601.435/635 Approximation Algorithms **Lecturer:** Michael Dinitz
Topic: Dual Fitting and Primal-Dual **Date:** 4/11/19
Scribe: Michael Dinitz

20.1 Introduction

Today I'm going to introduce two different but related techniques: primal-dual and dual fitting. Since these are the first "new" algorithmic technique I've introduced since tree embeddings, we're going to see it first in some relatively simple contexts, namely Set Cover. At a high level, in dual fitting we use an LP and duality *only* in the analysis – the algorithm itself has nothing to do with LPs. On the other hand, in primal-dual algorithms, our algorithm iteratively *builds* primal and dual solutions, and when it terminates we prove the approximation by bounding the gap between the primal and the dual solutions.

20.2 Set Cover

First, let's remember set cover and its LP relaxation.

Input: Universe $\mathcal{U}$, Collection $\mathcal{S} \subseteq 2^{\mathcal{U}}$, costs $c : \mathcal{S} \to \mathbb{R}^+$.

Feasible: $\mathcal{I} \subseteq \mathcal{S}$ s.t. $\bigcup_{S \in \mathcal{I}} S = \mathcal{U}$

Objective: $\min \sum_{S \in \mathcal{I}} c(S)$

20.2.1 LP for Set Cover

Primal Program:

$$\begin{aligned} \text{minimize:} \quad & \sum_{S \in \mathcal{S}} c(S) x_S & & \text{(SC-Primal)} \\ \text{subject to:} \quad & \sum_{S : e \in S} x_S \geq 1 & \forall e \in \mathcal{U} & \\ & x_S \geq 0 & \forall S \in \mathcal{S} & \end{aligned}$$

The dual is straightforward to write down. As usual coverings and packings are duals of each other. Here, the primal is a covering problem and the dual is a packing problem.

Dual Program:

$$\begin{aligned} \text{maximize:} \quad & \sum_{e \in \mathcal{U}} y_e & & \textbf{(SC-Dual)} \\ \text{subject to:} \quad & \sum_{e \in S} y_e \leq c(S) & \forall S \in \mathcal{S} \\ & y_e \geq 0 & \forall e \in \mathcal{U} \end{aligned}$$

20.3 Dual Fitting: Greedy

As mentioned, dual fitting is an analysis technique, not an algorithmic technique. So we're going to apply it to one of our favorite Set Cover algorithms: the greedy algorithm. Recall the greedy algorithm, which maximizes "bang-for-buck", or equivalently, minimized "cost per element covered". Slightly more formally, the greedy algorithm is the following:

- Initialize $\mathcal{U}' = \mathcal{U}$ and $I = \emptyset$.
- While $U' \neq \emptyset$:
  - Let $S = \text{argmin}_{T \in \mathcal{S}} \frac{c(T)}{|T \cap \mathcal{U}'}$
  - $I \leftarrow I \cup \{S\}$
  - $U' \leftarrow U' \setminus S$
- return I

Suppose at iteration t, greedy chooses S_t which covers n_t new elements and costs $c(S_t)$. Suppose greedy first covered e at time t. Set

$$y'_e = \frac{c(S_t)}{n_t}.$$

These y' values may not be feasible for the dual, but it is easy to see that

$$\sum_{S \in I} c(S) = \sum_{e \in \mathcal{U}} y'_e,$$

since we just split the cost of every set we choose between the elements that it covers.

Now we'll rescale y' to get y, where $y_e = y'_e / H_n$. We're going to prove the following lemma.

Lemma 20.3.1 *y is a feasible solution for the dual.*

Before we prove this lemma, let's use it to prove the main theorem.

Theorem 20.3.2 *Greedy is H_n-approximation.*

Proof: Let I^* be the optimal solution, and let x^* be the vector obtained by setting $x_S = 1$ if $S \in I^*$ and $x_S = 0$ otherwise. Then

$$\begin{aligned}
\text{greedy} &= \sum_{S \in I} c(S) = \sum_{e \in \mathcal{U}} y'_e = H_n \sum_{e \in \mathcal{U}} y_e \\
&\leq H_n \sum_{S \in \mathcal{S}} c(S) x^*_S && \text{(by Lemma 20.3.1 and weak duality)} \\
&= H_n \cdot \text{OPT}.
\end{aligned}$$

Note that the key step here, where we use weak duality, works because Lemma 20.3.1 implies that y is feasible, and by weak duality we know that every feasible dual solution has objective that is at most most the objective of any feasible primal solution (x^* in particular). ■

Now let's prove Lemma 20.3.1.

Proof of Lemma 20.3.1: To show that y is feasible for the dual, we'll pick an arbitrary $S \in \mathcal{S}$ and show that the dual constraint for S is satisfied. So let $S \in \mathcal{S}$. Suppose at the beginning of iteration k, a_k elements in S are uncovered. Let $A_k \subseteq S$ be the elements first covered in iteration k, hence

$$|A_k| = a_k - a_{k+1}.$$

The greedy algorithm count have picked S in iteration k, in which case the average cost would be $\frac{c(S)}{a_k}$. Since the greedy algorithm always chooses the set that minimizes the average cost, this implies that

$$y'_e \leq \frac{c(S)}{a_k}$$

for all $e \in A_k$. Let ℓ denote the total number of iterations of the algorithm, and consider the dual constraint for S (which is $\sum_{e \in S} y_e \leq c(S)$).

$$\begin{aligned}
\sum_{e \in S} y_e &= \frac{1}{H_n} \sum_{e \in S} y'_e = \frac{1}{H_n} \sum_{k=1}^{\ell} \sum_{e \in A_k} y'_e \leq \frac{1}{H_n} \sum_{k=1}^{l} \sum_{e \in A_k} \frac{c(S)}{a_k} \\
&= \frac{1}{H_n} \sum_{k=1}^{l} (a_k - a_{k+1}) \frac{c(S)}{a_k} = \frac{c(S)}{H_n} \sum_{k=1}^{l} \frac{a_k - a_{k+1}}{a_k} \\
&\leq \frac{c(S)}{H_n} \sum_{k=1}^{|S|} \frac{1}{i} = \frac{H_{|S|}}{H_n} c(S) \leq c(S).
\end{aligned}$$

Thus y is a feasible dual solution as claimed. ■

20.4 Primal-Dual

20.4.1 General Technique

Before we do primal-dual for Set Cover, let's talk about the general "schema".

1. Write down primal (min) and dual (max)
2. Start with $x = \vec{0}$ and $y = \vec{0}$. (x is primal infeasible, y is dual feasible.)
3. Until x feasible:
   (a) Increase y until some dual constraint become tight (maintaining feasibility of $\vec{y}$).
   (b) Select some of the tight dual constraints, increase corresponding primal variables integrally.
   (c) "Freeze" dual variables in tight dual constraints.

For analysis, prove $c^\top x \leq \alpha \cdot b^\top y$ for some α. Since x and y are a feasible primal/dual pair, this implies that the cost of x is at most $\alpha \cdot OPT$.

We can also think of this is drawing inspiration from complementary slackness. We always have a feasible dual solution, and when some dual constraint becomes tight, we set the associated primal variable to be nonzero (in fact, to be an integer). Since we require that x is integral we do not in fact have an optimal primal/dual pair, so complementary slackness is not satisfied, but we can think of this algorithm as taking inspiration from complementary slackness.

20.4.2 Primal-Dual for Set Cover

- $y = \vec{0}, \quad I = \emptyset$.
- While there exists $e \in \mathcal{U}$ such that $e \notin \bigcup_{S \in I} S$:
  - Increase y_e until there is some $S \notin I$ with $e \in S$ such that $\sum_{e' \in S} y_{e'} = c(S)$
  - Add S to I
- Return I.

It must be noted that in the first line of the while loop, we don't actually need to increase y_e continuously. Instead, we directly solve for the minimum needed increase:

$$\min_{\substack{S \notin \mathcal{I} \\ e \in S}} \left(c(S) - \sum_{e' \in S} y_{e'} \right).$$

Let $f = \max_{e \in \mathcal{U}} |\{S \in \mathcal{S} : e \in S\}|$ denote the maximum number of sets that any element is in. This is called the *frequency* of the instance.

Theorem 20.4.1 *PD is an f-approximation for* Set Cover.

Proof: Note that y is always feasible for the dual. To see this, consider the dual constraint for some set S. If this dual constraint never became tight, then it's clearly not violated. If it does become tight at some point, then we either include it in I or, if we don't, it's because all elements of S are already covered (this can happen if there are multiple dual constraints become

tight simultaneously). But in either case, all elements of S become covered, so we never increase any of the dual variables again and thus the constraint never gets gets violated.

Now it becomes straightforward to analyze the cost of the algorithm.

$$\begin{aligned}
\text{PD} &= \sum_{S \in I} c(S) \\
&= \sum_{S \in I} \sum_{e \in S} y_e && \text{(dual constraint of } S \text{ is tight)} \\
&= \sum_{e \in S} y_e \cdot |\{S \in I : e \in S\}| && \text{(switch order of summations)} \\
&\leq f \sum_{e \in S} y_e && \text{(definition of frequency)} \\
&\leq f \cdot \text{OPT} && \text{(by weak duality)}
\end{aligned}$$

■

20.5 Shortest $s - t$ Path

Let's see another easy example of primal-dual algorithms, this time for the $s - t$ shortest path problem. Let's assume al edge lengths are positive, so we know that we could always run Dijkstra's algorithm. But let's try to design a primal-dual algorithm. As it turns out, we'll essentially re-derive Dijkstra's algorithm!

Input:

- Graph $G = (V, E)$
- $s, t \in V$
- $c : E \mapsto \mathbb{R}^+$

Moreover, let $\mathcal{S} = \{S \subseteq V : s \in S, t \notin S\}$.

Primal Program:

$$\begin{aligned}
\text{minimize:} \quad & \sum_{e \in E} c(e) x_e && && \textbf{(SP-Primal)} \\
\text{subject to:} \quad & \sum_{e \in \delta(S)} x_e \geq 1 && \forall S \in \mathcal{S} \\
& x_e \geq 0 && \forall e \in E
\end{aligned}$$

Dual Program:

$$\begin{aligned} \text{maximize:} \quad & \sum_{S \in \mathcal{S}} y_S & & \textbf{(SP-Dual)} \\ \text{subject to:} \quad & \sum_{\substack{S \in \mathcal{S} \\ e \in \delta(S)}} y_S \leq c(e) & \forall e \in E \\ & y_S \geq 0 & \forall S \in \mathcal{S} \end{aligned}$$

PD Algorithm:

- $y = \vec{0}, \quad F = \emptyset$.
- While no $s-t$ path in (V, F):
  - Let C be the component of (V, F) that contains s
  - Increase Y_C until there is some $e \in \delta(C)$ such that
$$\sum_{\substack{S \in \mathcal{S} \\ e \in \delta(S)}} y_S = c(e).$$
  - Add e to F
- Return $s-t$ path P in F.

Before we prove correctness (that this does return the minimum $s-t$ path), we need to prove two easy lemmas.

Lemma 20.5.1 *At all steps in the PD Algorithm, F is a tree containing s.*

Proof: We use induction. This is clearly true initially, when $F = \emptyset$. Now suppose that it's true at some point in the algorithm, and the algorithm adds $e \in \delta(C)$ to F. By induction, F was a tree containing s, and e has exactly one endpoint in C. Thus F is still a tree containing s. ■

Lemma 20.5.2 *If $y_S > 0$, then $|P \cap \delta(S)| = 1$.*

Proof: Suppose that this is false, and let S be a set such that $y_S > 0$ and $|P \cap \delta(S)| > 1$. Then there must be a subpath P' of P such that the only vertices of P' in S are the start and ending vertices of P'. Since the algorithm increased y_S, when we did this F was a tree which spanned S (due to the Lemma 20.5.1). But this means that $F \cup P'$ contains a cycle, which is a contradiction since $F \cup P'$ is a subset of the final set F which by Lemma 20.5.1 is a tree. Thus $|P \cap \delta(S)| = 1$. ■

Theorem 20.5.3 *PD finds a shortest $s-t$ path.*

Proof: Because we only add an edge when its dual constraint becomes tight, we have that

$$\begin{aligned} \sum_{e \in P} c(e) &= \sum_{e \in P} \sum_{S : e \in \delta(S)} y_S = \sum_{S \in \mathcal{S}} y_S \cdot |P \cap \delta(S)| \\ &= \sum_{S \in \mathcal{S}} y_S \leq \text{OPT}, \end{aligned}$$

where we used Lemma 20.5.2 to bound $y_S \cdot |P \cap \delta(S)|$ and weak duality in the final step. Thus $\sum_{e \in P} c(e) = \text{OPT}$. ■