6.891 Approximation Algorithms	March 31, 2000
Lecture 8	
Lecturer: David P. Williamson	*Scribe: David P. Williamson*

1 The Primal-Dual Method

1.1 Introduction

We introduced the primal-dual method in looking at the set cover problem on the first day of class. The primal-dual method turns out to have a fairly wide range of applications, and we will look at some of them in the next two lectures.

To begin we look at the following problem.

Hitting Set

- **Input:**
  - ground set $E = \{e_1, e_2, \ldots, e_n\}$
  - subsets $T_1, T_2, \ldots, T_p \subseteq E$
  - costs $c_e \geq 0 \quad e \in E$
- **Goal:** Find min-cost $A \subseteq E$ s.t. $A \cap T_i \neq \emptyset \quad \forall i$.

Note that this problem is equivalent to the Set Cover problem. Namely, the ground element e_i of the Hitting Set corresponds to the subset S_i of the Set Cover, and the subset T_i of the Hitting Set corresponds to the ground element t_i of the Set Cover.

To apply the primal-dual method, we first need a primal and a dual. The integer program below models the hitting set problem.

$$\text{Min} \quad \sum_{e \in E} c_e x_e$$

subject to:

$$\sum_{e \in T_i} x_e \geq 1 \qquad \forall i$$

$$x_e \in \{0,1\}.$$

The integer program can be relaxed to a linear program in the usual way:

$$x_e \in \{0,1\} \rightarrow x_e \geq 0.$$

Once we have an LP relaxation, we can take the dual of this LP, and obtain the following:

$$\text{Max} \quad \sum_i y_i$$

subject to:

$$\begin{aligned} \sum_{i:e\in T_i} y_i &\le c_e && \forall e \in E \\ y_i &\ge 0 && \forall i. \end{aligned}$$

The basic goal in the primal dual method is to obtain a solution $A \subseteq E$ and a dual feasible solution y such that

$$\sum_{e\in A} c_e \le \alpha \sum_{i=1}^{p} y_i \le \alpha OPT,$$

for some value of α. For a while now, we have claimed that the last inequality is true (i.e. $\sum_{i=1}^{p} y_i \le OPT$). This is not difficult to prove, so we do it here.

Lemma 1 (Weak duality) *For any feasible dual solution y and any feasible primal solution x,*

$$\sum_{i=1}^{p} y_i \le \sum_{e\in E} c_e x_e.$$

Proof:

$$\begin{aligned} \sum_{e\in E} c_e x_e &\ge \sum_{e\in E}\left(\sum_{i:e\in T_i} y_i\right) x_e && (1)\\ &= \sum_{i=1}^{p} y_i \sum_{e\in T_i} x_e && (2)\\ &\ge \sum_{i=1}^{p} y_i, && (3) \end{aligned}$$

where inequality (1) follows by dual feasibility, equation (2) by switching the summations, and inequality (3) by primal feasibility. □

We'd like to get our desired inequality above by reversing these inequalities. The first step toward doing this is making the first inequality (1) an equation. Given a feasible dual solution y, we do this by setting $A = \{e \in E : \sum_{i:e\in T_i} y_i = c_e\}$. It then follows that

$$\sum_{e\in A} c_e = \sum_{e\in A}\sum_{i:e\in T_i} y_i = \sum_{i=1}^{p} y_i |A \cap T_i| \le \alpha \sum_{i=1}^{p} y_i,$$

where $\alpha = \max_{i:y_i>0} |A \cap T_i|$. Certainly $\alpha \le \max_i |T_i|$.

How do we ensure a feasible solution A, though? Note that if A is not feasible then it does not hit every set, so there is some T_k such that $A \cap T_k = \emptyset$. By construction of A this means that $\forall e \in T_k$, $\sum_{i:e\in T_i} y_i < c_e$. Now y_k appears in these constraints and only in these constraints. It thus follows that every constraint in which y_k participates is not tight ($c_e - \sum_{i:e\in T_i} y_i > 0$), and we can therefore increase y_k by the minimum of these differences while keeping all of these differences ≥ 0. Thus none of the constraints containing y_k are violated - though the one corresponding to that minimum is now tight. Given this new dual feasible solution we may now set x_e for e corresponding to that newly tight constraint to 1, i.e. we may add e to A and the new A will hit T_k as well.

This leads to the following basic algorithm:

Primal-Dual

$y \leftarrow 0$
$A \leftarrow \emptyset$
While A is not feasible
    Choose some violated T_k (i.e. T_k s.t. $A \cap T_k = \emptyset$)
    Increase y_k until $\exists\ e \in T_k$ such that $\sum_{i:e \in T_i} y_i = c_e$
    $A \leftarrow A \cup \{e\}$
Return A.

As an example, we apply this algorithm to the vertex cover problem. Note that vertex cover can be translated into a hitting set problem, where V is the ground set of elements, the costs c_i of the elements are the weights of the vertices, and we must hit the sets $T_i = \{u, v\}$ for each $(u, v) \in E$. Then since $|T_i| = 2$ for each set, it follows that $|A \cap T_i| \leq 2$ for all i, and by the reasoning above we have a 2-approximation algorithm for vertex cover.

1.2 Feedback Vertex Set Problem

Let us consider another example:

Feedback Vertex Set in Undirected Graphs

- **Input:**
  - Undirected graph $G = (V, E)$
  - Weights $w_i \geq 0 \quad \forall i \in V$
- **Goal:** Find $S \subseteq V$ minimizing $\sum_{i \in S} w_i$ such that for every cycle C in G, $C \cap S \neq \emptyset$. (Equivalently, find a min-weight set of vertices S such that removing S from the graph causes the remaining graph to be acyclic).

We claim that the feedback vertex set problem is just a hitting set problem with:

- Ground set V
- Cost w_i
- Sets to hit: $T_i = C_i$ for each cycle C_i in graph

We now have a hitting set problem with potentially an exponential number of sets to hit. How do we deal with this problem? The answer is that we do not need to enumerate or find all cycles: the algorithm only needs to find a violated set when one exists.

We want to apply the primal-dual method here and get a good performance guarantee. This follows if we can show that $|A \cap C_i| \leq \alpha$ for all cycles C_i on which $y_i > 0$. This would be true if $|C_i| \leq \alpha$ for all cycles i, but α can be as large as n in this case. To be able to bound $|A \cap C_i|$ in this case, we consider only "interesting" vertices. First of all, any vertex not in any cycle is not interesting, since it can't participate in the solution set. Additionally, on any path of nodes of degree 2, only one vertex is interesting; namely, the vertex of least cost, since any cycle which goes through the vertices on this path also goes through this vertex of least cost. Thus we may as well only consider the vertex of least cost.

To get our algorithm, we need the following lemma:

Lemma 2 *(Erdös, Posa) In every non-empty graph which is not a forest, there is a cycle of at most* $4 \log_2 n$ *interesting vertices.*

Proof: From G we create a graph G' consisting of only interesting vertices by deleting vertices not in any cycle, and shortcutting all remaining uninteresting vertices out of G. Note that there is still a one-to-one correspondance of cycles in G and G'. Now G' has no degree 1 vertices, and each vertex of degree 2 has two neighbors of degree more than 2.

We now find the requisite cycle. We do a breadth-first search of G'. By the properties of the graph, if we do not close a cycle by revisiting a previously explored node, then at least in every other level the number of explored nodes increases by a factor of 2. Thus at depth i, we will have explored $2^{i/2}$ nodes. By depth $2 \log_2 n$, we will have found a cycle. $\square$

We can now state our algorithm, but first we restate the primal and dual LPs in terms of cycles.

$$\text{Min} \quad \sum_{i \in V} w_i x_i$$

subject to:

$$\sum_{e \in C_i} x_i \geq 1 \qquad\qquad \forall \text{ cycles} C_i$$

$$x_e \geq 0$$

$$\text{Max} \quad \sum_{C} y_C$$

subject to:

$$\sum_{C:i\in C} y_C \le w_i \qquad \forall i \in V$$
$$y_C \ge 0 \qquad \forall \text{ cycles } C$$

Primal-DualFVS

$y \leftarrow 0$
$A \leftarrow \emptyset$
$I \leftarrow$ interesting vertices of G
While A is not feasible
    Find cycle C s.t. $|C \cap I| \le 4\log_2 n$
    Increase y_C until $\exists\ i \in I : \sum_{C:i\in C} y_C = w_i$
    $A \leftarrow A \cup \{i\}$
    Remove i from graph, remove all vertices no longer in cycles from I,
        remove new uninteresting verts (w.r.t. to prices
        $\tilde{w_i} = w_i - \sum_{C:i\in C} y_i$).
Return A.

We should observe that no uninteresting vertex ever becomes interesting again. Certainly any vertex not in any cycle will be part of a cycle again. Similarly, any uninteresting vertex on a path of degree two nodes will always be in the same cycles as the interesting vertex on that path, and thus choosing the interesting vertices in the way specified above guarantees that an interesting vertex will be one for which the dual inequality becomes tight first. This also implies that the dual solution we construct is feasible. We leave a formal proof of this to the reader.

Theorem 3 *(Bar-Yehuda, Geiger, Naor, Roth '94) PrimalDualFVS is a* $(4log_2 n)$*-approximation algorithm for the feedback vertex set problem in undirected graphs.*

Proof: By using the standard primal-dual analysis, we have that

$$\begin{aligned}
\sum_{i\in A} w_i &= \sum_{i\in A}\sum_{C:i\in C} y_C \\
&= \sum_C y_C |A \cap C| \\
&\le \sum_C y_C |I \cap C| \\
&\le (4\log_2 n)\sum_C y_C \\
&\le (4\log_2 n) OPT.
\end{aligned}$$

□

In fact, one can get a 2-approximation algorithm for this problem using the primal-dual method; however, one has to use a substantially different integer programming formulation of the problem, and we will not get into it here.

1.3 Shortest s-t path

Here we consider the problem of finding the shortest s-t path in an undirected graph. This problem can be seen as a hitting set problem as follows:

Ground Set : the set of edges E
Costs : $c_e \geq 0, \ \forall e \in E$
Sets to Hit : $T_i = \delta(S_i), \ s \in S_i, \ t \notin S_i$

where $\delta(S) = \{(u,v) \in E : \ u \in S \text{ and } v \notin S\}$. That is, the sets S_i are the s-t cuts and the sets $T_i = \delta(S_i)$ are the edges crossing the s-t cuts.

To see that this hitting set problem captures the shortest s-t path problem, we need that a set of edges contains an s-t path if and only if it hits every s-t cut[1]. First, if a set of edges A does not cross some s-t cut S_i then A must consist exclusively of edges joining two vertices of S_i or joining two vertices of the complement of S_i. Thus any path starting from $s \in S_i$ consisting of such edges can only bring us to vertices that are also in S_i, but $t \notin S_i$. Conversely, if a set of edges does not contain an s-t path then let S_i be the largest connected component (corresponding to those edges) containing s. By assumption $t \notin S_i$ and the set of edges could not contain any edge from $\delta(S_i)$ or else we could have found a larger connected component containing s by including the other vertex incident on that edge. Thus the absence of an s-t path implies that some s-t cut was not hit. We find then that a set of edges contains an s-t path if and only if it hits every s-t cut.

We now wish to apply the primal-dual method to this problem. We set up our primal and dual:

$$\text{Min} \quad \sum_{e \in E} c_e x_e$$

subject to:

$$\sum_{e \in \delta(S)} x_e \geq 1 \qquad \forall S : s \in S, t \notin S$$
$$x_e \geq 0$$

$$\text{Max} \quad \sum_{S} y_S$$

subject to:

$$\sum_{S : e \in \delta(S)} y_S \leq c_e \qquad \forall e \in E$$
$$y_S \geq 0 \qquad \forall S : s \in S, t \notin S$$

[1]This follows directly from the max-flow/min-cut theorem, but for completeness we prove it here.

To use our primal-dual algorithm, we have to say which violated s-t cut we will choose in each iteration of the algorithm. We will choose the cut S which is the connected component containing S. By the reasoning above, S will not contain t unless we have already selected an s-t path.

The problem with the standard primal-dual method is that it will include too many edges not on the path from s to t. Hence we will need to add some steps at the end to get rid of all edges not on the s-t path. We do this in a particular fashion below (in the reverse of the order in which edges are added) not so much because it is useful for this particular problem, but because it is useful for several other problems.

Primal-DualSP

```
y ← 0
A ← ∅
l ← 0 (l is a counter)
While A is not feasible
    l ← l + 1
    Let S be the connected component containing s
    Increase y_S until ∃ e_l ∈ δ(S) : Σ_{S:e_l∈δ(S)} y_S = c_{e_l}
    A ← A ∪ {e_l}
For j ← l down to 1
    If A − {e_j} is still feasible
        A ← A − {e_j}
Return A' ← A.
```

Once again, we have that

$$\sum_{e\in A'} c_e = \sum_{e\in A'} \sum_{S:e\in\delta(S)} y_S = \sum_S y_S |A' \cap \delta(S)|.$$

In the following theorem, we will see that for all S such that $y_S > 0$, $|A' \cap \delta(S)| = 1$. Thus this algorithm is optimal; in fact, it is just Dijkstra's algorithm cleverly disguised.

Theorem 4 *For all S such that $y_S > 0$, $|A' \cap \delta(S)| = 1$.*

Proof: Let A_S be the edges in A when we increased the variable y_S. Let $B = A' - A_S$. Let k be the iteration in which we increase y_S. Observe that $A_S \cup B$ contains an s-t path, and furthermore, that if we remove any edge $e \in B$, $A_S \cup B - e$ no longer contains an s-t path. This follows because when we go through the edge deletion step at the end of the algorithm, when we reach the consideration of edge e_k, the remaining edges in A are exactly $A_S \cup B$, and all edges in B were necessary for feasibility.

Now let $s, v_1, v_2, \ldots, v_l, t$ be an s-t path in $(V, A_S \cup B)$. Choose i such that $v_i \in S$, $v_{i+1} \notin S$ where i is as large as possible. Since S is a connected component there must be a s-v_i path exclusively in S of the form $s, w_1, w_2, \ldots, w_j, v_i$, where

$w_\ell \in S$. Thus $s, w_1, w_2, \ldots, w_j, v_i, v_{i+1}, \ldots, v_l, t$ is an s-t path. Since all the edges $(v_i, v_{i+1}), \ldots, (v_l, t)$ are not in A_S, they must be in B. Therefore

$$|A' \cap \delta(S)| = |B \cap \delta(S)| = |\{(v_i, v_{i+1})\}| = 1,$$

since the first edge is the only one to have an endpoint in S. □

1.4 Generalized Steiner Trees

We now consider another problem for which the primal-dual method gives a good approximation algorithm, the Generalized Steiner Tree problem.

Generalized Steiner Tree Problem

- **Input:**
  - An undirected graph $G = (V, E)$
  - l pairs of vertices $(s_i, t_i), i = 1 \ldots l$
  - Costs $c_e \geq 0$ for each edge $e \in E$
- **Goal:** Find a minimum-cost set of edges F such that for all i, s_i and t_i are in the same connected component of (V, F).

This can be modelled as a hitting set problem:

Ground Set : the set of edges E
Costs : $c_e \geq 0, \forall e \in E$
Sets to Hit : $T_i = \delta(S_i)$ iff $|S_i \cap \{s_j, t_j\}| = 1$ for some j (the s_j-t_j cuts).

Note that by the logic we used for the shortest s-t path problem that a set of edges will be feasible for this hitting set problem if and only if it contains a path between s_i and t_i for each i.

Let us consider how to apply the primal-dual method to this problem. Suppose we do more or less the same thing here we did for the shortest s-t path problem. We know that if A is not feasible then there must be some connected component S containing s_j but not t_j for some j. Suppose the algorithm picks $\delta(S)$ as the violated set and increases its dual. The difficulty is that the reasoning used above in the s-t path problem will not yield a good bound here since a solution A' may cross the cut many times. Consider the problem for which $s = s_1 = s_2 = \cdots = s_l$ and for which there are edges $(s, t_j), \forall j$ and say that $A' = \{(s, t_j)\}, j = 1 \ldots n$ is a solution in which all edges are necessary, and that crosses the cut $\delta(\{s\})$ l times, which implies an l-approximation algorithm (since $|A' \cap \delta(S)| = l$). This is not very good.

Perhaps we picked the wrong infeasible solution to augment; that is, maybe our previous algorithm will work if we change the way that we choose the violated set.

It turns out that this approach does not work either because it still leads to a l-approximation in the worst case scenario (where all the s's are in one group, unconnected to any of the t's).

Suppose we consider all the reasonable choices of violated sets (e.g. all connected components S containing one of s_j or t_j for some j). If we look at the example above, the average of $|A' \cap \delta(S)|$ over these S is $\frac{l+l}{l+1} < 2$. Therefore it might be wise to pick several violated sets and increase the associated dual variables all at the same time. We do this in the algorithm below:

Primal-DualGST

> $y \leftarrow 0$
> $A \leftarrow \emptyset$
> $l \leftarrow 0$ (l is a counter)
> While A is not feasible
> $l \leftarrow l + 1$
> $\mathcal{C}_l \leftarrow \{S : S$ a connected component of $(V, A) : |S \cap \{s_j, t_j\} = 1\}$
>   Increase y_S for all $S \in \mathcal{C}_i$ uniformly until $\exists\ e_l \notin A : \sum_{S: e_l \in \delta(S)} y_S = c_{e_l}$
> $A \leftarrow A \cup \{e_l\}$
> For $j \leftarrow l$ down to 1
>   If $A - \{e_j\}$ is still feasible
> $A \leftarrow A - \{e_j\}$
> Return $A' \leftarrow A$.

We claim that our statement about the average over this choice of violated sets holds in general.

Claim 5

$$\sum_{C \in \mathcal{C}_i} |A' \cap \delta(C)| \leq 2|\mathcal{C}_i|.$$

From the claim we can prove the following lemma.

Lemma 6 *Claim 5 implies that*

$$\sum_{e \in A'} c_e \leq 2 \sum_S y_S.$$

Proof: From prior analysis we know that

$$\sum_{e \in A'} c_e = \sum_{e \in A'} \sum_{S: e \in \delta(S)} y_S = \sum_S y_S |A' \cap \delta(S)|.$$

Hence we want to show that

$$\sum_S y_S |A' \cap \delta(S)| \leq 2 \sum_S y_S.$$

We prove this by induction on the construction of y by the algorithm. In the base case, $y = 0$, so the inequality is true. Suppose the inequality holds in the ith iteration. In iteration $i+1$, suppose we increase all dual variables y_S for $S \in \mathcal{C}_i$ by ϵ. Then the right-hand side of the inequality increases by $2\epsilon|\mathcal{C}_{i+1}|$, and the left-hand side of the inequality increases by $\epsilon \sum_{C \in \mathcal{C}_i} |A' \cap \delta(C)|$. But given Claim 5, this means that the increase of the left-hand side is no more than the increase of the right-hand side, and thus the inequality continues to hold. □

We now prove the claim.

Theorem 7

$$\sum_{C \in \mathcal{C}_i} |A' \cap \delta(C)| \leq 2|\mathcal{C}_i|.$$

Proof: Consider A_i in iteration i. Let $B = A' - A_i$. As we argued in the case of the shortest s-t path problem, $A_i \cup B$ is a feasible solution to the problem, but for any $e \in B$, $A_i \cup B - e$ is not feasible.

Suppose we contract every connected component of (V, A_i). In this contracted graph, call the nodes corresponding to the connected components in $\mathcal{C}_i$ red and the rest blue. Now consider the graph $G' = (V', B)$ where V' is the vertex set. We note that G' must be a forest, since if it had a cycle we could remove an edge of the cycle and maintain feasibility, contradicting the fact that every edge in B is necessary.

How does the inequality we wish to prove translate to the graph G'? Note that $|A' \cap \delta(C)|$ in G for a connected component C is equal to $deg(v)$ in G' for the vertex v corresponding to S. Similarly, $|\mathcal{C}_i|$ in G is simply the number of red vertices in G'. We let *Red* and *Blue* represent the sets of red and blue vertices in G', so that we can rewrite the above inequality as

$$\sum_{v \in Red} deg(v) \leq 2|Red|.$$

We will need the following claim.

Claim 8 *If $v \in Blue$ then $deg(v) \neq 1$.*

Proof: If $deg(v) = 1$ then we claim $A_i \cup B - e$ is feasible for $e \in B$ and adjacent to v; this is a contradiction. Let S be the connected component in G that corresponds to the vertex v in G'. If $A_i \cup B - e$ is not feasible, then there must be some s_j-t_j pair that is connected in $(V, A_i \cup B)$ but not in $(V, A_i \cup B - e)$. Thus either s_j or t_j is in S, and the other vertex is in $V - S$. But then it must have been the case that $S \in \mathcal{C}_i$ and $v \in Red$, which is a contradiction. □

To complete the proof, we first discard all blue nodes with $deg(v) = 0$. Then

$$\begin{aligned}
\sum_{v \in Red} deg(v) &= \sum_{v \in Red \cup Blue} deg(v) - \sum_{v \in Blue} deg(v) \\
&\leq 2(|Red| + |Blue|) - 2|Blue| \\
&= 2|Red|
\end{aligned}$$

The inequality follows since the sum of the degrees of nodes in a forest is at most twice the number of nodes, and since every blue node has degree at least two. □

This 2-approximation algorithm for the generalized Steiner tree is just an example of the kind of graph problem for which the primal-dual method can obtain a good approximation algorithm. A generalization of the proof above gives 2-approximation algorithms for many other graph problems. The result above is due to Agrawal, Klein, Ravi '95 and Goemans, W '95.