18 Multicut and Integer Multicommodity Flow in Trees

The theory of cuts in graphs occupies a central place not only in the study of exact algorithms, but also approximation algorithms. We will present some key results in the next four chapters. This will also give us the opportunity to develop further the two fundamental algorithm design techniques introduced in Chapters 14 and 15.

In Chapter 15 we used the primal-dual schema to derive a factor 2 algorithm for the weighted vertex cover problem. This algorithm was particularly easy to obtain because the relaxed dual complementary slackness conditions were automatically satisfied in any integral solution. In this chapter, we will use the primal-dual schema to obtain an algorithm for a generalization of this problem (see Exercise 18.1). This time, enforcing relaxed dual complementary slackness conditions will be a nontrivial part of the algorithm. Furthermore, we will introduce the procedure of reverse delete, which will be used in several other primal-dual algorithms.

18.1 The problems and their LP-relaxations

The following is an important generalization of the minimum \( s-t \) cut problem. In fact, it also generalizes the multiway cut problem (Problem 4.1).

**Problem 18.1 (Minimum multicut)** Let \( G = (V, E) \) be an undirected graph with nonnegative capacity \( c_e \) for each edge \( e \in E \). Let \( \{(s_1, t_1), \ldots, (s_k, t_k)\} \) be a specified set of pairs of vertices, where each pair is distinct, but vertices in different pairs are not required to be distinct. A **multicut** is a set of edges whose removal separates each of the pairs. The problem is to find a minimum capacity multicut in \( G \).

The minimum \( s-t \) cut problem is the special case of multicut for \( k = 1 \). Problem 18.1 generalizes multiway cut because separating terminals \( s_1, \ldots, s_l \) is equivalent to separating all pairs \( (s_i, s_j) \), for \( 1 \leq i < j \leq l \). This observation implies that the minimum multicut problem is \( \text{NP} \)-hard even for \( k = 3 \), since the multiway cut problem is \( \text{NP} \)-hard for the case of 3 terminals.

In Chapter 20 we will obtain an \( O(\log k) \) factor approximation algorithm for the minimum multicut problem. In this chapter, we will obtain a factor 2 algorithm for the special case when \( G \) is restricted to be a tree. Since \( G \) is
a tree, there is a unique path between \( s_i \) and \( t_i \), and the multicut must pick an edge on this path to disconnect \( s_i \) from \( t_i \). Although the problem looks deceptively simple, Exercise 18.1 should convince the reader that this is not so. The minimum multicut problem is \( \text{NP} \)-hard even if restricted to trees of height 1 and unit capacity edges.

Since we want to apply LP-duality theory to design the algorithm, let us first give an integer programming formulation for the problem and obtain its LP-relaxation. Introduce a \( 0/1 \) variable \( d_e \) for each edge \( e \in E \), which will be set to 1 iff \( e \) is picked in the multicut. Let \( p_i \) denote the unique path between \( s_i \) and \( t_i \) in the tree.

\[
\begin{align*}
\text{minimize} & \quad \sum_{e \in E} c_e d_e \\
\text{subject to} & \quad \sum_{e \in p_i} d_e \geq 1, \quad i \in \{1, \ldots, k\} \\
& \quad d_e \in \{0, 1\}, \quad e \in E
\end{align*}
\]

The LP-relaxation is obtained by replacing the constraint \( d_e \in \{0, 1\} \) by \( d_e \geq 0 \). As in the derivation of LP (13.2), there is no need to add the constraint \( d_e \leq 1 \) explicitly.

\[
\begin{align*}
\text{minimize} & \quad \sum_{e \in E} c_e d_e \quad \quad (18.1) \\
\text{subject to} & \quad \sum_{e \in p_i} d_e \geq 1, \quad i \in \{1, \ldots, k\} \\
& \quad d_e \geq 0, \quad e \in E
\end{align*}
\]

We can now think of \( d_e \) as specifying the fractional extent to which edge \( e \) is picked. A solution to this linear program is a \textit{fractional multicut}: on each path \( p_i \), the sum of fractions of edges picked is at least 1. In general, minimum fractional multicut may be strictly cheaper than minimum integral multicut. This is illustrated in Example 18.2.

We will interpret the dual program as specifying a \textit{multicommodity flow} in \( G \), with a separate commodity corresponding to each vertex pair \((s_i, t_i)\). Dual variable \( f_i \) will denote the amount of this commodity routed along the unique path from \( s_i \) to \( t_i \).

\[
\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{k} f_i \quad \quad (18.2) \\
\text{subject to} & \quad \sum_{e \in p_i} f_i \leq c_e, \quad e \in E \\
& \quad f_i \geq 0, \quad i \in \{1, \ldots, k\}
\end{align*}
\]
The commodities are routed concurrently. The object is to maximize the sum of the commodities routed, subject to the constraint that the sum of flows routed through an edge is bounded by the capacity of the edge. Notice that the sum of flows through an edge \((u, v)\) includes flow going in either direction, \(u\) to \(v\) and \(v\) to \(u\).

By the weak duality theorem, a feasible multicommodity flow gives a lower bound on the minimum fractional multicut and hence also on the minimum integral multicut. By the LP-duality theorem, minimum fractional multicut equals maximum multicommodity flow.

**Example 18.2** Consider the following graph with unit capacity edges and 3 vertex pairs:

![Diagram](image)

The arrows show how to send 3/2 units of flow by sending 1/2 unit of each commodity. Picking each edge to the extent of 1/2 gives a multicut of capacity 3/2 as well. These must be optimal solutions to the primal and dual programs. On the other hand, any integral multicut must pick at least two of the three edges in order to disconnect all three pairs. Hence, minimum integral multicut has capacity 2.

Finally, let us state one more problem.

**Problem 18.3 (Integer multicommodity flow)** Graph \(G\) and the source-sink pairs are specified as in the minimum multicut problem; however, the edge capacities are all integral. A separate commodity is defined for each \((s_i, t_i)\) pair. The object is to maximize the sum of the commodities routed, subject to edge capacity constraints and subject to routing each commodity integrally.

Let us consider this problem when \(G\) is restricted to be a tree. If in (18.2), the variables are constrained to be nonnegative integers, we would get an integer programming formulation for this problem. Clearly, the objective function value of this integer program is bounded by that of the linear program (18.2).
Furthermore, the best fractional flow may be strictly larger. For instance, in Example 18.2, maximum integral multicommodity flow is 1, since sending 1 unit of any of the three commodities will saturate two of the edges. This problem is NP-hard, even for trees of height 3 (though the capacity has to be arbitrary).

18.2 Primal–dual schema based algorithm

We will use the primal–dual schema to obtain an algorithm that simultaneously finds a multicut and an integer multicommodity flow that are within a factor of 2 of each other, provided the given graph is a tree. Hence, we get approximation algorithms for both problems, of factor 2 and 1/2, respectively.

Let us define the multicut LP to be the primal program. An edge $e$ is saturated if the total flow through it equals its capacity. We will ensure primal complementary slackness conditions, i.e., $\alpha = 1$, and relax the dual conditions with $\beta = 2$, where $\alpha$ and $\beta$ are the parameters used in the general description of the primal–dual schema given in Chapter 15.

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

**Relaxed dual conditions:** For each $i \in \{1, \ldots, k\}$, $f_i \neq 0 \Rightarrow \sum_{e \in p_i} d_e \leq 2$.
Equivalently, at most two edges can be picked from a path carrying nonzero flow. (Clearly, we must pick at least one edge from each $(s_i, t_i)$ path simply to ensure the feasibility of the multicut.)

Let us root the tree $G$ at an arbitrary vertex. Define the depth of vertex $v$ to be the length of the path from $v$ to the root; the depth of the root is 0. For two vertices $u, v \in V$, let $\text{lca}(u, v)$ denote the lowest common ancestor of $u$ and $v$, i.e., the minimum depth vertex on the path from $u$ to $v$. Let $e_1$ and $e_2$ be two edges on a path from a vertex to the root. If $e_1$ occurs before $e_2$ on this path, then $e_1$ is said to be deeper than $e_2$.

The algorithm starts with an empty multicut and flow, and iteratively improves the feasibility of the primal solution and the optimality of the dual solution. In an iteration, it picks the deepest unprocessed vertex, say $v$, and greedily routes integral flow between pairs that have $v$ as their lowest common ancestor. When no more flow can be routed between these pairs, all edges that were saturated in this iteration are added to the list $D$ in arbitrary order. When all the vertices have been processed, $D$ will be a multicut; however, it may have redundant edges. To remove them, a reverse delete step is performed: edges are considered in the reverse of the order in which they were added to $D$, and if the deletion of edge $e$ from $D$ still gives a valid multicut, $e$ is discarded from $D$. 
Algorithm 18.4 (Multicut and integer multicommodity flow in trees)

1. **Initialization:** $f \leftarrow 0; D \leftarrow \emptyset$.
2. **Flow routing:** For each vertex $v$, in nonincreasing order of depth, do:
   - For each pair $(s_i, t_i)$ such that lca$(s_i, t_i) = v$, greedily route integral flow from $s_i$ to $t_i$.
   - Add to $D$ all edges that were saturated in the current iteration in arbitrary order.
3. Let $e_1, e_2, \ldots, e_l$ be the ordered list of edges in $D$.
4. **Reverse delete:** For $j = l$ downto 1 do:
   - If $D \setminus \{e_j\}$ is a multicut in $G$, then $D \leftarrow D \setminus \{e_j\}$.
5. Output the flow and multicut $D$.

**Lemma 18.5** Let $(s_i, t_i)$ be a pair with nonzero flow, and let lca$(s_i, t_i) = v$. At most one edge is picked in the multicut from each of the two paths, $s_i$ to $v$ and $t_i$ to $v$.

**Proof:** The argument is the same for each path. Suppose two edges $e$ and $e'$ are picked from the $s_i$-$v$ path, with $e$ being the deeper edge. Clearly, $e'$ must be in $D$ all through reverse delete. Consider the moment during reverse delete when edge $e$ is being tested. Since $e$ is not discarded, there must be a pair, say $(s_j, t_j)$, such that $e$ is the only edge of $D$ on the $s_j$-$t_j$ path. Let $u$ be the lowest common ancestor of $s_j$ and $t_j$. Since $e'$ does not lie on the $s_j$-$t_j$ path, $u$ must be deeper than $e'$, and hence deeper than $v$. After $u$ has been processed, $D$ must contain an edge from the $s_j$-$t_j$ path, say $e''$.

Since nonzero flow has been routed from $s_i$ to $t_i$, $e$ must be added during or after the iteration in which $v$ is processed. Since $v$ is an ancestor of $u$, $e$ is added after $e''$. So $e''$ must be in $D$ when $e$ is being tested. This contradicts the fact that at this moment $e$ is the only edge of $D$ on the $s_j$-$t_j$ path.  

\[\square\]
Theorem 18.6 Algorithm 18.4 achieves approximation guarantees of factor 2 for the minimum multicut problem and factor 1/2 for the maximum integer multicommodity flow problem on trees.

Proof: The flow found at the end of Step 2 is maximal, and since at this point $D$ contains all the saturated edges, $D$ is a multicut. Since the reverse delete step only discards redundant edges, $D$ is a multicut after this step as well. Thus, feasible solutions have been found for both the flow and the multicut.

Since each edge in the multicut is saturated, the primal conditions are satisfied. By Lemma 18.5, at most two edges have been picked in the multicut from each path carrying nonzero flow. Therefore, the relaxed dual conditions are also satisfied. Hence, by Proposition 15.1, the capacity of the multicut found is within twice the flow. Since a feasible flow is a lower bound on the optimal multicut, and a feasible multicut is an upper bound on the optimal integer multicommodity flow, the claim follows.

Finally, we obtain the following approximate min–max relation from Theorem 18.6:

Corollary 18.7 On trees with integer edge capacities,

$$\max_{\text{int. flow } F} |F| \leq \min_{\text{multicut } C} c(C) \leq 2 \cdot \max_{\text{int. flow } F} |F|,$$

where $|F|$ represents the value of flow function $F$ and $c(C)$ represents the capacity of multicut $C$.

In Chapter 20 we will present an $O(\log k)$ factor algorithm for the minimum multicut problem in general graphs; once again, the lower bound used is an optimal fractional multicut. On the other hand, no nontrivial approximation algorithms are known for the integer multicommodity flow problem in graphs more general than trees. As shown in Example 18.8, even for planar graphs, the integrality gap of an LP analogous to (18.2) is lower bounded by $n/2$, where $n$ is the number of source–sink pairs specified.

Example 18.8 Consider the following planar graph with $n$ source–sink pairs. Every edge is of unit capacity. Any pair of paths between the $i$th and $j$th source–sink pairs intersect in at least one unit capacity edge. The magnified part shows how this is arranged at each intersection. Thus, sending one unit of any commodity blocks all other commodities. On the other hand, half a unit of each commodity can be routed simultaneously.