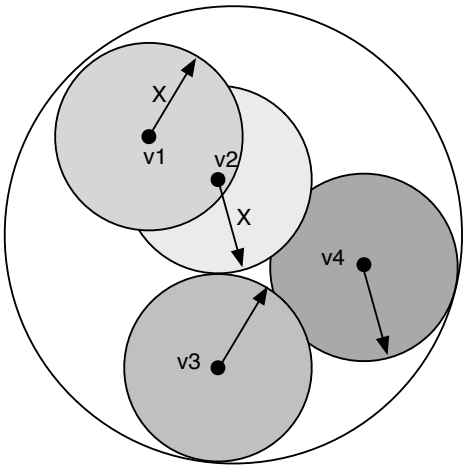
Observation 4.1. *Conditions* 1' *and and* 2' *together imply the original low-diameter decomposition properties in expectation.*

Clearly, if we set $r = D/2$, condition 1' implies condition 1; condition 2' implies condition 2 in expectation. One can imagine using the following procedure to guarantee condition 2: repeat the randomized construction until condition 2 is satisfied. Note that with probability at least $1/2$, the cost of the cut is within 2 times of the expectation; therefore, we only have to repeat the construction 2 times in expectation.

Algorithm: The algorithm performs the following steps:

1. Choose $X \in [\frac{r}{2}, r]$ uniformly at random.
2. Choose a random permutation $\pi : V \to V$ uniformly at random.
3. Consider the vertices one by one, in the order given by π. When we consider w, we assign all the yet-unassigned vertices v with $d(w, v) \leq X$ to w's partition.

For example, suppose the ordering given by π is v_1, v_2, v_3, v_4. The figure below illustrates the coverage when the vertices are visited by this process.



This construction directly implies the low-radius property, restated in the following claim.

Claim 4.2 (Low Radius)**.** *The output of the algorithm has the property that for all S_i, there exists $c_i \in V$ such that for all $u \in S_i$, $d(c_i, u) \leq r$.*

The real work is in showing that the construction has a low cutting probability.

Claim 4.3 (Low Cutting Probability)**.** *For each edge e,*

$$\mathbf{Pr}[e \textit{ belongs to the cut}] \leq \frac{d_e}{r} O(\log n).$$

Before proving this claim, let us state two definitions useful for the proof. Let $e = (u, v)$ be an edge in this graph. For the analysis only: suppose we renumber the vertices $w_1, w_2, \ldots, w_n$ in order of the distance from the edge $e = (u, v)$.

- **(Settling)** At some time instant in this procedure, one (or both) of u or v gets assigned to some w_i. We say that w_i *settles* the edge (u, v).
- **(Cutting)** At the moment the edge is settled, if only one endpoint of this edge is assigned, then we say that w_i *cuts* the edge (u, v).

According to these definitions, each edge is settled at exactly one time instant in the procedure, and it may or may not be cut at that time. Of course, once the edge is settled (with or without being cut), it is never cut in the future.

Proof. Consider w_j, and let $d(w_j, u) = a_j$ and $d(w_j, v) = b_j$. Assume $a_j < b_j$ (the other case is identical). If w_j cuts $e = (u, v)$ when the random values are X and π, the following two properties must hold:

1. The random variable X must lie in the interval $[a_j, b_j]$ (else either none or both endpoints of e would get marked).
2. The node w_j must come before $w_1, \ldots, w_{j-1}$ in the permutation π.

Suppose not, and one of them came before w_j in the permutation. Since all these vertices are closer to the edge than w_j is, then for the current value of X, they would have settled the edge (either capturing one or both of the endpoints) at some previous time point, and hence w_j would not settle—and hence not cut—the edge (u, v).

With these two properties, we establish

$$\begin{aligned}
\mathbf{Pr}[\text{edge } e \text{ is cut}] &= \sum_j \mathbf{Pr}[w_j \text{ cuts the edge } e] \\
&\leq \sum_j \mathbf{Pr}[X \in [a_j, b_j] \text{ and } w_j \text{ comes before } w_1, \ldots, w_{j-1} \text{ in } \pi] \\
&\leq \sum_j \frac{d_{uv}}{r/2} \cdot \frac{1}{j} \leq \frac{2d_e}{r} \cdot H_n = \frac{d_e}{r} O(\log n)
\end{aligned}$$

□

References

[CKR01] Gruia Calinescu, Howard J. Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. In *SODA*, pages 8–16, 2001.

[FF62] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

[GNS06] Daniel Golovin, Viswanath Nagarajan, and Mohit Singh. Approximating the k-multicut problem. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 621–630, New York, NY, USA, 2006. ACM.

[GVY97] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18(1):3–20, 1997.

[GVY04] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Multiway cuts in node weighted graphs. *J. Algorithms*, 50(1):49–61, 2004.

[Hu63] T. C. Hu. Multi-commodity network flows. *Operations Research*, 11(3):344–360, 1963.

[LR99] Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[LS93] Nathan Linial and Michael E. Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

[Sey79] Paul D. Seymour. A short proof of the two-commodity flow theorem. *J. Comb. Theory, Ser. B*, 26(3):370–371, 1979.