LP Duality : Set Cover and Vertex Cover[1]

- *Approximation Algorithms and the Dual.* So far, we have seen one way in which linear programs have been useful in approximation algorithms design : we move to an LP relaxation, solve the LP to get a fractional solution, and then round the solution to obtain an integral approximate solution. The approximation factor is often proved by comparing with the value of the LP relaxation rather than the integer optimum. The "LP solver" is thought of as a black box. We now show how the dual of a linear program is used in approximation algorithms. The algorithms are explicit in that no LPs are (explicitly) solved; these algorithms are often called "combinatorial algorithms", and, in certain cases, can be implemented faster[2].

  There are two broad ways in which the dual is used. One, and we have already seen this in disguise, is called ***dual fitting*** where the dual is mainly used to analyze an already existing algorithm with respect to the LP relaxation. The other, and perhaps more interesting, is called a ***primal-dual algorithm*** where the dual is really used to guide the design process. In this lecture, we revisit the greedy set cover algorithm and show that the "charging argument" we did way back is dual fitting, thereby removing some of the mystery from the process. We also see a primal-dual 2-approximation algorithm for the vertex cover problem.

- *Set Cover LP and Dual Fitting.* Let's begin with the LP relaxation for the set-cover problem and also write its dual.

$$\mathsf{lp}(\mathcal{S}) := \text{minimize} \quad \sum_{j=1}^{m} c(S_j)x_j \qquad \text{(Set Cover LP)}$$
$$\sum_{j:i\in S_j} x_j \geq 1, \qquad \forall i \in U \qquad (1)$$
$$x_j \geq 0, \qquad \forall j = 1,\ldots,m \qquad (2)$$

  The dual has a variable for every constraint above. Since there is a constraint per element $i \in U$, there is a dual variable y_i for every $i \in U$.

$$\mathsf{dual}(\mathcal{S}) := \text{maximize} \quad \sum_{i\in U} y_i \qquad \text{(Set Cover Dual)}$$
$$\sum_{i\in S_j} y_y \leq c(S_j), \qquad \forall j \in [m] \qquad (3)$$
$$y_i \geq 0, \qquad \forall i \in U \qquad (4)$$

  Now, I would ask you to recall (or re-read) the analysis for the GREEDY SET COVER algorithm from long back. If you recall, the algorithm proceeds in rounds, and in each round picks a set S_j with

[1] *Lecture notes by Deeparnab Chakrabarty. Last modified : 30th Jan, 2022*
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

[2] Though it is often hard to compete with the industry level LP solvers which have been improved over decades.

minimum $c(S_j)/|S_j \cap X|$ where X is the current set of uncovered elements. In the analysis, we devised a "charge" α_i for every elements with the property that

$$\text{for all sets } S_j, \quad \sum_{i \in S_j} \alpha_i \le c(S_j) \cdot H_d, \quad \text{and} \quad \mathsf{alg} := \sum_{i \in U} \alpha_i$$

where d is the maximum size of a set (and H_d is the dth harmonic number). This immediately implies that $y_i := \frac{\alpha_i}{H_d}$ is a ***feasible*** solution to (Set Cover Dual). And therefore, what we get is

$$\frac{\mathsf{alg}}{H_d} = \sum_{i \in U} y_i \underbrace{\le}_{\text{since } y \text{ is a feasible solution to a } \textbf{\textit{maximization}} \text{ LP}} \mathsf{dual}(\mathcal{S}) \underbrace{=}_{\text{Strong Duality, but note that only Weak Duality is needed}} \mathsf{lp}(\mathcal{S})$$

which in turn proves that $\mathsf{alg} \le H_d \cdot \mathsf{lp}(\mathcal{S})$. This is a *stronger* result than what we concluded way back in the course; then, we only compared ourselves with `opt`. But now, we see that the charging argument in fact put an upper bound of H_d on the integrality gap as well.

This technique of "fitting a feasible dual solution" to charge for the algorithm's performance, is called *dual fitting*.

- *Primal-Dual Algorithm for Vertex Cover.* We now describe the primal-dual methodology with the vertex cover problem. Here is the LP relaxation for the vertex cover problem.

$$\mathsf{lp}(G) := \text{minimize} \quad \sum_{v \in V} c(v) x_v \qquad \text{(Vertex Cover LP)}$$
$$x_u + x_v \ge 1, \qquad \forall (u,v) \in E \qquad (5)$$
$$x_v \ge 0, \qquad \forall v \in V$$

We have already seen that *solving* the LP and then picking the vertices with $x_v \ge 1/2$ gives a 2-approximation. The primal-dual schema gives an explicit 2-approximation algorithm. Let's begin with the dual.

$$\mathsf{dual}(G) := \text{maximize} \quad \sum_{e \in E} y_e \qquad \text{(Vertex Cover Dual)}$$
$$\sum_{e : v \in e} y_e \le c(v), \qquad \forall v \in V \qquad (6)$$
$$y_e \ge 0, \qquad \forall e \in E$$

The dual has a variable y_e per edge of the graph (since the primal has a constraint per edge) and the objective is to maximize the total y_e's. There is a constraint per vertex v; it says the total y-value "faced" by any vertex v can't be any bigger than the cost of the vertex.

The primal-dual schema, very generally, follows the following steps (at least for a minimization problem whose dual is a maximization problem).

a. Start with a feasible dual solution, usually the all zeros solution, and an empty (primal) solution to the problem at hand.

b. Increment a subset of the dual variables till some dual constraint gets tight.

c. Taking cue from *complementary slackness* select the corresponding primal variable in the solution.

d. Repeat till either one can't raise the dual any more, or one gets a feasible primal solution; often the two occur together.

e. (Problem Dependent Step) Do some post-processing.

So, for the vertex cover problem, we start with a solution $y_e = 0$ for all $e \in E$ which is a feasible dual solution. We also start with an empty vertex cover $C = \emptyset$. Then we try to raise y_e for all $e \in E$, and we continue doing so till for some vertex v_1, the constraint (6) becomes tight. We add this v to our vertex cover solution C. Note, this vertex will be the one which minimizes $c(v)/\deg(v)$; and indeed, this will also be the first vertex the greedy algorithm would pick. The greedy algorithm, we know, can't give a $O(1)$-approximation, and so what happens next is crucial.

Once the constraint corresponding to v_1 becomes tight, we can't increase y_e for any e incident on v_1. Such edges are called *frozen* and these edges are precisely the ones that have already been covered by C. The dual growing process continues but we only now increase y_e only for unfrozen, or active, edges. And then some other vertex v_2 becomes tight. Is v_2 the second vertex the greedy algorithm would've picked? Not necessarily, right? This is how this algorithm differs from the normal greedy algorithm. We add v_2 to C, freeze all active edges incident on v_2, and continue.

We proceed till all edges become frozen and we return C. Note that since all frozen edges are indeed covered, what we return will be a feasible vertex cover. What's probably not clear is how expensive it can be.

procedure PRIMAL-DUAL VERTEX COVER($G = (V, E), c(v)$):
  $y_e \leftarrow 0$ for all $e \in E$; $C \leftarrow \emptyset$; $A \leftarrow E$; $F \leftarrow \emptyset$.
  while $A \neq \emptyset$ **do**:
    Increase y_e for each $e \in A$ till some (6) is equality for some v.
    ▷ *Indeed, you can quickly figure this v: it is the one minimizing $c'(v)/\deg_A(v)$ where $c'(v) = c(v)$ minus the y-mass v already faces, and $\deg_A(v)$ is the number of A-edges incident on v.*
    $C \leftarrow C + v$; $A \leftarrow A \setminus \partial_G(\{v\})$; $F \leftarrow F \cup \partial_G$. ▷ *All edges incident on v are not covered.*
  return (C, y).

- *Analysis.*

Theorem 1. PRIMAL-DUAL VERTEX COVER returns (C, y) with $\sum_{v\in C} c(v) \leq 2\sum_{e\in E} y_e$, and y is a feasible solution to (Vertex Cover Dual). In particular, the algorithm is a 2-approximation algorithm.

Proof. First, we note that y is a feasible solution to (Vertex Cover Dual) by design since whenever a constraint becomes tight we stop increasing any variables participating in that constraint.

Next, we observe that whenever we pick a vertex in C it is only because $c(v) = \sum_{e:v\in e} y_e$ at that point. Therefore, we get

$$\sum_{v\in C} c(v) = \sum_{v\in C}\sum_{e:v\in e} y_e = \sum_{e\in E} y_e \cdot |C \cap e|$$

But an edge e contains at most two vertices, and worst case both are in C. Thus, $|C \cap e| \leq 2$, and this proves the theorem. $\square$

Notes

References