

Lecture 5

Lecturer: Debmalya Panigrahi

Scribe: Kevin Sun

1 Overview

In this lecture, we focus on the dual of the maximum multicommodity flow problem. This problem is known as the multicut problem, and we present an approximation algorithm. Along the way, we see a simpler algorithm for a special case known as the multiway cut problem.

2 The Multicut Problem

Let us recall the *dual* of the maximum multicommodity flow problem for *undirected* graphs. We are given a graph $G = (V, E)$ with edge costs (formerly capacities) $c(e)$, and a set of source-sink pairs (s_i, t_i) for $i \in \{1, 2, \dots, k\}$. In the dual, we seek to find edge lengths that minimize the total volume while separating every s_i - t_i pair. Let P_i denote the set of s_i - t_i paths, and let P be the union of P_i for $i = 1, \dots, k$. Written as an LP, this problem is the following:

$$\begin{aligned} \text{(D): } \min \quad & \sum_{e \in E} c(e) \ell_e \\ & \sum_{e \in p} \ell_e \geq 1 \quad \forall p \in P \\ & \ell_e \geq 0 \quad \forall e \in E \end{aligned}$$

Now suppose we impose the integrality constraints, i.e., we now require $\ell_e \in \{0, 1\}$ for every $e \in E$. The resulting problem is known as the *multicut problem*: we seek to find a minimum-cost subset of edges whose removal separates every s_i - t_i pair.

The multicut problem is NP-hard, which means there is little hope for a polynomial-time algorithm. However, as we will see, we can obtain a fast approximation algorithm. But first, we will first direct our attention to a special case known as the multiway cut problem.

2.1 The Multiway Cut Problem

In this problem, we are given an undirected graph $G = (V, E)$, edge costs $c(e) \geq 0$ for every $e \in E$, and a set "terminals" $s_1, \dots, s_k \in V$. Our goal is to find a minimum-cost subset of edges whose removal disconnects s_i from s_j for every $i \neq j$.

Notice that this problem is a generalization of the minimum s - t cut problem (in which $k = 2$), and the multicut problem is a generalization of this problem. Given a set of terminals for the multiway cut problem, we can set every (s_i, s_j) pair as a source-sink pair to obtain an equivalent instance of the multicut problem.

Our algorithm for this problem is a natural extension of an algorithm for minimum s - t cut we saw in Lecture 2. There are essentially just two differences: the random threshold is now chosen in $(0, 1/2)$ rather than $(0, 1)$, and our output is union over all terminals.

Algorithm 1 A randomized algorithm for Multiway Cut

- 1: Solve the LP (D) to obtain $\ell^* : E \rightarrow \mathbb{R}_{\geq 0}$
 - 2: $d(u, v) \leftarrow$ shortest path distance from u to v under ℓ^*
 - 3: Choose $r \in [0, 1/2)$ uniformly at random
 - 4: **for** $i = 1, \dots, k$ **do**
 - 5: $S_i \leftarrow \{v : d(s_i, v) \leq r\}$
 - 6: **return** $\cup_{i=1}^k \delta(S_i)$
-

Theorem 1. *Algorithm 1 is a 2-approximation for the multiway cut problem.*

Proof. Let F be the output of Algorithm 1 and $c(F)$ denote the (expected) cost of F . We first note that F is feasible because, by the feasibility of ℓ^* , we do not have any $d(s_i, s_j) \leq r$ for any $s_i \neq s_j$.

Now we also observe that the S_i are pairwise disjoint due to the triangle inequality. Indeed, if there exists u and $s_i \neq s_j$ such that $d(s_i, u) \leq r$ and $d(s_j, u) \leq r$, then $d(s_i, s_j) \leq r + r \leq 1$, which violates feasibility of ℓ^* .

Using this, we can bound the probability that an edge (x, y) is in F . Since all of the S_i are disjoint, there are only two ways for this to happen: x is in S_u for some u , or y is in S_v for some v . For the former, we need $r \in [d(s_u, x), 1/2)$ and for the latter, we need $r \in [d(s_v, y), 1/2)$. Thus, by the union bound, we have

$$\Pr((x, y) \in F) \leq \frac{1/2 - d(s_u, x)}{1/2} + \frac{1/2 - d(s_v, y)}{1/2} = 2(1 - d(s_u, x) - d(s_v, y)) \leq 2d(x, y), \quad (1)$$

where the final inequality follows from the triangle inequality and the fact that $1 \leq d(s_u, s_v)$.

We can now bound the expected cost of Algorithm 1. Let OPT denote the cost of the optimal multiway cut solution. From (1) and the fact that ℓ^* is the optimal fractional solution, we have

$$c(F) = \sum_{e \in E} c(e) \Pr(e \in F) \leq 2 \sum_{e \in E} c(e) \ell_e^* \leq 2 \cdot OPT. \quad \square$$

Remark: Although Algorithm 1 is simple and elegant, there are algorithms for multiway cut with better guarantees. In particular, Călinescu et al. [CKR00] gave a 3/2-approximation which works by embedding the vertices of the graph on a simplex with the terminals embedded into the corners. And this isn't the best approximation either—by using a technique known as exponential clocks, Buchbinder et al. [BNS13] obtain a 4/3-approximation. Whether or not this is the best approximation guarantee remains an open question.

2.2 The CKR Algorithm for Multicut

Now we turn our attention back to the multicut problem. Recall that we have k s_i - t_i pairs, and we must find a subset of edges that separates s_i from t_i for every $i \in \{1, \dots, k\}$. Now let us establish some notation: for any $S \subseteq V$, we let $\delta(S)$ denote the set of edges cut by S and $I(S)$ denote the set of edges with both endpoints in S . In other words, we have

$$\begin{aligned} \delta(S) &= \{(u, v) \in E : u \in S, v \notin S\} \\ I(S) &= \{(u, v) \in E : u, v \in S\}. \end{aligned}$$

We are now ready to state the algorithm for multicut given by Călinescu et al. [CKR05]. At a high level, it is somewhat similar to Algorithm 1 for multiway cut: they both set a threshold r and consider the vertices S_i within a distance of r from each s_i and the set of edges $\delta(S_i)$.

However, there are two key differences in Algorithm 2: the vertices are processed in a random order, and not *all* of the edges crossing each S_i are added to the solution. Instead, the algorithm excludes the edges that are entirely within previously processed balls. It might seem that the solution is no longer feasible, but we will see that this is not the case.

Algorithm 2 The CKR Algorithm for the Multicut problem

- 1: Solve the LP (D) to obtain $\ell^* : E \rightarrow \mathbb{R}_{\geq 0}$
 - 2: $d(u, v) \leftarrow$ shortest path distance from u to v under ℓ^*
 - 3: Choose $r \in [0, 1/2)$ uniformly at random
 - 4: Choose a permutation σ of $\{1, \dots, k\}$ uniformly at random
 - 5: **for** $i = 1, \dots, k$ **do**
 - 6: $S_{\sigma(i)} \leftarrow \{v : d(s_{\sigma(i)}, v) \leq r\}$
 - 7: $E_{\sigma(i)} \leftarrow \delta(S_{\sigma(i)}) \setminus \cup_{j < i} I(S_{\sigma(j)})$
 - 8: **return** $\cup_{i=1}^k E_i$
-

Lemma 2. *The output of Algorithm 2 is a feasible multicut solution.*

Proof. Let F denote the output of Algorithm 2, and for contradiction, suppose $s_{\sigma(i)}$ is connected to $t_{\sigma(i)}$ for some i after the removal of F . Notice that when we remove $E_{\sigma(i)}$, we disconnect $s_{\sigma(i)}$ from $V \setminus S_{\sigma(i)}$ except for possibly vertices in sets of the form $S_{\sigma(j)}$ where $j < i$ and $s_{\sigma(i)} \in S_{\sigma(j)}$. But by the triangle inequality, $t_{\sigma(i)}$ cannot be in any such set (or $d(s_i, t_i)$ would be less than 1). \square

Now that we know the algorithm is feasible, we are ready to bound the approximation ratio of Algorithm 2. But before we do so, we briefly motivate the reason Algorithm 2 fixes a random permutation σ . Fix an edge (x, y) and suppose we had the following:

$$\begin{aligned} d(s_k, x) = 0, d(s_{k-1}, x) = \frac{1}{2k}, \dots, d(s_1, x) = \frac{k-1}{2k} \\ d(s_k, y) = \frac{1}{2k}, d(s_{k-1}, y) = \frac{2}{2k}, \dots, d(s_1, y) = \frac{k}{2k}. \end{aligned}$$

If we process the terminals in the order s_1, s_2, \dots, s_k , then regardless of the value of r , the edge (x, y) is cut (i.e., returned by Algorithm 2) with probability 1. This is clearly not a desirable outcome, but we can rectify this by processing the terminals in reverse order. In this case, (x, y) is likely to be an internal edge by the time the algorithm considers cutting it, so it is unlikely to get cut.

We now bound the approximation ratio of Algorithm 1. For any integer $n \geq 0$, we let H_n denote the n -th *Harmonic number*, defined as $H_n = 1 + 1/2 + \dots + 1/n$.

Theorem 3. *Algorithm 2 is a $2H_k$ -approximation for the multicut problem.*

Proof. Let F be the output of Algorithm 2, and fix an edge $e = (x, y)$. Notice that it suffices to prove $\Pr(e \in F) \leq 2H_k \cdot d(x, y)$; the rest of the proof is identical to that of Theorem 1.

Let $A_i(x, y)$ denote the event $(x, y) \in \delta(S_i)$ and let $B_i(x, y)$ denote the event $(x, y) \notin I(S_{\sigma(j)})$ for any $j < i$. Then (x, y) is in F if only if $A_i(x, y)$ and $B_i(x, y)$ for some $i \in \{1, \dots, k\}$. Applying the union bound, we have the following:

$$\Pr(e \in F) \leq \sum_{i=1}^k \Pr\left(A_{\sigma(i)}(x, y) \wedge B_i(x, y)\right). \quad (2)$$

Recall that the probability an (x, y) is cut by any S_i , as we saw in the proof of Theorem 1, at most $2d(x, y)$. So for any i , we have

$$\Pr\left(A_{\sigma(i)}(x, y)\right) \leq 2d(x, y). \quad (3)$$

Furthermore, notice that $A_i(x, y)$ only depends on the radius r , not the permutation σ . However, the event $B_i(x, y)$ depends on both r and σ , so $A_i(x, y)$ and $B_i(x, y)$ are not independent.

To circumvent this, let us suppose the events $A_{\sigma(i)}(x, y)$ and $B_i(x, y)$ first occur together at some step i . Then (x, y) wasn't in, or cut by, $S_{\sigma(j)}$ for any $j < i$, so both endpoints of the edge (x, y) must be outside $S_{\sigma(j)}$. In particular, this means

$$d(s_{\sigma(i)}, x) < d(s_{\sigma(j)}, x) \quad \forall j < i. \quad (4)$$

Let $C_i(x)$ denote the above event, so $C_i(x)$ only depends on the permutation σ (not the radius r). From the discussion above, we have

$$\begin{aligned} \Pr\left(A_{\sigma(i)}(x, y) \wedge B_i(x, y)\right) &\leq \Pr\left(A_{\sigma(i)}(x, y) \wedge C_i(x)\right) \\ &= \Pr\left(A_{\sigma(i)}(x, y)\right) \cdot \Pr(C_i(x)) \\ &\leq 2d(x, y) \cdot \Pr(C_i(x)), \end{aligned}$$

where the equality follows from the independence of σ and r and the last line follows from (3).

We now show $\Pr(C_i(x)) \leq 1/i$ for any i ; for $i = 1$ this is trivially true. Let us consider $i = 2$: in this case, σ must satisfy $d(s_{\sigma(2)}, x) < d(s_{\sigma(1)}, x)$, and since σ is chosen uniformly at random, this happens with probability $1/2$. In general, we see that for $C_i(x)$ to occur, σ must assign i so that $d(s_{\sigma(i)}, x)$ is the smallest among i distance values, and this happens with probability $1/i$.

Putting this all together, we have

$$\Pr(e \in F) \leq 2d(x, y) \sum_{i=1}^k \Pr(C_i(x)) \leq 2d(x, y) \sum_{i=1}^k \frac{1}{i} = 2H_k \cdot d(x, y). \quad \square$$

3 Summary

In this lecture, we saw a simple randomized rounding algorithm for the multiway cut problem, a generalization of the minimum s - t cut problem. We also used some similar ideas to give an $O(\log k)$ -approximation for the multicut problem, where k is the number of source-sink pairs.

References

- [BNS13] Niv Buchbinder, Joseph Seffi Naor, and Roy Schwartz. Simplex partitioning via exponential clocks and the multiway cut problem. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 535–544. ACM, 2013.
- [CKR00] Grăia Călinescu, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for multiway cut. *Journal of Computer and System Sciences*, 60(3):564–574, 2000.
- [CKR05] Grăia Călinescu, Howard Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. *SIAM Journal on Computing*, 34(2):358–372, 2005.