Minimum Multicut

Vazirani: Chapter 18

Let $G = (V, E)$ be an undirected weighted graph with weights $c_e > 0$ for all edges $e \in E$.

Let $\{(s_1, t_1),\ (s_2, t_2), \ldots, (s_k, t_k)\}$ be k specified pairs of vertices. A *multicut* is a set of edges whose removal separates all of the pairs. The problem is to find a minimum weight multicut in G.

If we could solve this problem in polynomial time then we could also solve the *multiway cut* problem in polynomial time. Since multiway-cut is known to be NP-hard, this problem is also NP-Hard.

In this lesson we will see how to use the Primal-Dual Schema to design a 2-approximation algorithm for the special case when G is a tree. This case can be polynomially reduced to *minimal vertex cover* (see Vazirani, Exercise 18.1) so it is also NP-Hard.

If G is a tree then, for each (s_i, t_i) there is a *unique* path connecting s_i to t_i. The minimum multicut removes at least one edge from this path.

For each $e \in E$ let $d_e \in \{0, 1\}$ be a variable such that

$$d_e = 1 \text{ iff } e \text{ is in the multicut.}$$

Let p_i denote the set of edges on the unique path connecting s_i and t_i.

The integer LP corresponding to minimum multicut is

Minimize $\sum_{e \in E} c_e d_e$
subject to conditions

$$\forall i \in \{1, \ldots, k\}, \quad \sum_{e \in p_i} d_e \geq 1$$
$$\forall e \in E, \quad d_e \in \{0, 1\}$$

The relaxation of the LP is

Minimize $\sum_{e \in E} c_e d_e$
subject to conditions

$$\forall i \in \{1, \ldots, k\}, \quad \sum_{e \in p_i} d_e \geq 1$$
$$\forall e \in E, \quad d_e \geq 0$$

We now introduce a variable f_i corresponding to (s_i, t_i). The *dual* of the relaxed LP is then

Maximize $\sum_{i=1}^{k} f_i$
subject to conditions

$$\forall e \in E, \quad \sum_{i: e \in p_i} f_i \leq c_e$$
$$\forall i \in \{1, \ldots, k\}, \quad f_i \geq 0$$

Maximize $\sum_{i=1}^{k} f_i$
subject to conditions
$\forall e \in E, \quad \sum_{i: e \in p_i} f_i \le c_e$
$\forall i \in \{1, \ldots, k\}, \quad f_i \ge 0$

This dual can be thought of as describing the *multicommodity flow* problem. In this problem there are k different commodities with the ith commodity needing to be shipped from s_i to t_i. The object is to maximize the total amount shipped. The constraint is that the sum of the flows routed through any particular edge is at most c_e.

The *maximum integer multicommodity flow* problem is the multicommodity flow problem with the further restrictions that the f_i are all integers. Note that in this problem we may assume that the c_i are all integers as well; if they are not, we can round them down to $\lfloor c_i \rfloor$ without changing the maximum.

We will use the primal dual schema to derive an algorithm that simultaneously finds a multicut and an integer multicommodity flow that are within a factor of two of each other. This will give a 2-approximation algorithm for the minimum multicut problem *and* a $1/2$ "approximation algorithm" for the maximum integer multicommodity flow one.

Primal:

Minimize $\sum_{e \in E} c_e d_e$
subject to conditions
$\forall i \in \{1, \ldots, k\}, \quad \sum_{e \in p_i} d_e \geq 1$
$\forall e \in E, \quad d_e \geq 0$

Dual:

Maximize $\sum_{i=1}^{k} f_i$
subject to conditions
$\forall e \in E, \quad \sum_{i : e \in p_i} f_i \leq c_e$
$\forall i \in \{1, \ldots, k\}, \quad f_i \geq 0$

Edge $e \in E$ will be *saturated*
if total flow through e is c_e.
The complimentary slackness conditions will then be:

Primal: $\forall e \in E,\ d_e \neq 0, \Rightarrow \sum_{i : e \in p_i} f_i = c_e$.
This means *any edge picked in the multicut must be saturated*

Relaxed Dual: $\forall i,\ f_i \neq 0 \Rightarrow \sum_{e \in p_i} d_e \leq 2$.
This means *at most two edges can be picked from a path carrying non-zero flow*

We start by rooting the tree G at an arbitrary vertex *Depth* of $v \in V$ will be length of path from v to the root. For $u, v \in V$,

let $lca(u, v)$ be the *lowest common ancestor* of u and v. Let e_1, e_2 be two edges on the same path from a vertex to the root. If e_1 occurs before e_2 on this path, e_1 is *deeper* than e_2.

Algorithm starts with an empty **multicut** (satisfies primal c.s. conditions) and **empty flow** (feasible). It then iteratively improves feasibility of primal solution and optimality of dual solution. The edges in the multicut so far will be kept in a list D.

During an iteration it picks the deepest unprocessed vertex so far, v and greedily routes integral flow between pairs that have v as their lca. When no more flow can be routed between these pairs at least one edge has been saturated. All saturated edges are added to D.

After all vertices have been processed D will be a multicut. D might contain extra edges, though. The algorithm ends by working through the edges of D in reverse order in which they were added; if after the removal of an edge D remains a multicut, the edge is removed from D.

Minimum Multicut/ Integer Multicommodity Flow (Trees)

1. $\forall i\ f_i = 0, \quad D = \emptyset.$
2. For all vertices v, in non-increasing order of depth, do:
   For every pair (s_i, t_i) with $lca(s_i, t_i) = v$
   Greedily route integral flow from s_i to t_i
   Add to D all edges e that were saturated in this iteration.
3. Let $e_1, e_2, \ldots, e_l$ be edges in D ordered by insertion time.
4. For $j = l$ downto 1 do.
   If $D - \{e_j\}$ is a multicut in G remove e_j from D.
5. Output flow and multicut D.

First note that the algorithm outputs a legal flow since it starts with the empty (legal) flow and at every step maintains legality.

Next note that at the end of the algorithm D must contain a multicut since at least one edge on the unique (s_i, t_i) path must have been added to D (WHY).
Now construct the integral $0/1$ solution $d_e = 1$ **iff** $e \in D$.

Constructed primal and dual solutions are therefore both feasible. Furthermore, since every edge picked in the multicut was saturated, the solutions satisfy primal c.s. conditions. $\forall e \in E,\ d_e \neq 0, \Rightarrow \sum_{i: e \in p_i} f_i = c_e.$

Minimum Multicut/ Integer Multicommodity Flow (Trees)

1. $\forall i \; f_i = 0, \quad D = \emptyset.$
2. For all vertices v, in non-increasing order of depth, do:
   For every pair (s_i, t_i) with $lca(s_i, t_i) = v$
      Greedily route integral flow from s_i to t_i
   Add to D all edges e that were saturated in this iteration.
3. Let $e_1, e_2, \ldots, e_l$ be edges in D ordered by insertion time.
4. For $j = l$ downto 1 do.
   If $D - \{e_j\}$ is a multicut in G remove e_j from D.
5. Output flow and multicut D.

Constructed primal and dual solutions are both feasible and satisfy the primal c.s. conditions.

Lemma: Let (s_i, t_i) be a pair with non-zero flow and let $lca(s_i, t_i) = v$. Then at most one edge is picked in the multicut from each of the two paths s_i to v and v to t_i.

This lemma implies that, for each (s_i, t_i), at most two edges from D are on the path connecting s_i to t_i. Thus the relaxed dual conditions

$$\forall i, \; f_i \neq 0 \Rightarrow \sum_{e: e \in p_i} d_e \leq 2$$

are all satisfied as well.

Combining everything: since d_e and f_i are feasible solutions that satisfy the relaxed c.s. conditions with $\alpha = 2$ we have that

$$\sum_i f_i \leq \sum_e c_e d_e \leq 2 \cdot \sum_i f_i$$

and D has weight within a factor of two optimal solution of a minimum multicut.

This can also be read as

$$\frac{1}{2} \sum_e c_e d_e \leq \sum_i f_i \leq \sum_e c_e d_e$$

implying that the solution is also a $\frac{1}{2}$ "approximation" algorithm for integral multicommodity flow as well.

Lemma: Let (s_i, t_i) be a pair with non-zero flow and let $lca(s_i, t_i) = v$. then at most one edge is picked in the multicut from each of the two paths s_i to v and v to t_i.

Proof: We prove for the s_i to v path. Proof for v to t_i path is the same.

Suppose at the end of the algorithm there are two edges $e, e' \in D$ from the same s_i to v path with e being deeper. By definition e' must remain in D all through step 4.

Consider the moment in step 4 when e is being tested. e is not thrown away so there must be a pair (s_j, t_j) such that$e$ is the only edge on the path between s_j and t_j. Let $u = lca(s_j, t_j)$. Since e' is not on path between s_j and t_j (why) u must be deeper than e' and therefore deeper than v.

This in turn implies that after u was processed D must have contained an edge from the path between s_j and t_j. Call this edge e''.

Now, since non-zero flow was routed from s_i to t_i, e *must* have been added to D during or after the iteration that processed v. Since v is an ancestor of u, e is added after e''. But then e'' must be in D when e is being tested, contradicting fact that at this time e is only edge of D on the path between s_j and t_j.