CS880: Approximations Algorithms

Scribe: Tom Watson **Lecturer:** Shuchi Chawla
Topic: Primal-Dual Method: Vertex Cover and Steiner Forest **Date:** 3/6/2007

In the previous few lectures we have seen examples of LP-rounding, a method for obtaining approximation algorithms that involves solving a linear programming relaxation of the problem at hand and rounding the solution. In the last lecture we also discussed the basic theory of LP-duality. Today we will apply this theory to obtain a second LP-based technique for obtaining approximation algorithms — the *primal-dual method*. This technique has the advantage that it circumvents the need to actually solve an LP relaxation, leading to efficient algorithms that are purely combinatorial. We will apply this technique to the Vertex Cover and Steiner Forest problems. The primal-dual method in the context of approximation algorithms was first used by Goemans and Williamson [1].

13.1 Linear Programming Duality

Consider the general linear program

$$\begin{array}{rl} \text{minimize} & c^T x \\ \text{subject to} & Ax \geq b \\ & x \geq 0 \end{array}$$

where $x = (x_1, \ldots, x_n)^T$ is vector of variables, $A = (A_{ij})$ is an $m \times n$ matrix, and $c = (c_1, \ldots, c_n)^T$ and $b = (b_1, \ldots, b_m)^T$. Recall that the dual of this linear program is given by

$$\begin{array}{rl} \text{maximize} & b^T y \\ \text{subject to} & A^T y \leq c \\ & y \geq 0 \end{array}$$

where $y = (y_1, \ldots, y_m)^T$ is a vector of variables. The variables in y are in one-to-one correspondence with the constraints of the primal LP, and the variables in x are in one-to-one correspondence with the constraints of the dual LP. In the last lecture we showed that the dual of the dual of an LP is again the original LP. We also proved the following result, stating that the objective value of every feasible solution to the dual lower bounds the objective value of every feasible solution to the primal. We reiterate the proof since it will come in handy later.

Theorem 13.1.1 (Weak Duality Theorem) *If x and y are feasible solutions to the primal and dual respectively, then $c^T x \geq b^T y$.*

Proof: We have

$$c^T x = \sum_{i=1}^{n} c_i x_i \geq \sum_{i=1}^{n} \Big(\sum_{j=1}^{m} A_{ji} y_j \Big) x_i = \sum_{j=1}^{m} \Big(\sum_{i=1}^{n} A_{ji} x_i \Big) y_j \geq \sum_{j=1}^{m} b_j y_j = b^T y,$$

where the two inequalities hold by feasibility of x and y. ■

We also stated, but did not prove, the following result, which establishes an intimate connection between the primal and dual LPs.

Theorem 13.1.2 (Strong Duality Theorem) *The optimal objective value of the primal is finite if and only if the optimal objective value of the dual is finite, and in this case the optimal objective values are equal.*

If x and y are optimal solutions to the primal and dual LPs, then Theorem 13.1.2 tells us that $c^T x = b^T y$, and it follows that both inequalities in the proof of Theorem 13.1.1 must hold with equality. That is,

$$\sum_{i=1}^{n} c_i x_i = \sum_{i=1}^{n} \Big(\sum_{j=1}^{m} A_{ji} y_j \Big) x_i$$

and

$$\sum_{j=1}^{m} b_j y_j = \sum_{j=1}^{m} \Big(\sum_{i=1}^{n} A_{ji} x_i \Big) y_j.$$

Let us consider the first equation. Since $\sum_{j=1}^{m} A_{ji} y_j \leq c_i$ for all i, the only way the first equation can hold is if $c_i x_i = (\sum_{j=1}^{m} A_{ji} y_j) x_i$ for all i. This is certainly true if $c_i = \sum_{j=1}^{m} A_{ji} y_j$ for all i, i.e. all dual constraints are tight. However, this is not necessary; even if $c_i < (\sum_{j=1}^{m} A_{ji} y_j)$ for some i, we can still have $c_i x_i = (\sum_{j=1}^{m} A_{ji} y_j) x_i$ provided $x_i = 0$. Similarly, for all j, either $y_j = 0$ or $b_j = \sum_{i=1}^{n} A_{ji} x_i$. Conversely, if x and y are feasible solutions to the primal and dual respectively such that these conditions hold, then equality holds throughout the proof of Theorem 13.1.1, implying that x and y have the same objective value and are thus both optimal. This proves the following result.

Lemma 13.1.3 *Let x and y be feasible solutions to the primal and dual respectively. Then x and y are both optimal if and only the following two conditions hold.*

Primal complementary slackness conditions: *For $i = 1, \ldots, n$, either $x_i = 0$ or $\sum_{j=1}^{m} A_{ji} y_j = c_i$.*

Dual complementary slackness conditions: *For $j = 1, \ldots, m$, either $y_j = 0$ or $\sum_{i=1}^{n} A_{ji} x_i = b_j$.*

This suggests an approach for finding optimal solutions to the primal and dual LPs: search for feasible solutions satisfying both complementary slackness conditions. We can use this idea to obtain approximation algorithms by searching for feasible solutions satisfying a relaxed version of the complementary slackness conditions. We say that x and y satisfy the *α-approximate dual complementary slackness conditions* if for $j = 1, \ldots, m$, either $y_j = 0$ or $\sum_{i=1}^{n} A_{ji} x_i \leq \alpha b_j$. That is, when $y_j \neq 0$, we don't require the corresponding primal constraint to be tight, but it shouldn't be too far from being tight. The following lemma is useful for proving approximation guarantees.

Lemma 13.1.4 *Suppose x and y are feasible solutions to the primal and dual respectively, satisfying the primal complementary slackness conditions and the α-approximate dual complementary slackness conditions. Then x is an α-approximate solution to the primal LP.*

Proof: We have

$$c^T x = (A^T y)^T x = y^T A x \leq \alpha y^T b,$$

where the first equality follows from the primal complementary slackness conditions and the inequality follows from the α-approximate dual complementary slackness conditions. The lemma follows since $y^T b$ is at most the optimal objective value of the primal LP, by Theorem 13.1.1. ■

Our strategy is to construct feasible solutions x and y such that x is integral and the primal complementary slackness conditions and α-approximate dual complementary slackness conditions are satisfied. We do so *without actually solving the LP*, which makes this approach appealing from a practical standpoint. Lemma 13.1.4 then guarantees that x is an α-approximate solution to the LP relaxation and hence an α-approximate solution to the problem at hand. This is the main idea behind the primal-dual method.

This is not the only way to design primal-dual algorithms, however. For example, our primal-dual algorithm for the Steiner Forest problem does not satisfy the 2-approximate complementary slackness conditions for every j, yet we can show that nevertheless, we obtain a 2-approximate solution.

13.2 Vertex Cover

13.2.1 Linear Programming Formulation

Our first example of a primal-dual algorithm is for the weighted version of the Vertex Cover problem.

Definition 13.2.1 (Vertex Cover) *Given a graph $G = (V, E)$ and vertex weights $w : V \to \mathbb{R}^+$, find a minimum weight subset of the vertices such that every edge is covered.*

We can capture this problem with an ILP that is very similar to the ILP we saw in a previous lecture for the unweighted version. We introduce a variable x_v for each vertex v to indicate whether v is in the chosen vertex cover. We seek to minimize the total weight of the picked vertices subject to the constraint that for every edge, at least one of its endpoints is picked.

$$\begin{array}{rll} \text{minimize} & \sum_{v \in V} w_v x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \forall \{u, v\} \in E \\ & x_v \in \{0, 1\} & \forall v \in V \end{array}$$

We relax this to the following LP.

$$\begin{array}{rll} \text{minimize} & \sum_{v \in V} w_v x_v & \\ \text{subject to} & x_u + x_v \geq 1 & \forall \{u, v\} \in E \\ & x_v \geq 0 & \forall v \in V \end{array}$$

The dual of this LP can be written by inspection. Since the constraints of the primal correspond to the edges of G, we have a dual variable y_e for each edge.

$$\begin{array}{rll} \text{maximize} & \sum_{e \in E} y_e & \\ \text{subject to} & \sum_{e : v \in e} y_e \leq w_v & \forall v \in V \\ & y_e \geq 0 & \forall e \in E \end{array}$$

Incidentally, if all vertex weights are 1, then the ILP corresponding to this dual LP exactly captures the Maximum Matching problem. Theorem 13.1.1 then tells us that the cardinality of every matching is a lower bound on the size of every vertex cover, which is exactly the lower bound we used to design a 2-approximation algorithm for the unweighted version of Vertex Cover in the first lecture. That algorithm can be viewed as a primal-dual algorithm and the algorithm we are about to describe can be viewed as a generalization.

13.2.2 Primal-Dual Algorithm

We begin by giving a high-level overview of the primal-dual method. We assume throughout this discussion that the problem at hand is a minimization problem where both c and b are positive. We start with a dual feasible solution and a primal infeasible solution, typically $y = (0, \ldots, 0)^T$ and $x = (0, \ldots, 0)^T$. These solutions certainly satisfy the primal and dual complementary slackness conditions. We iteratively modify x and y, making x closer to being feasible and y closer to being optimal, while maintaining the dual feasibility of y, the primal complementary slackness conditions, and the α-approximate dual complementary slackness conditions. If we can get to a point where x is feasible, then we can use Lemma 13.1.4 to conclude that x is an α-approximate solution.

A more detailed plan is as follows.

- Start with $x = (0, \ldots, 0)^T$ and $y = (0, \ldots, 0)^T$.
- Intuitively, we want our dual solution y to have as high an objective value as possible, since this is our lower bound. So begin by raising some y variables, improving the dual objective value, until some dual constraint goes tight. The crucial design aspect is deciding how to raise the variables — this depends on the structure of the problem at hand.
- When some dual constraint goes tight, we are then free to raise the x variable corresponding to that constraint while maintaining the primal complementary slackness condition. This makes the primal solution x closer to being feasible. In particular, we usually raise the x variable in such a way that all the primal constraints involving this variable are satisfied.
- At this point, raising any of the y variables involved in the constraint that just went tight would violate that constraint, so "freeze" all of these variables and repeat the whole process, alternating between raising y and raising x, and only selecting unfrozen y variables to raise.
- Finally, show that when all y variables become frozen, the primal solution x becomes feasible, and that the α-approximate dual complementary slackness conditions are satisfied.

We now show how to employ this method to obtain a 2-approximation algorithm for Vertex Cover. We start out with $x = (0, \ldots, 0)^T$, i.e. no vertex is picked, and $y = (0, \ldots, 0)^T$, i.e. every edge has a label of 0. We seek to improve the feasibility of the primal, so we pick an arbitrary unsatisfied edge e and raise its label y_e until some dual constraint, say the one corresponding to vertex v, goes tight, which is to say the sum of the labels of edges incident with v is exactly the weight of v. The fact that the primal complementary slackness conditions would still be satisfied if we set $x_v = 1$ is a hint that we should pick v to be in the cover. So we set $x_v = 1$, and since increasing the label

of any edge incident with v would violate the dual constraint corresponding to v, we freeze the dual variables associated with these edges. If the constraints corresponding to both endpoints of e go tight at the same time, then we put both of them in the cover and freeze all edges incident with them. Then we repeat the process by picking an unfrozen edge, raising its label until some dual constraint goes tight, putting the corresponding vertex in the cover, and freezing the edges incident with it. We continue until all edges are frozen, and then output the set of vertices v such that $x_v = 1$.

We remark that although we think of the dual variable y_e as being raised in continuous time, an implementation of this algorithm just needs to set $y_e = \min_{v \in e} (w_v - \sum_{e' \ni v} y_{e'})$.

Lemma 13.2.2 *At termination, y is dual feasible.*

Proof: When a dual constraint goes tight, we freeze all dual variables appearing in it, so this constraint cannot be violated. ■

Lemma 13.2.3 *At termination, the primal complementary slackness conditions are satisfied.*

Proof: We only set $x_i = 1$ when the corresponding dual constraint is tight. ■

Lemma 13.2.4 *At termination, x is primal feasible.*

Proof: Suppose for contradiction that some edge e is not covered by the corresponding vertex cover. Since all frozen edges are incident with picked vertices, it must be the case that e is not frozen. But the algorithm does not terminate with unfrozen edges. ■

Lemma 13.2.5 *At termination, the 2-approximate dual complementary slackness conditions are satisfied.*

Proof: This follows from the fact that for each edge $\{u, v\}$, $x_u \leq 1$ and $x_v \leq 1$ and therefore $x_u + x_v \leq 2$. ■

Although trivial to prove, the above lemma dictates our final approximation guarantee.

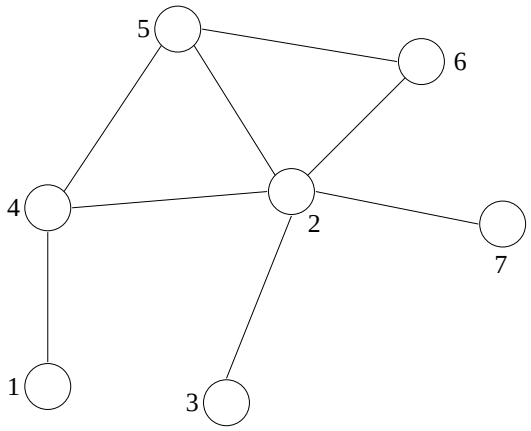
Theorem 13.2.6 *The primal-dual algorithm described above is a 2-approximation algorithm for Vertex Cover.*

Proof: This follows from Lemma 13.1.4 and the previous four lemmas. ■

13.2.3 Examples

We illustrate the behavior of our algorithm on the example in the following figure.

The algorithm may begin by choosing to raise the edge between the vertices of weight 4 and 5. Once the label of this edge hits 4, the constraint corresponding to the vertex of weight 4 goes tight, since the sum of the labels of the edges incident with it is $4 + 0 + 0 = 4$. This vertex is picked to be in the cover, and the three edges incident with it become frozen. The algorithm may then pick the edge between the vertices of weight 5 and 6 to raise. Once the label of this edge hits 1, the constraint corresponding to the vertex of weight 5 goes tight, since the sum of the labels of the edges incident with it is $4 + 0 + 1 = 5$. This vertex is picked to be in the cover, and the two edges incident with it that weren't already frozen become frozen. There are now three unfrozen edges; the algorithm may pick the edge between the vertices of weight 2 and 7 to raise. Once the label of



this edge hits 2, the constraint corresponding to the vertex of weight 2 goes tight, so this vertex is picked to be in the cover, and the remaining three edges become frozen. Now each edge is frozen, so each edge is incident on a picked vertex. The algorithm outputs the vertices of weight 2, 4, and 5, which form a vertex cover of total weight 11. Note that the output is not optimal; the vertex of weight 4 could be replaced with the vertex of weight 1 to yield a vertex cover of total weight 8.

Finally, we observe how this algorithm behaves when all vertex weights are 1. It selects an edge and raises the corresponding variable to 1, at which point the constraints corresponding to both endpoints go tight, so both endpoints are picked to be in the cover. All edges incident with these two endpoints become frozen, so every edge raised in the future will not share an endpoint with the edge raised in this iteration. In other words, the algorithm finds a maximal matching in G and outputs all endpoints of edges in the matching. This is exactly the 2-approximation algorithm for the unweighted version of Vertex Cover that was described in the first lecture.

13.3 Steiner Forest

13.3.1 Linear Programming Formulation

The primal-dual algorithm we just described for Vertex Cover worked by raising dual variables one at a time. Our next example is a primal-dual algorithm that operates differently; it raises many dual variables simultaneously until one of the corresponding primal constraints goes tight. This algorithm also has the feature that Lemma 13.1.4 does not apply directly; a more ad-hoc argument is needed. We choose to study the following problem instead of the Steiner Tree problem since the extra generality doesn't end up complicating the algorithm or analysis significantly.

Definition 13.3.1 (Steiner Forest) *Given a graph $G = (V, E)$, edge costs $c : E \to \mathbb{R}^+$, and sets $S_i \subseteq V$, find a minimum-cost forest F such that for all i and all $u, v \in S_i$, there is a path from u to v in F.*

Our first task is to fomulate this problem as an ILP. There are several ways of doing this. One natural approach would be to have a variable for every edge indicating whether it is in the forest,

as well as a variable for each path connecting two vertices in the same S_i, and constraints enforcing that every pair of vertices in the same S_i is connected by some path and that every edge on this path is indeed picked. This yields an ILP with exponentially many constraints. A polynomial-sized ILP can be obtained by expressing the problem as a flow problem, with a different commodity for each pair of vertices in the same S_i. We will use an equivalent but more structured formulation with a dual that is simpler to state.

We still have a variable x_e for each edge indicating whether it is in the forest, but instead of thinking of pairs of vertices in the same S_i as being *connected*, we think of them as being *not disconnected*. That is, every cut separating them must contain some picked edge. More formally, for every $S \subseteq V$ such that $S \cap S_i \notin \{\emptyset, S_i\}$ for some i, we require that $x_e = 1$ for at least one edge $e \in \delta(S)$ have . Letting $\mathcal{S}$ denote the set of all such S, we have the following ILP formulation.

$$\begin{array}{rll} \text{minimize} & \sum_{e \in E} c_e x_e & \\ \text{subject to} & \sum_{e \in \delta(S)} x_e \geq 1 & \forall S \in \mathcal{S} \\ & x_e \in \{0, 1\} & \forall e \in E \end{array}$$

An immediate concern is that the above ILP has exponentially many constraints. However, this serves to illustrate a key point about the primal-dual method: the linear programming formulation of a problem is merely a conceptual tool; the algorithms are purely combinatorial. In this case, our algorithm only ever needs to deal with polynomially many of the primal constraints and dual variables, so the exponential size of the LP formulation is not prohibitive.

We obtain the LP relaxtion for this problem.

$$\begin{array}{rll} \text{minimize} & \sum_{e \in E} c_e x_e & \\ \text{subject to} & \sum_{e \in \delta(S)} x_e \geq 1 & \forall S \in \mathcal{S} \\ & x_e \geq 0 & \forall e \in E \end{array}$$

For the dual, we introduce a variable y_S for each $S \in \mathcal{S}$. The dual LP can be written by inspection.

$$\begin{array}{rll} \text{maximize} & \sum_{s \in \mathcal{S}} y_S & \\ \text{subject to} & \sum_{S : e \in \delta(S)} y_S \leq c_e & \forall e \in E \\ & y_S \geq 0 & \forall S \in \mathcal{S} \end{array}$$

13.3.2 Primal-Dual Algorithm

Our primal-dual algorithm starts with the primal infeasible solution $x = (0, \ldots, 0)^T$ and the dual feasible solution $y = (0, \ldots, 0)^T$. We would like to raise some dual variables, thereby increasing the dual objective, until some dual constraint goes tight, at which time the primal complementary slackness conditions would indicate that it is safe to put the corresponding edge in our forest. For the Vertex Cover problem, we picked a single unsatisfied primal constraint and raised the corresponding dual variable until some dual constraint went tight. The order in which the unsatisfied constraints were picked didn't affect the final cost much. In the present settting, it turns out that raising one dual variable at a time does not work. The costs for satisfying different constraints can vary

wildly, so ordering matters more. Instead, we raise many dual variables simultaneously until some dual constraint goes tight. This way, we are not focusing on making progress toward satisfying any particular primal constraint, but rather taking a more global perspective.

Another difference between our Vertex Cover algorithm and the current example is that we are not able to guarantee that the α-approximate dual complementary slackness conditions are satisfied. But nevertheless we can show that our algorithm yields a 2-approximate solution, by a similar argument to the one used to show that if the 2-approximate dual complementary slackness conditions are satisfied then the resulting solution is 2-approximate. In particular, we will show that these constraints hold "on average".

We will see more details about this algorithm in the next lecture. We will also see a primal-dual algorithm for the Facility Location problem.

References

[1] M. X. Goemans and D. P. Williamson. A General Approximation Technique for Constrained Forest Problems. In *SIAM Journal on Computing*, 24, 1995, pp. 296-317.