Lecturer: Hung Q. Ngo Last update: November 30, 2006

Approximation Algorithms Based on the Primal-Dual Method

The *primal-dual method* (or primal-dual schema) is another means of solving linear programs. The basic idea of this method is to start from a feasible solution $\mathbf{y}$ to the dual program, then attempt to find a feasible solution $\mathbf{x}$ to the primal program that satisfies the complementary slackness conditions. If such an $\mathbf{x}$ cannot be found, it turns out that we can find a better $\mathbf{y}$ in terms of its objective value. Then, another iteration is started.

The above idea can also be modified to design approximation algorithms. An approximate solution to the primal IP and a feasible solution to the dual LP can be constructed simultaneously and improved step by step. In the end, the approximate solution can be compared with the dual feasible solution to estimate the approximation ratio. One of the key strengths of this method is that it often allows for a combinatorial algorithm (based on the primal/dual view) which is very efficient.

1 Motivations: duality-base algorithms for VERTEX COVER and SET COVER

Recall the (unweighted) VERTEX COVER problem. Given a graph $G = (V, E)$, taking all vertices of a maximal matching of G would give a vertex cover for G. This is a very efficient 2-approximation algorithm for the vertex cover problem.

The above algorithm runs much faster than the rounding algorithm we have seen. One might wonder how this algorithm looks from the angle of linear programming. The answer is a surprisingly nice one. Let us consider the linear relaxation of the integer program for vertex cover:

$$\begin{array}{lll} \min & \displaystyle\sum_{v \in V} x_v & \\ \text{subject to} & x_u + x_v \geq 1, & \forall uv \in E, \\ & x_v \geq 0, & \forall v \in V. \end{array} \tag{1}$$

The dual program is

$$\begin{array}{lll} \max & \displaystyle\sum_{uv \in E} y_{uv} & \\ \text{subject to} & \displaystyle\sum_{u:\ uv \in E} y_{uv} \leq 1, & \forall v \in V, \\ & y_{uv} \geq 0, & \forall uv \in E. \end{array} \tag{2}$$

An integral feasible solution to (2) corresponds to a matching of G. Based on the idea of a maximal matching in the view of this linear program, we define a feasible solution $\mathbf{y}$ to be *maximal* if there is no feasible solution $\mathbf{y}'$ for which $y'_{uv} \geq y_{uv}, \forall uv \in E$, and $\sum_{uv \in E} y'_{uv} > \sum_{uv \in E} y_{uv}$. *In other words,* $\mathbf{y}$ *is maximal iff we cannot increase any component of* $\mathbf{y}$ *without making it infeasible.*

Now that we had the linear programming semantics of a maximal matching, the next question is: what does it mean to take both vertices of the maximal matchings? Easy, this corresponds to setting $x_v = 1$ whenever $\sum_{u:\ uv \in E} y_{uv} = 1$.

Theorem 1.1. *Let* $\bar{\mathbf{y}}$ *be a maximal feasible solution to* (2)*, then the strategy of setting* $x_v^A = 1$ *whenever* $\sum_{u:\ uv \in E} \bar{y}_{uv} = 1$ *gives a* 2*-approximation to the* VERTEX COVER *problem.*

Proof. We first show that $\mathbf{x}^A$ indeed defines a feasible vertex cover. If there is an edge $uv \in E$ such that both x_u^A and x_v^A are 0, then we must have

$$\sum_{s:\, us\in E} \bar{y}_{us} < 1, \quad and \quad \sum_{t:\, tv\in E} \bar{y}_{tv} < 1.$$

But then $\bar{y}_{uv}$ can be increased by an amount of

$$\delta = \min\left\{ \left(1 - \sum_{s:\, us\in E} \bar{y}_{us}\right), \left(1 - \sum_{t:\, tv\in E} \bar{y}_{tv}\right) \right\},$$

contradicting the maximality of $\bar{\mathbf{y}}$.

Secondly, we need to verify the approximation ratio of 2. This could be done easily using weak duality. We know $\bar{\mathbf{y}}$ gives a lower bound on the optimal value for (1), which is a lower bound of the optimal vertex cover:

$$\sum_v x_v^A \le \sum_v \sum_{u:\, uv\in E} \bar{y}_{uv} = 2 \sum_{uv\in E} \bar{y}_{uv} \le 2 \cdot \text{OPT}.$$

□

This algorithm can be extended straightforwardly to the weighted case, while the matching idea does not extend that well. Instead of solving the weighted case, let us see how this idea can be extended to the WEIGHTED SET COVER problem.

Recall that in the WEIGHTED SET COVER problem, we are given a universe set U, and a collection $\mathcal{C}$ of subsets of U Implicitly, let $m = |U|$ and $n = |\mathcal{C}|$. Each set S in $\mathcal{C}$ is weighted with a non-negative integer weight w_S. The corresponding integer program is

$$\begin{aligned} \min \quad & \sum_{S\in\mathcal{C}} w_S x_S \\ \text{subject to} \quad & \sum_{S\ni i} x_S \ge 1, \quad \forall i \in U, \\ & x_S \in \{0,1\}, \quad \forall S \in \mathcal{C}. \end{aligned} \tag{3}$$

The LP relaxation for (3) is

$$\begin{aligned} \min \quad & \sum_{S\in\mathcal{C}} w_S x_S \\ \text{subject to} \quad & \sum_{S\ni i} x_S \ge 1, \quad \forall i \in U, \\ & x_S \ge 0, \quad \forall S \in \mathcal{C}. \end{aligned} \tag{4}$$

The corresponding dual program can be written as follows.

$$\begin{aligned} \max \quad & \textstyle\sum_{i\in U} y_i \\ \text{subject to} \quad & \sum_{i\in S} y_i \le w_S, \quad \forall S \in \mathcal{C}, \\ & y_i \ge 0, \quad \forall i \in U. \end{aligned} \tag{5}$$

Combinatorially, to each set S of $\mathcal{C}$ we associate a non-negative number x_S and to each element i of U we associate a non-negative number y_i. The primal constraints say that the sum of numbers corresponding to sets containing an element i is at least one. The dual constraints say that the sum of numbers corresponding to elements in a set S is at most the weight w_S of the set.

A feasible solution $\bar{\mathbf{y}}$ for (5) is said to be *maximal* iff we cannot increase any component of $\bar{\mathbf{y}}$ without making it infeasible. The following theorem can be shown in a similar manner as that of Theorem 1.1. We will prove a stronger version of this theorem in the next section.

Theorem 1.2. *Let $\bar{\mathbf{y}}$ be a maximal feasible solution to* (5)*, then the strategy of setting $x_j^A = 1$ whenever $\sum_{i \in S_j} y_i = w_j$ gives an f-approximation to the* WEIGHTED SET COVER *problem, where*

$$f = \max_{i \in U} |\{S \mid S \in \mathcal{C},\ i \in S\}|.$$

Note that, one way to get a maximal dual feasible solution is to solve the dual LP. An optimal solution is certainly maximal.

2 The basic primal-dual method

We can get away with solving the dual LP altogether, which is great because solving the dual LP takes quite a bit of running time. All we wanted was a maximal dual feasible solution. This section presents an approach to get such a solution. (There are other approaches, which we shall not discuss here.) In fact, we will not even need to explicitly compute the maximal dual feasible solution at all. The linear program is really used as an analytical device, guiding our search for a good approximation algorithm.

Consider the dual LP (5) for the WEIGHTED SET COVER problem. Certainly $\mathbf{y} = \mathbf{0}$ is a feasible solution. One way to obtain $\bar{\mathbf{y}}$ is to find an appropriate component y_i of $\mathbf{y}$ and increase it as much as possible to turn one more of the inequalities into equality. When this is no longer possible, we get our maximal feasible solution. The following algorithm implements this idea.

PRIMAL-DUAL BASIC

```
1: $\mathbf{y} \leftarrow 0$
2: $C \leftarrow \emptyset$   // this is like setting $x_S^A = 0, \forall S$.
3: while $C$ is not a cover do
4:    Choose an uncovered element $k$
5:    Increase $y_k$ until $\exists S : \sum_{i \in S} y_i = w_S$   // in other words, the $S$-constraint is "binding"
6:    Add $S$ into $C$   // same as setting $x_S^A = 1$ because its constraint is binding.
7: end while
8: Return $C$   // we will refer to this final solution as $\overline{C}$.
```

The idea of the algorithm is very clear. Obviously $\overline{C}$ is a feasible solultion. For the approximation ratio, notice that from the linear programming view $x_S^A = 1$ only when $\sum_{i \in S} y_i = w_S$ (in this case we say S is *saturated*). Also note that there might be saturated sets S which are not in $\overline{C}$ (i.e., $x_S^A \neq 1$), because we only chose one saturated S to add into C at each iteration. The analysis goes as follows.

$$\text{cost}(\mathbf{x}^A) = \sum_{S \in \overline{C}} w_S x_S^A = \sum_{S \in \overline{C}} \left(\sum_{i \in S} y_i \right) = \sum_{i \in U} |\{S : S \in \overline{C}, i \in S\}| y_i.$$

For any collection C of subsets of U, and any $i \in U$, let

$$g(C, i) = |\{S : S \in C, i \in S\}|. \tag{6}$$

If there exists a number ρ such that $y_i > 0$ implies $g(\overline{C}, i) \leq \rho$, then we have

$$\text{cost}(x^A) = \sum_{i \in U} g(\overline{C}, i) y_i \leq \rho \sum_{i \in U} y_i \leq \rho \cdot \text{OPT}.$$

We just proved the following theorem.

Theorem 2.1. *Let $\overline{C}$ be the collection returned by* PRIMAL-DUAL BASIC. *Let ρ be any number such that, for each $i \in U$, $y_i > 0$ implies $g(\overline{C}, i) \leq \rho$. Then,* PRIMAL-DUAL BASIC *is a ρ-approximation algorithm for* SET COVER.

For the WEIGHTED SET COVER problem, ρ can be chosen to be the maximum number of appearances of an element in the given collection, thus proving Theorem 1.2.

Exercise 1. Consider the following algorithm to compute an approximate solution to the WEIGHTED SET COVER problem:

```
WSC-PRIMAL-DUAL-B
1:  $C \leftarrow \emptyset$
2:  $I \leftarrow U$
3:  while $I \neq \emptyset$ do
4:      Let $S$ be a set in $\mathcal{C}$ whose ratio $w_S/|S|$ is minimum.   // NOTE: the fraction is $\infty$ if $S$ is empty.
5:      $r \leftarrow \frac{w_S}{|S|}$,   $C \leftarrow C \cup \{S\}$,   $I \leftarrow I - S$
6:      for each $T \in \mathcal{C}$ do
7:          $w_T \leftarrow w_T - |T \cap S| r$
8:          $T \leftarrow T - S$
9:      end for
10: end while
11: Return $C$
```

(i) In words, explain what the algorithm does.

(ii) Show that the algorithm is an f-approximation algorithm to the WEIGHTED SET COVER problem, where
$$f = \max_{i \in U} |\{S \mid S \in \mathcal{C},\ i \in S\}|.$$

Exercise 2. Given a universe U and a collection $\mathcal{C}$ of subsets of U. Consider the following integer program.

$$\begin{aligned} \text{min} \quad & \sum_{S \in \mathcal{C}} w_S x_S \\ \text{subject to} \quad & \sum_{S \in \mathcal{C}} |S \cap T| x_S \geq |T|, \quad \forall T \subseteq U, \\ & x_S \in \{0,1\}, \quad \forall S \in \mathcal{C}. \end{aligned} \tag{7}$$

(i) Prove that this program is equivalent to the WEIGHTED SET COVER problem. Equivalence means every optimal solution to WEIGHTED SET COVER corresponds to an optimal solution of (7) and vice versa.

(ii) Write down its LP relaxation and the LP's dual program.

(iii) Design an approximation algorithm using the primal-dual method similar to the one presented in the lecture. Your algorithm should have approximation factor f, as usual.

You should briefly describe the idea and write down the pseudo code. Also, what's the running time of your method?

Exercise 3. Consider the HITTING SET problem, in which we are given a universe set U with weighted elements, and a collection of subsets $T_1, \dots, T_k$ of U. The problem is to find a minimum-weight subset A of U which hits every T_i, namely $A \cap T_i \neq \emptyset$, for all $i \in [k]$.

- Write an integer program formulation of this problem. Also write the dual linear program of the relaxed version of the IP.
- Devise a primal-dual approximation algorithm for this problem.
- What's your approximation ratio and running time?

3 Optimization problems formulated as SET COVER

There are quite a few optimization problems that can be formulated as a specical case of the WEIGHTED SET COVER problem. In the following problems, we assume all weights are non-negative integers.

Consider the SHORTEST s-t PATH PROBLEM, in which we need to find a shortest path between two vertices s and t of a graph $G = (V, E)$. Let the universe U be the set of all s-t cuts. For each edge $(u, v) \in E$, let S_{uv} be the set of all cuts in U that contain (u, v). Let $\mathcal{C}$ be the collection of all $S_{u,v}$. If E is weighted, then the weight of $S_{u,v}$ is the weight of (u, v). Then, by the max-flow min-cut theorem it is easy to see that SHORTEST s-t PATH is equivalent to the WEIGHTED SET COVER on U and $\mathcal{C}$. The basic idea is to pick a minimum-weight set of edges that "covers" all s-t cuts!

Similarly, in the MINIMUM SPANNING TREE problem we need a minimum-weight set of edges that cover all cuts in a graph G.

The GENERALIZED STEINER TREE problem is defined as follows. Given an edge-weighted undirected graph $G = (V, E)$ and m pairs of vertices (s_j, t_j), $j \in [m]$. Find a minimum-weight subset of edges $C \subseteq E$ such that s_j and t_j are connected in (V, C), for each $j \in [m]$. In this problem, we want C to cover all s_j-t_j cuts.

In the FEEDBACK VERTEX SET problem, we are given an undirected graph $G = (V, E)$ with weighted vertices. The goal is to find a minimum-weight subset C of vertices so that every cycle in G contains some vertex in C. Thus, we want C to cover all cycles of G.

The MINIMUM-COST ARBORESCENCE problem, also called the MINIMUM-COST BRANCHING PROBLEM is defined as follows. Given a directed, edge-weighted graph $G = (V, E)$ with a special vertex r called the root. Find a minimum-cost spanning tree where edges are directed away from r. In this case, we want the edges of the tree to cover all r-directed cuts.

Many of the problems above can be formulated with the following integer program. We assume that an edge-weighted graph $G = (V, E)$ is given, and edge e is weighted with $w_e \in \mathbb{Z}^+$.

$$\begin{aligned} \min \quad & \sum_{e \in E} w_e x_e \\ \text{subject to} \quad & \sum_{e \in \delta(X)} x_e \geq f(X), \quad \emptyset \neq X \subset V, \\ & x_e \in \{0, 1\}, \qquad \forall e \in E. \end{aligned} \tag{8}$$

Here $\delta(X) = [X, \overline{X}]$, and $f : 2^V \to \mathbb{Z}^+$ is a function that counts how many edges must cross $\delta(X)$ in a feasible solution. In the SHORTEST s-t PATH problem, for instance, each s-t cut must be crossed at least once. Other problems follow the same trend, except for the FEEDBACK VERTEX SET problem.

Exercise 4 (T-join). The T-JOIN problem is defined as follows. Given a subset T of an even number of vertices of an edge-weighted graph G, find a minimum cost forest of G such that all vertices in T have odd degrees and the rest of the vertices have even degrees in the forest.

Show that formulation (8) can be used to capture this problem by defining what $f(X)$ is, and argue why the formulation is correct.

Exercise 5 (Steiner tree). In the STEINER TREE problem, we are given a subset T of vertices of an edge-weighted graph G. We need to find a minimum-cost connected subgraph of G that contains all vertices in T (and perhaps some other vertices).

Show that formulation (8) can be used to capture this problem by defining what $f(X)$ is, and argue why the formulation is correct.

A very general problem of this type is called the SURVIVABLE NETWORK DESIGN PROBLEM, also called the GENERALIZED STEINER PROBLEM. We are given an edge-weighted graph $G = (V, E)$. For each pair u, v of vertices, there is a non-negative integer m_{uv}. We must find a least-cost subset C of edges such that in (V, C) there are m_{uv} edge disjoint paths joining u and v, for each pair u, v. In this case, we want $f(X) = \max\limits_{u \in X, v \notin X} m_{uv}$ for each subset X of vertices.

We shall see how the primal-dual method helps design approximation algorithms for problems formulated as (8) for several classes of functions f.

4 Tuning PRIMAL-DUAL BASIC

4.1 Be particular in choosing the uncovered element k

Consider the FEEDBACK VERTEX SET problem on a graph $G = (V, E)$. We want to cover the universe of cycles of G with vertices in V. For a collection of vertices $\overline{C}$ and some cycle i, $g(\overline{C}, i)$ is the number of vertices of $\overline{C}$ in the cycle. The rough bound is n, because a cycle may contain as many as n vertices. This is a very bad approximation ratio.

Fortunately, to obtain a ratio of ρ we only need $g(\overline{C}, i) \leq \rho$ when $y_i > 0$, i.e. when cycle i is chosen at some point during the execution of PRIMAL-DUAL BASIC. Thus, if we try our best to pick small cycles, we would have a better approximation. Further more, not all vertices in a chosen cycle will be in $\overline{C}$ at the end. Hence, $g(\overline{C}, i)$ can still be smaller than the length of cycle i.

To this end, one needs to be creative to take advantage of the above observations. We will limit the set of vertices which have the potential to be in the final cover. We shall refer to them as *interesting* vertices. If we never pick a uninteresting vertex to be in the cover, then $g(\overline{C}, i)$ is upper-bounded by the number of interesting vertices on cycle i, which can potentially be smaller than the size of this cycle.

All vertices of degree 1 are not interesting. If there was a path P of G, all of whose internal vertices have degree 2, then only a least-weight internal vertex v needs to be interesting among all internal vertices, because a cycle containing v will contain all of P. Consequently, if at each step of the algorithm we can somehow choose an uncovered cycle containing a small number of interesting vertices, then we would be able to improve the approximation ratio. A result of Erdős and Pósa in 1962 tells us how to do this.

Theorem 4.1 (Erdős-Pósa [13]). *Let $G' = (V', E')$ be a graph with no degree-1 vertex in which each degree-2 vertex is adjacent to two vertices of degrees at least 3. Then, G' has a cycle of length at most $4 \lg |V'|$. Moreover, this cycle can be found in polynomial time.*

Exercise 6. Consider a graph G with no degree-1 vertex in which each degree-2 vertex is adjacent to two vertices of degrees at least 3. Let H be the graph obtained from G by shortcutting all vertices of degree 2, namely for each vertex $v \in V(G)$ whose only two neighbors is u and w, we remove v and connect u and w with an edge. (Note that H now has only vertices with degree more than 2.)

1. Suppose we build a breadth-first tree starting from some vertex r of H. Prove that by the time we reach depth $\lg |V(H)|$, we will have discovered a cycle of length at most $2 \lg |V(H)|$.

2. Prove Erdős-Poså theorem.

This theorem suggests the following algorithm for FEEDBACK VERTEX SET (FVS).

Algorithm 4.2. FVS-1

```
1: $\mathbf{y} \leftarrow 0$
2: $C \leftarrow \emptyset$
3: Let $G'$ be a graph obtained from $G$ by removing all uninteresting vertices.
4: while $G'$ is not empty do
5:     Choose cycle $k$ in $G'$ of length at most $4 \lg |V(G')|$
6:     // note that this cycle corresponds uniquely to a cycle in the original graph $G$
7:     Increase $y_k$ until there is some saturated vertex $v$
8:     Add $v$ into $C$
9:     Remove $v$ from $G'$ and then all uninteresting vertices from $G'$
10: end while
11: Return $C$ (call it $\overline{C}$)
```

The following theorem is now immediate from the above analysis.

Theorem 4.3. *Algorithm* FVS-1 *is a* $4 \lg n$*-approximation algorithm for* FEEDBACK VERTEX SET.

A 2-approximation for this problem can be obtained with a different integer programming formulation, as we shall see in a later section.

4.2 Refining the final solution with reverse deletion

4.2.1 Motivations for reverse deletion

Consider the s-t SHORTEST PATH problem and how algorithm PRIMAL-DUAL BASIC applies to it. In this case, we want to pick a minimum-weight subset of edges which covers all s-t cuts $\delta(X)$.

Consider a typical iteration of the algorithm with the current set C of edges chosen so far. If s and t is not connected in (V, C), then there will be a number of "uncovered" cuts to choose from. A sensible approach is to choose a minimal cut that contains s, i.e. we chose $\delta(X)$ where X is the set of vertices of the connected component of (V, C) that contains s. It is not difficult to see that this strategy corresponds to Dijkstra's algorithm for the SINGLE SOURCE SHORTEST PATH problem.

Exercise 7. Prove that the strategy above corresponds to Dijkstra's algorithm.

Unfortunately, this algorithm produces redundant edges, which are in a shortest path tree rooted at s. Hence, it makes sense to remove redundant edges from the final solution $\overline{C}$. For some problems, it is better to remove redundant elements in the reverse order of their addition. This step is called the *reverse deletion step* and is illustrated in the following refinement of the basic algorithm.

Algorithm 4.4. PRIMAL-DUAL WITH REVERSE DELETION

```
1: $\mathbf{y} \leftarrow 0$,  $C \leftarrow \emptyset$,  $j \leftarrow 0$
2: while $C$ is not a cover do
3:     $j \leftarrow j + 1$
4:     $k \leftarrow$ UNCOVERED-ELEMENT($C$)  // we can adapt this procedure for different problems
5:     Increase $y_k$ until some $S$ is saturated, denote $S$ by $S_j$
6:     Add $S_j$ into $C$
7: end while
8: $\overline{C} \leftarrow$ REVERSE DELETE($C$)
```

REVERSE DELETE(C)

```
1: for $j = |C|$ downto $1$ do
2:     if $C - \{S_j\}$ is feasible then
3:         remove $S_j$ from $C$
4:     end if
5: end for
6: Return $C$
```

Fix a k for which $y_k > 0$. As before we would like to estimate an upper bound for $g(\overline{C}, k)$. The situation becomes a little bit more complicated because of the reverse deletion. For notational conveniences, let $k(C) =$ UNCOVERED-ELEMENT(C), where we assume a deterministic strategy of choosing a k given a C.

At the point where we are about to increase y_k, the current solution is $C = \{S_1, \dots, S_{j-1}\}$ for some j and $k = k(C)$ is not in any of these sets. Thus,

$$g(\overline{C}, k) = g\big(\overline{C} \cup C, k(C)\big).$$

The collection $A = \overline{C} \cup C$ is a *minimal augmentation* of C in the sense that A is feasible, and removing any member from $A - C$ will result in an infeasible solution. This follows from the reverse deletion step. The following theorem follows readily.

Theorem 4.5. *If for any iteration of algorithm* PRIMAL-DUAL WITH REVERSE DELETION *with its infeasible solution C,*

$$\max_{A\,:\,\text{min. aug. of } C} g(A, k(C)) \le \rho$$

then the algorithm has approximation ratio ρ.

Exercise 8. Suppose we apply PRIMAL-DUAL WITH REVERSE DELETION to the s-t SHORTEST PATH problem, using the rule of picking a minimal cut containing s each time. Show that the $\rho = 1$ satisfies Theorem 4.5. In other words, the algorithm returns an optimal solution.

Exercise 9. In the MINIMUM-COST ARBORESCENCE problem, we are given a directed edge-weighted graph $G = (V, E)$ and a root vertex r. The goal is to find a minimum-cost tree rooted at r all of whose edges are directed away from r.

This problem can be viewed as a special case of WEIGHTED SET COVER in the following sense: for each subset X of $V - \{r\}$, we want the minimum-cost set of edges to cover $\delta^-(X)$, where $\delta^-(X) = \{(u, v) \mid u \notin X, v \in X\}$. (Here, (u, v) covers $\delta^-(X)$ iff $(u, v) \in \delta^-(X)$.)

Suppose we apply PRIMAL-DUAL WITH REVERSE DELETION to this problem.

(a) Consider an infeasible set of edges C in some iteration of the algorithm and the graph $G' = (V, C)$. Show that there is a strongly connected component of G' with vertex set X such that $r \notin X$, and $C \cap \delta^-(X) = \emptyset$.

(b) Describe how to find this component in polynomial time.

(c) Let UNCOVERED-ELEMENT(C) return $\delta^-(X)$ with X found as in (a). Apply Theorem 4.5 to show that this algorithm has approximation ratio 1, i.e. it returns an optimal solution.

4.2.2 An application: MULTICUT IN TREES

In the MULTICUT problem, we are given a graph $G = (V, E)$ with edge capacity function $c : E \to \mathbb{Z}^+$, and m pairs of vertices (s_i, t_i), $i \in [m]$. The pairs are different, but the vertices in different pairs do not have to be distinct. A *multicut* is a set of edges whose removal disconnects s_i and t_i, for all $i \in [m]$. The problem is to find a multicut with minimum capacity.

Exercise 10. Show that MULTICUT is **NP**-hard even when G is a tree by a reduction from VERTEX COVER.

Throughout this section, we assume G is a tree, so that there is a unique path P_i from s_i to t_i in G. This can be viewed as a SET COVER problem in which an edge e covers all s_i, t_i-paths that contain e. We will show that the algorithm PRIMAL-DUAL WITH REVERSE DELETION gives an approximation ratio 2 for this problem.

The LP-relaxation of the IP for this problem is

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{subject to} \quad & \sum_{e \in P_i} x_e \geq 1 \quad i \in [m], \\ & x_e \geq 0, \quad \forall e \in E. \end{aligned} \tag{9}$$

The dual program is

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} y_i \\ \text{subject to} \quad & \sum_{i:\, e \in P_i} y_i \leq c_e \qquad e \in e, \\ & y_i \geq 0, \quad \forall i \in [m]. \end{aligned} \tag{10}$$

In the context of this problem, we need to be very specific on choosing an uncovered s_k, t_k-path at each iteration. Fix a vertex r of G as the root of the tree. For each pair (s_i, t_i), let LCA(s_i, t_i) denote the *least common ancestor* of s_i and t_i, which is the vertex at the intersection of the paths from s_i to r and from t_i to r. Note that LCA(s_i, t_i) could be s_i or t_i. Let $d(s_i, t_i)$ denote the *depth* of LCA(s_i, t_i), which is the distance between r and LCA(s_i, t_i). Our primal-dual based algorithm is as follows.

Algorithm 4.6. PRIMAL-DUAL FOR MULTICUT IN TREES

```
1: $C \leftarrow \emptyset$, $y \leftarrow 0$, $j \leftarrow 0$
2: **while** $C$ is not a multicut **do**
3:     Choose an uncovered path $P_k$ with largest $d(s_k, t_k)$
4:     Increase $y_k$ until there is a saturated edge $e_j$
5:     Add $e_j$ into $C$
6: **end while**
7: $\overline{C} \leftarrow$ REVERSE-DELETE$(C)$
```

Theorem 4.7. *The algorithm above gives approximation ratio* 2*.*

Proof. Let C be an infeasible set of edges. Let A be a minimal augmentation of C. Let P_k be the uncovered path returned by the algorithm in line 3. We only have to show that there are at most 2 edges of P_k on A.

Let $v =$ LCA(s_k, t_k). Suppose there are at least 3 edges of A on P_k. These edges are not in C. Without loss of generality, assume there are two edges e_1 and e_2 of A on the part of P_k from s_k to v (otherwise, there are at least two edges on the part from t_k to v). Suppose e_1 is closer to v than e_2. Since A is minimal, there is a path P_i such that $e_2 \in P_i$ but $e_1 \notin P_i$. Consequently, $d(s_i, t_i) > d(s_k, t_k)$, which means P_i should have been chosen instead of P_k. This is a contradiction. □

Note that the integer version of (10) is the MAXIMUM INTEGER MULTI-COMMODITY FLOW problem in trees. The algorithm above implicitly gives a feasible solution for this problem (defined by the y_k). This feasible solution is at least half of the optimal cost of the primal IP, hence it is also a 2 approximation for the multi-commodity flow problem.

Open Problem 1. No non-trivial approximation algorithm is known for the MAXIMUM INTEGER MULTI-COMMODITY FLOW problem on graphs more general than trees. (Trivial ones give factor $\Omega(n)$.)

Exercise 11. Suppose instead of doing REVERSE-DELETE (which deletes edges in the reverse order of which they were added into C), we apply the following procedure: sort edges in the final C by decreasing capacity and remove redundant edges in this order. What factor can you prove for the modified algorithm?

Exercise 12. Give a polynomial time algorithm to compute a maximum integer multi-commodity flow on trees with unit edge-capacities. You can use as a subroutine a maximum matching algorithm. (**Hint**: dynamic programming.)

4.2.3 Another application: better approximation for FEEDBACK VERTEX SET

This application illustrates several very important points about the primal-dual method, and about LP-based approximation algorithms in general. Firstly, the right LP/DLP formulation is key to having a good approximation ratio. Secondly, the primal-dual method applies just as well to multicover type of problems.

We shall derive an interesting IP formulation for the FEEDBACK VERTEX SET problem. For any subset S of vertices of the graph G under consideration, let $d_S(v)$ be the degree of vertex v in the subgragh $G[S] = (S, E[S])$ of G induced by S.

Exercise 13. Define $b(S) := |E[S]| - |S| + 1$. Show that, for any subset S of V, and any feedback vertex set F of G, we have

$$\sum_{v \in F} (d_S(v) - 1) \geq b(S).$$

Exercise 14. Consider the following integer program

$$\begin{aligned} \min \quad & \sum_{v \in V} w_v x_v \\ \text{subject to} \quad & \sum_{v \in S} (d_S(v) - 1) x_v \geq b(S) \quad S \subseteq V, \\ & x_v \in \{0, 1\}, \quad \forall v \in V. \end{aligned} \tag{11}$$

Prove that $\mathbf{x}$ is a feasible solution to (11) iff $\mathbf{x}$ is the characteristic vector of a feedback vertex set. In other words, the above IP is equivalent to the FVS problem.

In light for the previous exercise, we look at the LP-relaxation of (11)

$$\begin{aligned} \min \quad & \sum_{v \in V} w_v x_v \\ \text{subject to} \quad & \sum_{v \in S} (d_S(v) - 1) x_v \geq b(S) \quad S \subseteq V, \\ & x_v \geq 0, \quad \forall v \in V. \end{aligned} \tag{12}$$

$$\begin{aligned} \max \quad & \sum_{v \in V} b(S) y_S \\ \text{subject to} \quad & \sum_{S \ni v} (d_S(v) - 1) y_S \leq w_v \quad S \subseteq V, \\ & y_S \geq 0, \quad \forall S \subseteq V. \end{aligned} \tag{13}$$

A *semi-disjoint cycle* of a graph G is a cycle of G with has at most one vertex of degree more than 2. We apply the same idea as that of Algorithm 4.4. Basically, let F be the feedback vertex set constructed so far, and V' be the rest of the vertices. If $G[V']$ has a semi-disjoint cycle C, then we increase the dual variable corresponding to $V(C)$. Otherwise, we increase the dual variable corresponding to V'. We add a saturated vertex to F at each step. When the loop is finished, reverse deletion is applied as before. Formally, the algorithm is described as follows.

Algorithm 4.8. FSV-2

```
1: $F \leftarrow \emptyset,\ \ y \leftarrow 0,\ \ j \leftarrow 0$
2: $V' \leftarrow V,\ \ E' \leftarrow E$
3: **while** $F$ is not feasible **do**
4:    $j \leftarrow j+1$
5:    Recursively remove degree-one vertices and edges from $V'$ and $E'$
6:    **if** $(V', E')$ contains a semi-disjoint cycle $C$ **then**
7:       $S \leftarrow C$
8:    **else**
9:       $S \leftarrow V'$
10:   **end if**
11:   Increase $y_S$ until there is a saturated $v_j \in S$
12:   $F \leftarrow F \cup \{v_j\}$
13:   Remove $v_j$ from $(V', E')$
14: **end while**
15: **for** $i \leftarrow j$ **downto** $1$ **do**
16:   **if** $F - \{v_j\}$ is feasible **then**
17:      $F \leftarrow F - \{v_j\}$
18:   **end if**
19: **end for**
20: **Return** $F$ (refer to it as $\overline{F}$)
```

Because the primal-dual pair (12) and (13) do not have the exact format as those in (4) and (5), Theorem 4.5 does not apply directly. Algorithm 4.8 can be analyzed in a similar manner, as follows.

$$\text{cost}(\overline{F}) = \sum_{v \in \overline{F}} w_v = \sum_{v \in \overline{F}} \sum_{S : v \in S} (d_S(v) - 1) y_S = \sum_{S \subseteq V} \left(\sum_{v \in S} (d_S(v) - 1) \right) y_S.$$

We want to bound

$$\sum_{S \subseteq V} \left(\sum_{v \in S} (d_S(v) - 1) \right) y_S \leq \rho \cdot \sum_{S \subseteq V} b(S) y_S,$$

for some ratio ρ. This will hold true if

$$y_S > 0 \ \text{ implies } \ \sum_{v \in S} (d_S(v) - 1) \leq \rho \cdot b(S).$$

Exercise 15. Consider any S for which $y_S > 0$. Show that $S \cap \overline{F}$ is a minimal feedback vertex set of $G[S]$. Also, show that, if S is a semi-disjoint cycle, then

$$\sum_{v \in S} (d_S(v) - 1) \leq 2 \cdot b(S).$$

Exercise 16. Show that, for any minimal feedback vertex set F of a graph $G = (V, E)$ which contains no semi-disjoint cycle,

$$\sum_{v \in S} (d_S(v) - 1) \leq 2 \cdot b(S).$$

The previous two exercises finished our analysis of Algorithm 4.8.

Theorem 4.9. *Algorithm 4.8 is a* 2*-approximation algorithm.*

Exercise 17. (TBD) Primal dual method for multi-cover problem.

4.3 Increasing simultaneously multiple dual variables

A MINIMUM SPANNING TREE (MST) instance can be viewed as a WEIGHTED SET COVER instance where the edges need to cover all non-trivial cuts in the graph. Prim's algorithm for MST can be thought of as a special case of PRIMAL-DUAL BASIC. However, Kruskal's algorithm is different. Kruskal's algorithm corresponds to increasing all dual variables simultaneously at the same rate until some edge becomes saturated. This idea is summarized in the following more general form of the primal dual method.

GENERAL PRIMAL-DUAL

```
1: $\mathbf{y} \leftarrow 0, \quad C \leftarrow \emptyset, \quad j \leftarrow 0$
2: while $C$ is not a cover do
3:     $j \leftarrow j + 1$
4:     $\nu_j \leftarrow$ UNCOVERED-ELEMENTS($C$)   // pick a subset of uncovered elements
5:     For all $k \in \nu_j$, increase all $y_k$ at the same rate, until some $S$ is saturated.
6:     Refer to $S$ as $S_j$ and add it into $C$
7: end while
8: $\overline{C} \leftarrow$ REVERSE DELETE($C$)
```

Let l be the total number of iterations. Let ϵ_j be the amount by which each variable in ν_j was increased. In the end we have

$$\sum_{k \in U} y_k = \sum_{j=1}^{l} |\nu_j| \epsilon_j.$$

Following our usual line of analysis:

$$\text{cost}(\mathbf{x}^A) = \sum_{i \in U} g(\overline{C}, i) y_i = \sum_{i \in U} g(\overline{C}, i) \sum_{j : \nu_j \ni i} \epsilon_j = \sum_{j=1}^{l} \left(\sum_{i \in \nu_j} g(\overline{C}, i) \right) \epsilon_j$$

Let $\nu(C)$ denote UNCOVERED-ELEMENTS(C). The following theorem follows naturally. The Theorem is a generalization of Theorem 4.5.

Theorem 4.10. *If for any iteration* j *of algorithm* GENERAL PRIMAL-DUAL *with infeasible solution* C,

$$\max_{A \,:\, \text{min. aug. of } C} \sum_{i \in \nu(C)} g(A, i) \leq \rho |\nu(C)|$$

then the algorithm has approximation ratio ρ.

We shall apply this algorithm to get a 2-approximation for the GENERALIZED STEINER TREE problem. Recall that we have an edge-weighted graph $G = (V, E)$ and m pairs of vertices (s_j, t_j) and we need to find a minimum-cost set of edges covering all s_j-t_j cuts. In this algorithm, $\nu(C) =$ UNCOVERED-ELEMENTS(C) is the set of all cuts $\delta(X)$ where X is a connected component of (V, C) for which $|X \cap \{s_j, t_j\}| = 1$ for some j.

Theorem 4.11. *The algorithm for* GENERALIZED STEINER TREE *as described above has approximation ratio* 2*.*

Proof. Consider an infeasible solution C. The graph (V, C) has several connected components. If A is a minimal augmentation of C, then the graph (V, A) is a forest if we view the connected components of (V, C) as vertices. Let T denote this forest.

The forest T has two types of vertices: the *red* vertices correspond to the connected components X where $\delta(X) \in \nu(C)$, and the rest are *blue* vertices. Let R denote the set of red vertices and B the set of blue vertices with positive degrees. Noting that $|\nu(C)| = |R|$, we have

$$\begin{aligned}
\sum_{i\in\nu(C)} g(A,i) &= \sum_{v\in R} \deg_T(v) \\
&= 2|E(T)| - \sum_{v\in B} \deg_T(v) \\
&\leq 2(|R| + |B|) - \sum_{v\in B} \deg_T(v) \\
&\leq 2(|R| + |B|) - 2|B| \\
&= 2|\nu(C)|.
\end{aligned}$$

The last inequality follows because no blue vertex has degree one, otherwise A is not a minimal augmentation of C. □

Exercise 18. Many of the problems we have discussed can be formulated with the following integer program. We assume that an edge-weighted graph $G = (V, E)$ is given, and edge e is weighted with $w_e \in \mathbb{Z}^+$.

$$\begin{aligned}
\min \quad & \sum_{e\in E} w_e x_e \\
\text{subject to} \quad & \sum_{e\in\delta(X)} x_e \geq f(X), \quad \emptyset \neq X \subset V, \\
& x_e \in \{0,1\}, \qquad \forall e \in E.
\end{aligned} \tag{14}$$

Here $\delta(X) = [X, \overline{X}]$, and $f : 2^V \to \mathbb{Z}^+$ is a function that counts how many edges must cross $\delta(X)$ in a feasible solution.

The dual of the LP relaxation of the above program is

$$\begin{aligned}
\max \quad & \sum_{\emptyset\neq X\subset V} f(X) y_X \\
\text{subject to} \quad & \sum_{X: e\in\delta(X)} y_X \leq w_e, \qquad e \in E, \\
& y_X \geq 0, \quad \forall X, \emptyset \neq X \subset V.
\end{aligned} \tag{15}$$

In this problem, we shall develop an approximation algorithm for this general setting where f is a special class of function. To solve this problem, you must understand thoroughly the algorithm we developed for the GENERALIZED STEINER TREE problem, which will be a special case of this problem.

We assume that f has the following properties:

- $f(X) \in \{0,1\}$, for all $X \subseteq V$. In other words, f is a 01-function. This problem is thus a special case of our WEIGHTED SET COVER problem in which each cut $\delta(X)$ with $f(X) = 1$ has to be covered.
- $f(V) = 0$. This is natural since $\delta(V) = \emptyset$.

- $f(X) = f(\overline{X})$ for all subsets X of V. This is also natural, since $\delta(X) = \delta(\overline{X})$ in an undirected graph.
- If X and Y are two disjoint subsets of V, then $f(X) = f(Y) = 0$ implies $f(X \cup Y) = 0$. This means if $\delta(X)$ and $\delta(Y)$ do not have to be covered, then so does $\delta(X \cup Y)$.

A function f satisfying the above properties is called a 01-*proper function*.

1. Let C be an infeasible subset of edges of G (with respect to f, of course). Prove that there is some connected component X of (V, C) for which $f(X) = 1$. (Here, we use X to also denote the set of vertices of the connected component X.)
2. Let C be an infeasible subset of edges of G. Let X be a connected component of (V, C). Let Y be a subset of vertices such that $Y \cap X \neq \emptyset$ and $X \not\subseteq Y$. Prove that C covers $\delta(Y)$.

   (Note: this means that we only have to worry about covering the $\delta(Y)$ for which Y contains one or a few connected components of (V, C).)
3. Consider the following algorithm for our problem.

   Algorithm 4.12. PRIMAL-DUAL FOR 01-PROPER FUNCTION
   ```
   1: $\mathbf{y} \leftarrow 0$; $C \leftarrow \emptyset$; $j \leftarrow 0$
   2: while $C$ is infeasible do
   3:     $j \leftarrow j + 1$
   4:     Let $\nu_j$ be the set of all $X$ which is a connected component of $(V, C)$ and $f(X) = 1$
   5:     Increase all $y_X$ at the same rate, $X \in \nu_j$, until $\exists e : \sum_{Z : e \in \delta(Z)} y_Z = w_e$
   6:     Refer to $e$ as $e_j$ and add it into $C$
   7: end while
   8: $\overline{C} \leftarrow$ REVERSE DELETE($C$)
   ```

   Prove that this is a 2-approximation algorithm for our problem.

5 Advanced applications of the primal-dual method

5.1 Metric uncapacitated facility location

FACILITY LOCATION is a fundamental optimization problem appearing in various context. In the uncapacitated version of the problem, we are given a complete bipartite graph $G = (F, C; E)$ where F represents a set of "facilities" and C a set of "cities." The cost of opening facility i is f_i, and the cost of assigning city j to facility i is c_{ij}. The problem is to find a subset $I \subseteq F$ of facilities to be open and an assignment $a : C \to I$ assigning every city j to some facility $a(j)$ to minimize the cost function

$$\sum_{i \in I} f_i + \sum_{j \in C} c_{a(j),j}.$$

In the metric version of the problem, the cost c_{ij} satisfies the triangle inequality.

Designate a variable x_i indicating if facility i is open and y_{ij} indicating if city j is assigned to facility i, we get the following integer program:

$$\begin{array}{lll} \min & \sum_{i \in F} f_i x_i + \sum_{i \in F, j \in C} c_{ij} y_{ij} & \\ \text{subject to} & \sum_{i \in F} y_{ij} \geq 1 & j \in C, \\ & x_i - y_{ij} \geq 0 & i \in F,\ j \in C, \\ & x_i, y_{ij} \in \{0, 1\}, & i \in F,\ j \in C. \end{array} \tag{16}$$

Relaxing this integer program gives the following linear program

$$\begin{aligned} \min \quad & \sum_{i\in F} f_i x_i + \sum_{i\in F, j\in C} c_{ij} y_{ij} \\ \text{subject to} \quad & \sum_{i\in F} y_{ij} \geq 1 && j \in C, \\ & x_i - y_{ij} \geq 0 && i \in F,\ j \in C, \\ & x_i, y_{ij} \geq 0, && i \in F,\ j \in C. \end{aligned} \tag{17}$$

The dual linear program is

$$\begin{aligned} \max \quad & \sum_{j\in C} s_j \\ \text{subject to} \quad & \sum_{j\in C} t_{ij} \leq f_i && i \in F, \\ & s_j - t_{ij} \leq c_{ij} && i \in F,\ j \in C, \\ & s_j, t_{ij} \geq 0, && i \in F,\ j \in C. \end{aligned} \tag{18}$$

This primal-dual pair does not fit nicely into the covering primal-dual approximation framework we discussed. In particular, there are negative coefficients. The general idea of applying the primal-dual method to this problem is still to find some sort of "maximal" dual-feasible solution, and then set to 1 the primal variables corresponding to saturated primal constraints.

PRIMAL-DUAL FOR METRIC UNCAPACITATED FACILITY LOCATION

Phase 1.

```
1:  $O \leftarrow \emptyset$;  // set of temporarily open facilities
2:  $J \leftarrow \emptyset$;  // set of connected cities thus far
3:  $\mathbf{s} \leftarrow 0$; $\mathbf{t} \leftarrow 0$
4:  while $J \neq C$ do
5:      Increase uniformly all $s_j$, $j \in C - J$
6:      After a while, if for some edge $ij$, we reach $s_j - t_{ij} = c_{ij}$, then increase uniformly $t_{ij}$ also.
7:      // Edges with $s_j - t_{ij} = c_{ij}$ are called "tight"   Edges with $t_{ij} > 0$ are called "special"
8:      As soon as an edge $ij$ becomes tight, if $i \in O$ then add $j$ into $J$ and declare $i$ the "connection
        witness" for $j$
9:      After a while, there is some $i$ such that $f_i = \sum_{j\in C} t_{ij}$.
10:     for each such $i$ in any order do
11:         $O \leftarrow O \cup \{i\}$
12:         for each tight edge $ij$ with $j \notin J$ do
13:             $J \leftarrow J \cup \{j\}$   // $i$ is called a "connection witness" for $j$
14:         end for
15:     end for
16: end while
```

Phase 2.

```
1:  Let $H = (O, C)$ be the bipartite graph containing only special edges
2:  Let $I$ be a maximal subset of $O$ such that there is no path of length 2 in $H$ between any two vertices
    in $I$
3:  for each $j \in C$ do
4:      if $\exists i \in I$ such that $ij$ is special then
5:          $a(j) \leftarrow i$   // call $j$ "directly connected" to $i$
```

```
6:    else
7:       Let $i$ be the connection witness for $j$
8:       if $i \in I$ then
9:          $a(j) \leftarrow i$   call $j$ “directly connected” to $i$  // note that $ij$ is tight but not special
10:      else
11:         There must be some $i' \in I$ within $H$-distance 2 from $i$
12:         $a(j) \leftarrow i'$   call $j$ “indirectly connected” to $i'$
13:      end if
14:   end if
15: end for
```

We shall use $(\bar{\mathbf{s}}, \bar{\mathbf{t}})$ to denote the returned dual-feasible solution $(\mathbf{s}, \mathbf{t})$.

Theorem 5.1. *The algorithm above gives approximation ratio* 3.

Proof. The idea is to compare the cost of the approximated solution

$$\sum_{i \in I} f_i + \sum_{j \in C} c_{a(j),j}$$

to the cost of the dual-feasible solution $(\bar{s}, \bar{t})$, which is $\sum_{j \in C} \bar{s}_j$.

We shall break each $\bar{s}_j$ into two parts, and write $\bar{s}_j = \bar{s}_j^f + \bar{s}_j^c$ in the following way. If j is directly connected to $i = a(j)$, then set $\bar{s}_j^f = \bar{t}_{ij}$ and $\bar{s}_j^c = c_{ij}$. If j is indirectly connected to $i = a(j)$, then set $\bar{s}_j^f = 0$ and $\bar{s}_j^c = \bar{s}_j$. Intuitively, the term $\bar{s}_j^f$ is the contribution of j into opening facility i, and the term $\bar{s}_j^c$ is the contribution of j to the cost of having edge ij.

Firstly, if $i \in I$ and ij is special, then j is directly connected to i. Consequently

$$\sum_{i \in I} f_i = \sum_{i \in I} \sum_{j : ij \text{ special}} \bar{t}_{ij} = \sum_{j \in C} \bar{s}_j^f.$$

Secondly, we claim that $c_{ij} \leq 3\bar{s}_j^c$, where $i = a(j)$. If j is directly connected to i, then $c_{ij} = \bar{s}_j^c$ by definition. When j is indirectly connected to i, there is an $i' \in I$, $j' \in C$ such that ij' and $i'j'$ are special, and that i' is a connection witness for j. By the triangle inequality, it is sufficient to prove that all three of $c_{i'j}$, $c_{i'j'}$, and $c_{ij'}$ are at most $\bar{s}_j$.

Since i' is a connection witness for j, the edge $i'j$ is tight, implying $c_{i'j} \leq \bar{s}_j$. If we can show that $s_{j'} \leq s_j$, then the other two inequalities follow. Since i' is a connection witness for j, s_j got increased until right at or after the time i' became temporarily open. Since both $i'j'$ and ij' are special, $s_{j'}$ could not have gotten increased after the time i' became temporarily open. We thus have $s_{j'} \leq s_j$.

Consequently,

$$\sum_{i \in I} f_i + \sum_{j \in C} c_{a(j),j} \leq \sum_{j \in C} \left(\bar{s}_j^f + 3\bar{s}_j^c \right) \leq 3\textsc{opt}.$$

$\square$

Exercise 19. The vector $\bar{\mathbf{s}}$ found by the algorithm above is maximal in the sense that, if we increase any $\bar{s}_j$ and keep other $\bar{s}_j$ the same, then there is no way to adjust the $\bar{t}_{ij}$ so that $(\bar{\mathbf{s}}, \bar{\mathbf{t}})$ is still dual-feasible. Is every maximal solution $\bar{\mathbf{s}}$ within 3 times the optimal solution to the dual linear program?

Hint: consider n facilities with opening cost of 1 each, n cities connected to distinct facilities with cost ϵ each. In addition, there is another city that is connected to each facility with an edge of cost 1.

Exercise 20. Suppose the cost of connecting city i to facility j is c_{ij}^2, where the costs c_{ij} still satisfy the triangle inequality (but their squares may not). Show that our algorithm gives performance ratio 9.

Exercise 21. Suppose we slightly change the problem in the following way. Each city j has a demand d_j. The cost of connecting j to an open facility i is now $c_{ij}d_j$. (Previously, all d_j are 1.) Modify our algorithm to get a 3-approximation for this problem. (**Hint**: raise s_j at rate d_j.)

5.2 Metric k-median

The k-MEDIAN problem is very similar to the FACILITY LOCATION problem. The difference is that there is no cost for opening facilities. On the other hand, there is an upper bound of k on the number of open facilities.

Keeping x_i and y_{ij} as in the previous section, we can obtain an integer program for the k-MEDIAN problem. Its LP-relaxation is as follows.

$$\begin{array}{rlr} \min & \displaystyle\sum_{i\in F, j\in C} c_{ij}y_{ij} & \\ \text{subject to} & \displaystyle\sum_{i\in F} y_{ij} \geq 1 & j \in C, \\ & x_i - y_{ij} \geq 0 & i \in F,\ j\in C, \\ & \displaystyle\sum_{i\in F}(-x_i) \geq -k & \\ & x_i, y_{ij} \geq 0, & i\in F,\ j \in C. \end{array} \tag{19}$$

The dual linear program is

$$\begin{array}{rlr} \max & \displaystyle\sum_{j\in F} s_j - ku & \\ \text{subject to} & \displaystyle\sum_{j\in C} t_{ij} \leq u & i \in F, \\ & s_j - t_{ij} \leq c_{ij} & i \in F,\ j\in C, \\ & s_j, t_{ij}, u \geq 0, & i\in F,\ j \in C. \end{array} \tag{20}$$

This primal-dual pair looks strikingly similar to the primal-dual pair of the FACILITY LOCATION problem. In fact, if we assign a cost of u to each facility in the FACILITY LOCATION problem and solve for the primal optimal solution $(\mathbf{x}, \mathbf{y})$ and dual optimal solution $(\mathbf{s}, \mathbf{t})$, then by strong duality

$$\sum_{i\in F} ux_i + \sum_{i\in F, j\in C} c_{ij}y_{ij} = \sum_{j\in F} s_j.$$

Consequently, if there is a value of u such that the primal optimal solution $(\mathbf{x}, \mathbf{y})$ opens exactly k facilities (fractionally), i.e. $\sum_{i\in F} x_i = k$, then it is clear that $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{s}, \mathbf{t}, u)$ are optimal solutions to the primal-dual pair of the k-MEDIAN problem.

On the same line of thought, suppose we can find a value of u for which the approximation algorithm for FACILITY LOCATION returns an integral solution $(\mathbf{x}, \mathbf{y})$ and a dual-feasible solution $(\mathbf{s}, \mathbf{t})$ such that exactly k facilities are open, then

$$\begin{aligned} 3\sum_{i\in F} ux_i + \sum_{i\in F, j\in C} c_{ij}y_{ij} &= 3\sum_{i\in I} f_i + \sum_{j\in C} c_{a(j),j} \\ &\leq 3\sum_{j\in C}(s_j^f + s_j^c) \\ &= 3\sum_{j\in C} s_j. \end{aligned}$$

This implies

$$\sum_{i\in F, j\in C} c_{ij}y_{ij} = \sum_{j\in C} c_{a(j),j} \leq 3\left(\sum_{j\in C} s_j - ku\right),$$

and we would have gotten a 3-approximation algorithm for the k-MEDIAN problem. Unfortunately, it is an open problem to find a u so that this happens. We will take a different path.

Let n_c be the number of cities, n_f number of facilities, $n = n_c + n_f$, and $m = n_c n_f$ the number of edges in the graph.

In the algorithm for FACILITY LOCATION, the larger u is the fewer number of facilities will be opened. (More edges will become tight before the cost u is reached.) When $u = 0$, all facilities will be opened. When $u = n_c c_{\max}$, where $c_{\max}$ is the maximum c_{ij} and n is the number of cities, only one facility will be opened, because all edges are tight when this cost is reached. Assuming we break ties canonically, it is easy to see that the number of opened facilities is inversely proportional to u.

Apply binary search on the interval $[0, n_c c_{\max}]$ to find two values $u_1 < u_2$ such that the corresponding number of opened facilities k_1, k_2 satisfy $k_1 > k > k_2$ and that $u_2 - u_1 \leq c_{\min}/(12n_f^2)$, where $c_{\min}$ is the minimum value of c_{ij}. (If we can find a value of u for which the number of opened facilities is k, then we are done.) Let the corresponding integral primal solutions be $(\mathbf{x}^{(1)}, \mathbf{y}^{(1)})$ and $(\mathbf{x}^{(2)}, \mathbf{y}^{(2)})$, and the corresponding (fractional) dual solutions be $(\mathbf{s}^{(1)}, \mathbf{t}^{(1)})$ and $(\mathbf{s}^{(2)}, \mathbf{t}^{(2)})$, respectively.

First, the idea is to get a convex combination

$$(\mathbf{x}, \mathbf{y}) = \alpha(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}) + \beta(\mathbf{x}^{(2)}, \mathbf{y}^{(2)})$$

such that $(\mathbf{x}, \mathbf{y})$ opens k facilities fractionally. This means $\alpha k_1 + \beta k_2 = k$ and $\alpha + \beta = 1$, which implies $\alpha = \frac{k-k_2}{k_1-k_2}$, and $\beta = \frac{k_1-k}{k_1-k_2}$. Let

$$(\mathbf{s}, \mathbf{t}) = \alpha(\mathbf{s}^{(1)}, \mathbf{t}^{(1)}) + \beta(\mathbf{s}^{(2)}, \mathbf{t}^{(2)}).$$

Note that $(\mathbf{s}, \mathbf{t}, u_2)$ is a feasible solution to the dual program (20).

Let us estimate how good this fractional combination is. We have

$$\sum_{i\in F, j\in C} c_{ij}y_{ij}^{(1)} \leq 3\left(\sum_{j\in C} s_j^{(1)} - k_1u_1\right) \tag{21}$$

$$\sum_{i\in F, j\in C} c_{ij}y_{ij}^{(2)} \leq 3\left(\sum_{j\in C} s_j^{(2)} - k_2u_2\right). \tag{22}$$

We want to turn u_1 in the first inequality into u_2 with a small increase in the factor 3, and then take the convex combination to estimate the cost of $(\mathbf{x}, \mathbf{y})$. Using the facts that $u_2 - u_1 \leq \frac{c_{\min}}{12n_f^2}$, $k_1 \leq n_f$, and $c_{\min} \leq \sum_{i\in F, j\in C} c_{ij}y_{ij}^{(1)}$, we get

$$\sum_{i\in F, j\in C} c_{ij}y_{ij}^{(1)} \leq (3 + 1/n_f)\left(\sum_{j\in C} s_j^{(1)} - k_1u_2\right). \tag{23}$$

Now, an α, β convex combination of (23) and (22) gives

$$\sum_{i\in F, j\in C} c_{ij}y_{ij} \leq (3 + 1/n_f)\left(\sum_{j\in C} s_j - ku_2\right).$$

Thus, the fractional solution $(\mathbf{x}, \mathbf{y})$ is within $(3 + 1/n_f)$ of the optimal. To turn the fractional solution into an integral solution, we apply randomized rounding.

Let I_1 and I_2 be the set of facilities returned by the algorithm corresponding to u_1 and u_2 respectively. We know $|I_1| = k_1$ and $|I_2| = k_2$, and that $k_1 > k > k_2$. The fractions α and β indicate how much the solution should be leaning towards I_1 and I_2. Hence, it is natural to use them as rounding probability. The trouble is that I_2 does not have enough elements, while I_1 has too many elements.

We resolve this problem in the following way. For each i in I_2, let $\nu(i)$ be a facility in I_1 nearest to i. Let I_ν be the set of these $\nu(i)$. Clearly $|I_\nu| \le k_2 < k_1$. We arbitrarily pad I_ν with elements from I_1 until $|I_\nu| = k_2$. Our rounding procedure goes as follows.

- Open all facilities in I_2 with probability β and all facilities in I_ν with probability α.
- Pick uniformly a subset I_3 of $I_1 - I_\nu$ of size $|I_3| = k - k_2$ and open all facilities in I_3. Note that each element in $I_1 - I_\nu$ has a probability of

$$\frac{\binom{k_1-k_2-1}{k-k_2-1}}{\binom{k_1-k_2}{k-k_2}} = \frac{k-k_2}{k_1-k_2} = \alpha$$

  of being chosen.
- Return the set I of k opened facilities.

The next thing to do is to assign cities to these opened facilities. Consider any city j. Let i_1 and i_2 be the facilities that j was connected to in the solutions I_1 and I_2. In the first case, suppose $i_1 \in I_\nu$. Since either i_1 or i_2 is open, we set $a(j)$ to be the open facility. In the second case, suppose $i_1 \notin I_\nu$, in which case we connect j to i_1 if it is open (i.e. $i_1 \in I_3$), otherwise to i_2 if it is open. If both i_1 and i_2 are not open, we connect j to $i_3 = \nu(i_2)$.

We estimate the expected cost of connecting j to $a(j)$. In the first case ($i_1 \in I_\nu$),

$$\mathrm{E}[c_{a(j),j}] = \alpha c_{i_1 j} + \beta c_{i_2 j}.$$

In the second case when $i_1 \notin I_\nu$, there is a probability of α that i_1 is in I_3, a probability of $(1-\alpha)\beta = \beta^2$ that $i_1 \notin I_3$ but i_2 is open, and a probability of $(1-\alpha)(1-\beta) = \alpha\beta$ that j will be connected to i_3. Thus, in this case

$$\mathrm{E}[c_{a(j),j}] = \alpha c_{i_1 j} + \beta^2 c_{i_2 j} + \alpha\beta c_{i_3 j}.$$

By the triangle inequality, we have

$$c_{i_3 j} \le c_{i_3 i_2} + c_{i_2 j} \le c_{i_1 i_2} + c_{i_2 j} \le c_{i_1 j} + 2c_{i_2 j}.$$

Consequently,

$$\mathrm{E}[c_{a(j),j}] \le \alpha(1+\beta)c_{i_1 j} + \beta(1+\alpha)c_{i_2 j} \le (1 + \max\{\alpha, \beta\})[\alpha c_{i_1 j} + \beta c_{i_2 j}].$$

We have just shown the following theorem.

Theorem 5.2. *The above rounding procedure gives expected cost at most* $(1 + \max\{\alpha, \beta\})$ *the cost of* $(\mathbf{x}, \mathbf{y})$.

Thus, in total the rounded solution is of cost at most $(3+1/n_f)(1+\max\{\alpha, \beta\})$ of the optimal. Since $\max\{\alpha, \beta\}$ is at most $n_f/(n_f+1)$, and $(3+1/n_f)(1+n_f/(n_f+1)) \le 6$, we obtain a 6-approximation.

The algorithm can be derandomized with the method of conditional expectation. Note that the expectations $\mathrm{E}[c_{a(j),j}]$ can be calculated explicitly and efficiently. To derandomize this algorithm, we compute the expectations of the final cost given that we open I_2 or I_ν. We then follow the smaller expectation (in the α or β weighted sense). To compute which I_3 to open, we can compute the conditional expectations of the cost given that elements $i_1, \ldots, i_{k_2-k_1}$ of $I_1 - I_2$ are in I_3, then repeat this process.

Exercise 22 (Vazirani's book - Exercise 25.3). Use Lagrangian relaxation technique to give a constant factor approximation algorithm for the following common generalization of the FACILITY LOCATION and k-MEDIAN problems. Consider the UNCAPACITATED FACILITY LOCATION problem with the additional constraint that at most k facilities can be opened.

Exercise 23 (Jain and Vazirani [25]). Consider the l_2^2-CLUSTERING problem. Given a set of n points $S = \{v_1, \dots, v_n\}$ in $\mathbb{R}^d$ and a positive integer k, the problem is to find a minimum cost k-clustering, i.e., to find k points, called *centers,* $f_1, \dots, f_k \in \mathbb{R}^d$, so as to minimize the sum of squares of distances from each point v_i to its closest center. This naturally defines a partitioning of the n points into k clusters. Give a constant factor approximation algorithm for this problem.

(**Hint**: first show that restricting the centers to a subset of S increases the cost of the optimization solution by a factor of at most 2.)

Historical Notes

The primal-dual method was proposed by Dantzig, Ford, and Fulkerson [11] to solve linear programs. This method was motivated by the works of Egerváry [12] and Kuhn [26] on the so-called "Hungarian algorithm" for the assignment problem (or the minimum cost bipartite perfect matching problem). The primal-dual method is not effective as a method for solving linear programs in general. Its strength lies in the fact that it can be used to "transform" a weighted optimization problem into a purely combinatorial and unweighted optimization problem. Many fundamental combinatorial optimization algorithms are either a special case of this method or can be understood in terms of it. For more details, consult standard combinatorial optimization textbooks such as [10, 18, 29–32].

Bar-Yehuda and Even [3] gave the first truly primal-dual algorithm to approximate the VERTEX COVER and the SET COVER problem, as presented in algorithm WSC-PRIMAL-DUAL-A. The LP-algorithms for these problems were due to Hochbaum [21]. The algorithm for the GENERAL COVER problem is by Hall and Hochbaum [20]. Chapter 3 of [23] is a good survey on covering and packing problems.

The survey papers by Goemans and Williamson [16], Williamson [37], Shmoys [33], and many chapters in [23, 36] discuss the primal-dual method for approximation in more details.

The 2-approximation for multicut in trees was due to Garg, Vazirani, and Yannakakis [15]. Recent works on integer multi-commodity flow can be found in [6–8, 19, 35]. For an example of multi-commodity flow in networking, see [28].

For the FEEDBACK VERTEX SET problem, Goemans and Williamson [17] gave a $9/4$-approximation for planar graphs, Becker and Geiger [4] and independently Bafna, Berman, and Fujito [2] gave 2-approximation algorithms, whose primal-dual interpretation was given by Chudak, Goemans, Hochbaum, and Williamson [9]. Later, Fujito [14] generalizes this idea to design primal-dual algorithms for node-deletion problems for *hereditary* graph properties.

For the UNCAPACITATED FACILITY LOCATION problem, Hochbaum [22] obtained ratio $O(\lg n)$, Shmoys, Tardos, and Aardal [34] got 3.16 ratio with an LP-rounding based algorithm. The 3-approximation algorithm we described was due to Jain and Vazirani [25]. Jain, Mahdian and Saberi [24] reduce the ratio further to 1.61 with a greedy algorithm analyzed by the dual-fitting method.

For the METRIC k-MEDIAN problem, Bartal [] gave the first algorithm which achieved approximation ratio $O(\lg n \lg \lg n)$. Charikar, Guha, Tardos, and Shmoys [5] achieved $6\frac{2}{3}$ using ideas from Lin and Vitter [27]. The 6-approximation algorithm we described was due to Jain and Vazirani [25]. Arya, Garg, Khandekar, Meyerson, Munagala, and Pandit [1] achieved ratio $(3 + 2/p)$ with running time $O(n^p)$, for any p using the local search method. Jain, Mahdian, and Saberi [24] gave a hardness ratio of $1 + 2/e$ for approximating this problem.

References

[1] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, *Local search heuristics for k-median and facility location problems*, SIAM J. Comput., 33 (2004), pp. 544–562 (electronic).

[2] V. Bafna, P. Berman, and T. Fujito, *A 2-approximation algorithm for the undirected feedback vertex set problem*, SIAM J. Discrete Math., 12 (1999), pp. 289–297 (electronic).

[3] R. Bar-Yehuda and S. Even, *A linear-time approximation algorithm for the weighted vertex cover problem*, J. Algorithms, 2 (1981), pp. 198–203.

[4] A. Becker and D. Geiger, *Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem*, Artificial Intelligence, 83 (1996), pp. 167–188.

[5] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys, *A constant-factor approximation algorithm for the k-median problem*, J. Comput. System Sci., 65 (2002), pp. 129–149. Special issue on STOC, 1999 (Atlanta, GA).

[6] C. Chekuri and S. Khanna, *Edge disjoint paths revisited*, in Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (Baltimore, MD, 2003), New York, 2003, ACM, pp. 628–637.

[7] C. Chekuri, S. Khanna, and F. B. Shepherd, *The all-or-nothing multicommodity flow problem*, in STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, New York, NY, USA, 2004, ACM Press, pp. 156–165.

[8] C. Chekuri, S. Khanna, and F. B. Shepherd, *Edge-disjoint paths in undirected planar graphs*, in FOCS '04 (Rome, Italy, ACM, New York, 2004, pp. ??–?? (electronic).

[9] F. A. Chudak, M. X. Goemans, D. S. Hochbaum, and D. P. Williamson, *A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs*, Oper. Res. Lett., 22 (1998), pp. 111–118.

[10] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial optimization*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons Inc., New York, 1998. A Wiley-Interscience Publication.

[11] G. B. Dantzig, L. R. Ford, Jr., and D. R. Fulkerson, *A primal-dual algorithm for linear programs*, in Linear inequalities and related systems, Annals of Mathematics Studies, no. 38, Princeton University Press, Princeton, N. J., 1956, pp. 171–181.

[12] J. Egerváry, *Matrixok kombinatorikus tulajdonságairól*, Mathematikai és Fizikai Lápok, 38 (1931), pp. 19–28.

[13] P. Erdős and L. Pósa, *On the maximal number of disjoint circuits of a graph*, Publ. Math. Debrecen, 9 (1962), pp. 3–12.

[14] T. Fujito, *Approximating node-deletion problems for matroidal properties*, J. Algorithms, 31 (1999), pp. 211–227.

[15] N. Garg, V. V. Vazirani, and M. Yannakakis, *Primal-dual approximation algorithms for integral flow and multicut in trees*, Algorithmica, 18 (1997), pp. 3–20.

[16] M. X. Goemans and D. Williamson, *The primal-dual method for approximation algorithms and its application to network design problems*, in Approximation Algorithms for NP-Hard Problems, D. Hochbaum, ed., PWS Publishing Company, 1997, pp. 144–191.

[17] M. X. Goemans and D. P. Williamson, *Primal-dual approximation algorithms for feedback problems in planar graphs*, Combinatorica, 18 (1998), pp. 37–59.

[18] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*, vol. 2 of Algorithms and Combinatorics, Springer-Verlag, Berlin, second ed., 1993.

[19] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis, *Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems*, J. Comput. System Sci., 67 (2003), pp. 473–496.

[20] N. G. Hall and D. S. Hochbaum, *A fast approximation algorithm for the multicovering problem*, Discrete Appl. Math., 15 (1986), pp. 35–40.

[21] D. S. HOCHBAUM, *Approximation algorithms for the set covering and vertex cover problems*, SIAM J. Comput., 11 (1982), pp. 555–556.

[22] ———, *Heuristics for the fixed cost median problem*, Math. Programming, 22 (1982), pp. 148–162.

[23] D. S. HOCHBAUM, ed., *Approximation Algorithms for NP Hard Problems*, PWS Publishing Company, Boston, MA, 1997.

[24] K. JAIN, M. MAHDIAN, AND A. SABERI, *A new greedy approach for facility location problems*, in STOC '02: Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing, ACM Press, 2002, pp. 731–740.

[25] K. JAIN AND V. V. VAZIRANI, *Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation*, J. ACM, 48 (2001), pp. 274–296.

[26] H. W. KUHN, *The Hungarian method for the assignment problem*, Naval Res. Logist. Quart., 2 (1955), pp. 83–97.

[27] J.-H. LIN AND J. S. VITTER, *Approximation algorithms for geometric median problems*, Inform. Process. Lett., 44 (1992), pp. 245–249.

[28] A. E. OZDAGLAR AND D. P. BERTSEKAS, *Optimal solution of integer multicommodity flow problems with application in optical networks*, in Frontiers in global optimization, vol. 74 of Nonconvex Optim. Appl., Kluwer Acad. Publ., Boston, MA, 2004, pp. 411–435.

[29] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial optimization: algorithms and complexity*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1982.

[30] A. SCHRIJVER, *Combinatorial optimization. Polyhedra and efficiency. Vol. A*, vol. 24 of Algorithms and Combinatorics, Springer-Verlag, Berlin, 2003. Paths, flows, matchings, Chapters 1–38.

[31] ———, *Combinatorial optimization. Polyhedra and efficiency. Vol. B*, vol. 24 of Algorithms and Combinatorics, Springer-Verlag, Berlin, 2003. Matroids, trees, stable sets, Chapters 39–69.

[32] ———, *Combinatorial optimization. Polyhedra and efficiency. Vol. C*, vol. 24 of Algorithms and Combinatorics, Springer-Verlag, Berlin, 2003. Disjoint paths, hypergraphs, Chapters 70–83.

[33] D. B. SHMOYS, *Computing near-optimal solutions to combinatorial optimization problems*, in Combinatorial optimization (New Brunswick, NJ, 1992–1993), vol. 20 of DIMACS Ser. Discrete Math. Theoret. Comput. Sci., Amer. Math. Soc., Providence, RI, 1995, pp. 355–397.

[34] D. B. SHMOYS, É. TARDOS, AND K. AARDAL, *Approximation algorithms for facility location problems (extended abstract)*, in STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, ACM Press, 1997, pp. 265–274.

[35] K. VARADARAJAN AND G. VENKATARAMAN, *Graph decomposition and a greedy algorithm for edge-disjoint paths*, in Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms (SODA'04), Philadelphia, PA, USA, 2004, Society for Industrial and Applied Mathematics, pp. 379–380.

[36] V. V. VAZIRANI, *Approximation algorithms*, Springer-Verlag, Berlin, 2001.

[37] D. P. WILLIAMSON, *The primal-dual method for approximation algorithms*, Math. Program., 91 (2002), pp. 447–478. ISMP 2000, Part 1 (Atlanta, GA).