CS787: Advanced Algorithms

Scribe: Mayank Maheshwari, Dan Rosendorf **Lecturer:** Shuchi Chawla
Topic: Primal-Dual Algorithms **Date:** October 19 2007

15.1 Introduction

In the last lecture, we talked about Weak LP-Duality and Strong LP-Duality Theorem and gave the concept of Dual Complementary Slackness (DCS) and Primal Complementary Slackness (PCS). We also analyzed the problem of finding maximum flow in a graph by formulating its LP.

In this lecture, we will develop a 2-approximation for Vertex Cover using LP and introduce the Basic Primal-Dual Algorithm. As an example, we analyze the primal-dual algorithm for vertex cover and later on in the lecture, give a brief glimpse into a 2-player zero-sum game and show how the pay-offs to players can be maximized using LP-Duality.

15.2 Vertex Cover

We will develop a 2-approximation for the problem of weighted vertex cover. So for this problem:

Given: A graph $G(V, E)$ with weight on vertex v as w_v.

Goal: To find a subset $V' \subseteq V$ such that each edge $e \in E$ has an end point in V' and $\sum_{v \in V'} w_v$ is minimized.

The Linear Program relaxation for the vertex cover problem can be formulated as:

The variables for this LP will be x_v for each vertex v. So our objective function is

$$min \sum_{v} x_v w_v$$

subject to the constraints that

$$x_u + x_v \geq 1 \qquad \forall (u, v) \in E$$

$$x_v \geq 0 \qquad \forall v \in V$$

The Dual for this LP can be written with variables for each edge $e \in E$ as maximizing its objective function:

$$max \sum_{e \in E} y_e$$

subject to the constraints

$$\sum_{v:(u,v) \in E} y_{uv} \leq w_u \qquad \forall u \in V$$

$$y_e \geq 0 \qquad \forall e \in E$$

Now to make things simpler, let us see how the unweighted case for vertex cover can be formulated in the form of a Linear Program. We have the objective function as:

$$max \sum_{e \in E} y_e$$

such that

$$\sum_{e \text{ incident on } u} y_e \leq 1 \qquad \forall u \in V$$

$$y_e \geq 0 \qquad \forall e \in E$$

These constraints in the linear program correspond to finding a matching in the graph G and so the objective function becomes finding a maximum matching in the graph. Hence, this is called the *Matching LP*.

Now remember from the previous lecture where we defined Dual and Primal Complementary Slackness. These conditions follow from the Strong Duality Theorem.

Corollary 15.2.1 (Strong Duality) *x and y are optimal for Primal and Dual LP respectively iff they satisfy:*

1. *Primal Complementary Slackness (PCS) i.e. $\forall i$, either $x_i = 0$ or $\sum_j A_{ij} y_j = c_i$.*
2. *Dual Complementary Slackness (DCS) i.e. $\forall j$, either $y_j = 0$ or $\sum_i A_{ij} x_i = b_j$.*

15.3 Basic Primal-Dual Algorithm

1. Start with $x = 0$ (variables of primal LP) and $y = 0$ (variables of dual LP) . The conditions that:

   - y is feasible for Dual LP.
   - Primal Complementary Slackness is satisfied.

   are invariants and hence, hold for the algorithm. But the condition that:

   - Dual Complementary Slackness is satisfied.

   might not hold at the beginning of algorithm. x does not satisfy the primal LP as yet.

2. *Raise* some of the y_j's, either simultaneously or one-by-one.
3. Whenever a dual constraint becomes tight, *freeze* values of corresponding y's and raise value of corresponding x.
4. Repeat from *Step 2* until all the constraints become tight.

Now let us consider the primal-dual algorithm for our earlier example of vertex cover.

15.3.1 Primal-Dual Algorithm for Vertex Cover

1. Start with $x = 0$ and $y = 0$.
2. Pick any edge e for which y_e is not frozen yet.
3. Raise the value of y_e until some vertex constraint v goes tight.
4. Freeze all y_e's for edges incident on v. Raise x_v to 1.
5. Repeat until all y_e's are frozen.

Let us see an example of how this algorithm works on an instance of the vertex cover problem. We consider the following graph:

Example: Given below is a graph with weights assigned to vertices as shown in the figure and we start with assigning $y_e = 0$ for all edges $e \in E$.

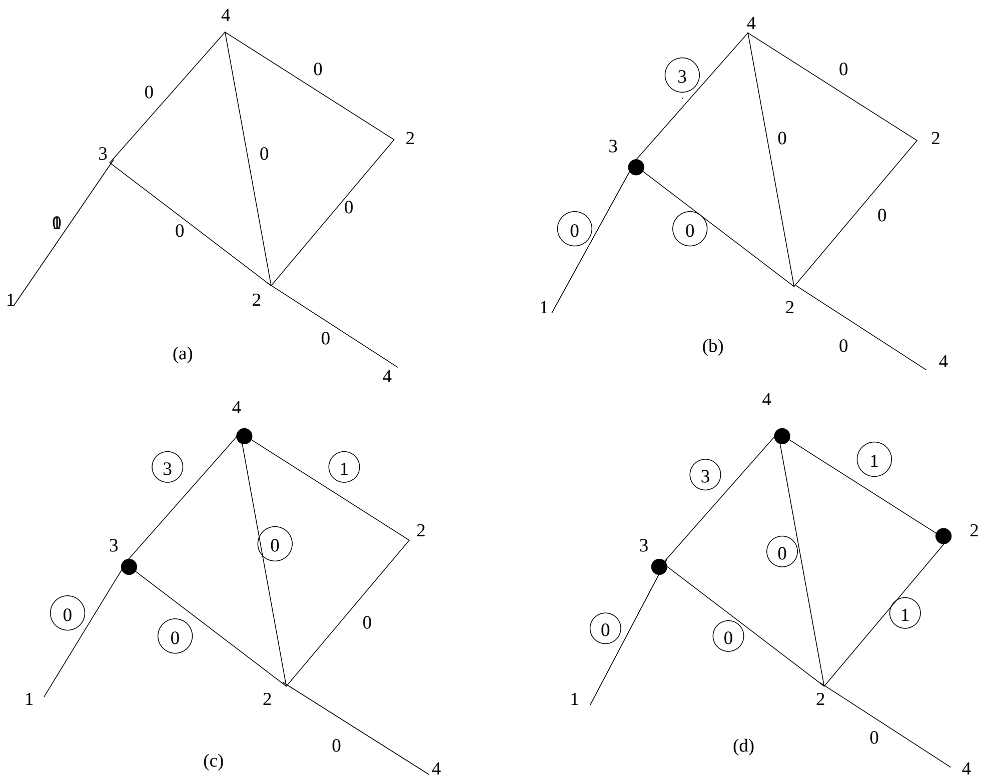

Fig 1: The primal-dual algorithm for vertex cover.

So the algorithm proceeds as shown in the figure above. In steps (a)-(d), an edge is picked for which y_e is not frozen and the value of y_e is raised until the corresponding vertex constraint goes tight. All the edges incident on that vertex are then frozen and value of x_v is raised to 1.

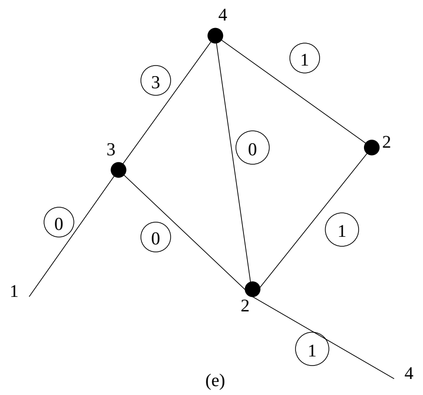

Fig 1: The vertex cover for the example graph.

When all the y_e's get frozen, the algorithm terminates. So the *Value of Primal=11* and the *Value of Dual=6*.

Hence,

$$Val_p(x) \leq 2Val_d(y)$$

and so we get the 2-approximation for our instance of vertex cover.

Lemma 15.3.1 *When the algorithm ends, x is a feasible solution for the primal and y is a feasible solution for the dual.*

Proof: That y is a feasible solution is obvious since at every step we make sure that y is a feasible solution, so it is feasible at the last when the algorithm ends. To see that x is a feasible solution we proceed by contradiction. Let us suppose it is not a feasible solution. Then there is a constraint $x_u + x_v \geq 1$ which is violated. That means both x_v and x_u are zero and thus it must be possible to raise the value of the edge between them since neither of them has a tight bound. This is a contradiction with the fact that all y_e's are frozen. ■

Lemma 15.3.2 *x and y satisfy PCS.*

Proof: This is obvious since at every step we make sure that x and y satisfy PCS. ■

Definition 15.3.3 *Let x and y be feasible solutions to the primal and dual respectively. Then we say that x and y satisfy α-approximate DCS if $\forall j, (y_j \neq 0) \rightarrow (\sum_i A_{ij}x_i \leq \alpha b_j)$.*

Lemma 15.3.4 *x and y satisfy 2-approximate DCS.*

Proof: This follows from the fact that x_v is either 0 or 1 for any v so $x_u + x_v \leq 2$ for any $e = (u, v)$. ■

The next lemma shows us why we would want an α-approximation for DCS.

Lemma 15.3.5 *Suppose x and y satisfy PCS are feasible for Primal and Dual respectively and α-approximate DCS then $Val_P(x) \leq \alpha Val_D(y)$.*

Proof: To prove this we only need to write out the sums. We have $Val_P(x) = \sum_i c_i x_i$ now since we know that x,y satisfy PCS we have that $\sum_i c_i x_i = \sum_i (\sum_j A_{ij} y_j) x_i$ by reordering the summation we get $\sum_j(\sum_i A_{ij}x_i)y_j \leq \sum_j \alpha b_j y_j = \alpha \sum_j b_j y_j = \alpha Val_D(y)$ where the $\leq$ follows from α-approximate DCS. ■

The last two lemmas then directly yield the desired result which is that our algorithm is a 2-approximation for Vertex cover.

We should also note that we never used anywhere in our analysis what order we choose our edges in or how exactly we raise the values of y_e's. A different approach might be to raise the values of all our y_e's simultaneously until some condition becomes tight. Using this approach with the graph we had earlier would give a different result. We would start by raising the values of all edges to $\frac{1}{2}$ at which point the vertex at the bottom of the diamond would become full and it's tightness condition would be met. We freeze all the edges adjecent to that vertex add it to our vertex cover and continue raising the values of the other y_e's. The next condition to be met will be the left bottom most node when we raise the value of the incoming edge to 1. After that we continue rasing the values of the remaining unfrozen edges until $1\frac{1}{2}$ when both the left and right edge of the diamond become full and we freeze all the remaining edges. The run of this algorithm with the circled numbers denoting frozen edge capacities and the emptied nodes denoting nodes in the graph cover can be seen in the following figures.

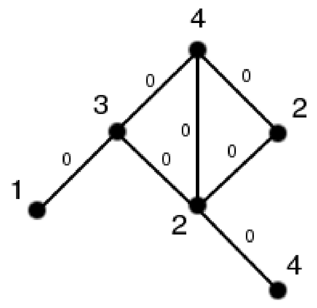

Figure 15.3.1: Step 0

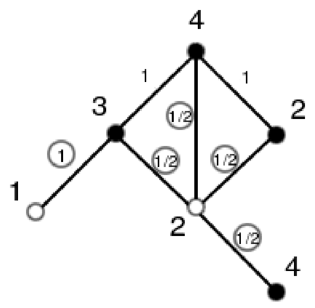

Figure 15.3.3: Step 2

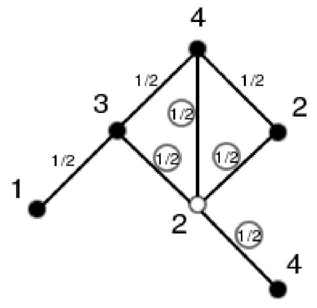

Figure 15.3.2: Step 1

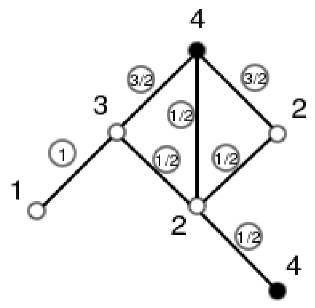

Figure 15.3.4: Step 3

Note that while this approach gives a better result in this case than the first one we used, it is not that case in general. In fact there are no known polynomial time algorithms that give a result of

$(2-\epsilon)$-approximation for any ϵ small constant.

15.4 Minimax principle

Next we will look at an application of LP-duality in the theory of games. In particular we will prove the Minimax theorem using LP-duality. First though we will need to explain some terms. By a 2-player zero sum game, we mean a protocol in which 2 players choose strategies in turn and given two strategies x and y, we have a valuation function $f(x,y)$ which tells us what the payoff for the first player is. Since it is a zero sum game, the payoff for the second player is exactly $-f(x,y)$. We can view such a game as a matrix of payoffs for one of the players.

As an example take the game of Rock-paper-scissors, where the payoff is one for the winning party or 0 if there is a tie. The matrix of winnings for player one will then be the following:

$$\begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$$

Where A_{ij} corresponds to the payoff for player one if player one picks the i-th element and player two the j-th element of the sequence $(\text{Rock}, \text{Paper}, \text{Scissors})$. We will henceforth refer to player number two as the column player and player number one as the row player. If the row player goes first, he obviously wants to minimize the possible gain of the column player. So the payoff to the row player in that case will be $min_j(max_i(A_{ij}))$. On the other hand if the column player goes first, we get the payoff being $max_i(min_j(A_{ij})$. It is clearly the case that $max_i(min_j(A_{ij})) \leq min_j(max_i(A_{ij}))$.

The Minimax theorem states that if we allow the players to choose probability distributions instead of a given column or row then equality holds or slightly more formally:

Theorem 15.4.1 *If x and y are probability vectors then $max_y(min_x(y^TAx) = min_x(max_y(y^TAx))$.*

Proof: We will only give a proof sketch. Notice that once we have chosen a strategy, if our opponent wants to minimize his loss, he will always just pick the one row which gives the best result, not a distribution. For $max_y(min_x(y^TAx))$, we wish to find a distribution over the rows such that whatever column gets picked, we get at least a payoff of t such that t is maximal. This can be formulated in terms of an LP-problem as maximizing t given the following set of constraints:

$$\forall j \quad \sum_i y_i A_{ij} \geq t$$

and

$$\sum_i y_i = 1$$

, which can be changed into a more standard form by writing the equations as

$$\forall j \quad t - \sum_i y_i A_{ij} \leq 0$$

and relaxing the second condition as

$$\sum_j y_j \leq 1.$$

On the other hand $min_x(max_y(y^T Ax))$ can be thought of as trying to minimize the loss and can thus be rewritten in terms of an LP-problem in a similar fashion as minimizing s given that

$$\sum_j x_j \leq 1$$

and

$$\forall i \quad s - \sum_j A_{ij} x_j \leq 0$$

.

In other words, we are trying to minimize the loss, no matter which column our opponent chooses. We leave it as an exercise to prove that the second problem is a dual of the first problem and thus by the Strong duality theorem, the proof is concluded. ■