CS 270 Combinatorial Algorithms and Datastructures **Spring 2017**

The Online Cover-Pack Framework

Scribe: Antares Chen, Steven Lin, Eric Sheng

1 Introduction

In CS270, we utilized the dual of a problem to design efficient approximation algorithms. Such technique has been very powerful, allowing us to create algorithms for congestion routing via tolling, bipartite matching via price functions, and facility location via its dual LP. In general, this method first reduces a combinatorial problem to its Linear Program (LP) form, then grows its primal variables in relation with its dual.

However, the problems we have studied have largely been *offline*: knowledge of the entire instance is given before executing the algorithm. With the growth of large-scale data and computing, it is now important to consider problems like streaming where the instance is given incrementally, i.e. *online*. Thus, we present Buchbinder and Naor's generalization of the primal-dual method to the online setting using *cover-pack* LPs.

Our presentation differs as we allow the set-cover problem and a method named dual-fitting to guide our analysis. Section 1 reviews set-cover and provides a dual-fitting derivation of its approximation. Section 2 introduces online set-cover and constructs the general algorithm. Doing so, we note the algorithm actually constitutes a framework and can be applied beyond set-cover. Thus, section 3 demonstrates this by apply the algorithm to approximate congestion routing.

2 Offline Set-Cover

Consider the set-cover problem: provided items $\mathcal{X} = \{e_1, e_2, \ldots, e_n\}$ and a collection of subsets of items $\mathcal{S} = \{s_1, s_2, \ldots, s_m\}$ where $s_j \subseteq \mathcal{X}$ is associated with a cost c_s, find a collection of sets $\mathcal{C} \subseteq \mathcal{S}$ (called a cover) with minimal total cost where the union of its elements is $\mathcal{X}$.

Set-cover has a well known greedy $O(\log n)$-approximation scheme, classically derived using a potential function. However, we can use set-cover's LP reduction to provide a much cleaner analysis. We first derive Set-Cover's LP reduction, which we call the *cover-pack programs*. We then provide a dual-fitting analysis of the greedy approximation algorithm, highlighting design choices that will be critical to analyzing the general online algorithm.

2.1 The Cover-Pack Linear Programs

Designate $x_s \in \{0, 1\}$ as decision variables for if s should be included in our final cover. We wish to minimize total cost $\sum_{s \in \mathcal{C}} c_s$ subject to the constraint that all elements are included in some set in the final cover. Relaxing the integrality constraint to $x_s \geq 0$ gives the primal *covering LP* which is the LP for set-cover.

$$
\begin{array}{llll}
P : \min \displaystyle\sum_{s \in \mathcal{S}} c_s x_s & & D : \max \displaystyle\sum_{e \in \mathcal{X}} y_{e_i} & \\
\text{s.t.} \displaystyle\sum_{s : e_i \in s} x_s \geq 1 & \forall i \in [n] & \text{s.t.} \displaystyle\sum_{e_i \in s} y_{e_i} \leq c_s & \forall s \in \mathcal{S} \\
x_s \geq 0 & \forall s \in \mathcal{S} & y_{e_i} \geq 0 & \forall i \in [n]
\end{array}
$$

The dual LP, displayed next to the primal, is designated as the *packing LP* and its simple to see why. We can consider each y_{e_i} as the charge associated with each element e_i. Our goal is then to *pack in* as much total profit as possible such that the cumulative charge across elements in one set does not exceed the cost of that set. Now removing the context of set-cover, these two programs together are in general known as the *cover-pack LPs*.

2.2 Set-Cover via Dual Fitting

The intuition behind the $O(\log n)$-approximation for set-cover is to build the cover by choosing the set that maximizes cost per yet uncovered elements.

Algorithm 2.2:
Let $\mathcal{U}$ denote the set of uncovered elements and initialize $\mathcal{C} = \emptyset$. While $\mathcal{U} \neq \emptyset$ do:

1. Choose the set $s = \text{argmin}_{s \in \mathcal{S}} \left\{ \frac{c_s}{|\mathcal{U} \cap s|} \right\}$
2. Add $s \to \mathcal{C}$ and set $x_s = 1$
3. For each $e_i \in \mathcal{U} \cap s$: increment $y_{e_i} = \frac{c_s}{|\mathcal{U} \cap s|}$

The classical derivation of this algorithm's approximation ratio uses a potential function. However, using the LP reduction and a technique called *dual-fitting* leads to a more structured analysis. We briefly sketch the proof here. The proof relies on three conditions:

1. Algorithm 2.2 produces a solution that is primal feasible.
2. For each iteration, change in the primal objective lower-bounds change in the dual objective: $\Delta P' \leq \Delta D'$
3. Each packing constraint in the dual program is violated by at most $O(\log n)$.

Condition (1) allows us to argue, via weak duality, that $D \leq P'$ for any feasible dual solution D. However, D' is not feasible, but *bounded infeasible* which then by condition (3) we can claim $\frac{D'}{c \cdot \log n}$ is a feasible objective for some constant c. Finally by condition(2), $P' \leq D'$ thus we have:

$$\frac{D'}{c \cdot \log n} = D \qquad \implies \qquad D' = D(c \cdot \log n) \qquad \implies \qquad D \leq P' \leq O(\log n) \cdot D$$

Immediately, $O(\log n)$ is the approximation ratio. Dual-fitting provides a well-structured framework because it relies on showing three intuitive properties that allow us to *fit* one objective value within another. At its heart, the argument relies upon weak-duality since that is the principle which ties the proof together.

3 Online Set-Cover

Let's now lift set-cover to the online setting. We are still given items $\mathcal{X}$ and sets $\mathcal{S}$. However, we will only need to cover a subset of elements $\mathcal{X}' \subseteq \mathcal{X}$ whose members are revealed across discrete time-steps. Additionally, once a set is selected, it *cannot* be un-selected.

This formulation provides for a clear analogue in the LP reduction. When an element $e_{i'} \in \mathcal{X}'$ comes online, a primal constraint $\sum_{s : e_{i'} \in s} x_s \geq 1$ is revealed along with its corresponding dual variable $y_{e_{i'}}$. However, the arrival of $y_{e_{i'}}$ updates all dual constraints $\sum_{e_i \in s} y_{e_i} \leq c_s$ where $e_{i'} \in s$. This means that while primal constraints are revealed one-by-one, dual constraints are all simultaneously updated as the algorithm progresses through time. Finally for any algorithm, we may only increase the value of the primal and dual variables over time as decisions in the previous time-step cannot be undone.

3.1 An Online Primal-Dual Algorithm

We opt to solve the fractional relaxation of online set-cover, noting that one can construct an integral solution via a randomized rounding scheme. Before deriving the algorithm, we first assume that $c_s \geq 1$ for any $s \in \mathcal{S}$ and define $\Delta_1 = \max_{i \in [n]} |\{s \in \mathcal{S} : e_i \in s\}|$ which is just the maximum frequency of any element.

Now the relationship between the combinatorial formulation of online set-cover and its LP reduction motivates us to look for an LP based algorithm. Since all LP variables must increase monotonically, we need to determine a precise schedule of growth to achieve the tightest objective value. Suppose we increase the dual variable continuously, then intuitively, we are increasing the amount of money willing to be charged for $e_{i'}$.

For each of the sets s where $e_{i'} \in s$, it's sensible to increase the rate of their inclusion in the final cover relative to how much they cost: $\frac{x_s}{c_s}$. However, the rate will always be 0 since x_s is initially 0. Consequently, we fix this by adding a perturbation factor of $\frac{1}{\Delta_1}$. This is exactly the "primal-dual" like update we present in the main algorithm and interestingly, the analysis shows the perturbation factor driving the entire cost!

Algorithm 3.1
Upon arrival of new primal constraint $\sum_{s:e_{i'} \in s} x_s \geq 1$ and corresponding $y_{e_{i'}}$ while $\sum_{s:e_{i'} \in s} x_s < 1$ do:

1. Continuously increase dual variable $y_{e_{i'}}$
2. For each x_s such that $e_{i'} \in s$, increase x_s subject to the following:

$$\frac{\partial x_s}{\partial y_{e_{i'}}} = \frac{1}{c_s}\left(x_s + \frac{1}{\Delta_1}\right)$$

Modulo the context of set-cover, Δ_1 is actually just the matrix ℓ_1-norm of the constraints. This hints that the algorithm can be applied and manipulated in a broader sense – which it can. One can imagine applying this technique to an arbitrary combinatorial problem so long as we can find:

1. A covering/packing LP reduction for the problem.
2. A, hopefully intuitive, update for primal variable x subject to the rate $\frac{\partial x}{\partial y}$.

Indeed, Algorithm 3.1 is an example of a broader framework, and to this point, we will later demonstrate how to apply this to solve a congestion routing problem.

3.2 Competitive Analysis and Approximation Bounds

It is unclear how to analyze the algorithm's approximation ratio in the online setting because classical techniques are hard to apply. Simply applying worst-case analysis could lead us to believe an algorithm is arbitrarily inaccurate since the uncertainty introduced by partial knowledge of an instance allows for one to always construct an arbitrarily inapproximable instance.

The key insight is to compare an online algorithm to the best offline analogue. Suppose I is an input sequence to an online algorithm $\mathcal{A}$ and $\mathcal{A}(I)$ is its associated cost. Let $\mathrm{OPT}(I)$ be the cost incurred by the optimal algorithm in hindsight. That is the algorithm has access to the entire input I, hence why it may be considered as offline. We aim to show an algorithm is k-competitive defined as follows.

Definition 1 (k-competitive)**.** *$\mathcal{A}$ is k-competitive if for every input sequence I, the following holds.*

$$\mathcal{A}(I) \leq k \cdot \mathrm{OPT}(I) + \alpha$$

We refer to k as the competitive ratio and α is some constant additive term. The framework that analyzes algorithms under this metric is called Competitive Analysis.

We now provide the competitive ratio for Algorithm 3.1. Note how closely it resembles the sketch of the offline set-cover approximation algorithm's ratio.

Theorem 2. *Algorithm 3.1 produces a fractional covering solution that is feasible and $O(\log \Delta_1)$-competitive.*

Proof. Let P and D be the objective values of the primal (covering) and dual (packing) solutions produced by the algorithm. We first prove three claims:

1. The algorithm produces a primal feasible solution
2. For each item $e_{i'}$ revealed, we have $\frac{\partial P}{\partial y_{e_{i'}}} \leq 2\frac{\partial D}{\partial y_{e_{i'}}}$
3. Upon termination, each dual constraint is violated by at most a $O(\log \Delta_1)$ factor.

Proof of 1: notice that for each time step, the algorithm always increases x_s until the constraint given is satisfied. Because all variables are monotonically increasing, the algorithm must terminate with a feasible primal solution.

Proof of 2: first observe that $\frac{\partial D}{\partial y_{e_{i'}}} = 1$. Now we bound $\frac{\partial P}{\partial y_{e_{i'}}}$.

$$\frac{\partial P}{\partial y_{e_i}} = \sum_{s \in \mathcal{S}} c_s \frac{\partial x_s}{\partial y_{e_i}} = \sum_{s \in \mathcal{S}} c_s \left[\frac{1}{c_s} \left(x_s + \frac{1}{\Delta_1} \right) \right] = \sum_{s \in \mathcal{S}} \left(x_s + \frac{1}{\Delta_1} \right) \leq 2$$

The last bound follows from $\sum_{s \in \mathcal{S}} x_s < 1$ since the constraint is primal constraint is unsatisfied and $\sum_{s \in \mathcal{S}} \frac{1}{\Delta_1} \leq 1$ since $|\mathcal{S}| \leq \Delta_1$. Thus $\frac{\partial P}{\partial y_{e_i}} \leq 2\frac{\partial D}{\partial y_{e_i}}$.

Proof of 3: we wish to bound the total change in $y_{e_{i'}}$ for any $e_{i'}$. Let the $y_{e_{i'}}^{(t)}$ and $x_s^{(t)}$ be the value of each variable after executing the algorithm after the arrival of the t-th item (constraint). As the change in x_s with respect to y_{e_i} is given by $\frac{\partial x_s}{\partial y_{e_i}} = \frac{1}{c_s}\left(x_s + \frac{1}{\Delta_1}\right)$. We can solve a first order differential equation to obtain:

$$y_{e_i} = c_s \ln \left(x_s + \frac{1}{\Delta_1} \right) + C$$

Now suppose an arbitrary $e_{i'}$ arrives at time t. As $y_{e_{i'}}$ is only raised during step t, we have the following.

$$y_{e_{i'}} = y_{e_{i'}}\Big|_{t}^{t-1} = c_s \ln \left(x_s + \frac{1}{\Delta_1} \right)\Big|_{t-1}^{t} = c_s \ln \left(\frac{x_s^{(t)} + 1/\Delta_1}{x_s^{(t-1)} + 1/\Delta_1} \right)$$

Assuming the algorithm terminates after T steps, we sum over both the sides.

$$\sum_{e_i : e_i \in S} y_{e_i} = c_s \sum_t \ln \left(\frac{x_s^{(t)} + 1/\Delta_1}{x_s^{(t-1)} + 1/\Delta_1} \right) = c_s \ln \left(\frac{x_s^{(T)} + 1/\Delta_1}{x_s^{(0)} + 1/\Delta_1} \right)$$

Observe that $x_s^{(T)} - x_s^{(0)} \leq 1$ since all covering constraints are upper-bounded by 1 and the algorithm will never increase any x_s above 1. This gives the following.

$$\sum_{e : e \in S} y_e \leq c_s \ln \left(\frac{1 + 1/\Delta_1}{1/\Delta_1} \right) \leq c_s \ln(1 + \Delta_1) \in O\big(\log \Delta_1 \big)$$

It follows that the dual constraints are only ever violated by a factor of $O(\log \Delta_1)$. Now the algorithm initializes $P = D = 0$. By condition (2), we know that $\Delta P \leq 2\Delta D$. By condition (3) we have $P \leq O(\log n) \cdot D$. However, condition (1) implies weak duality holds thus $D \leq P \leq O(\log n) \cdot D$ as required. □

4 Online Routing

We now demonstrate how to apply ideas behind algorithm 3.1 to solve other combinatorial optimization problems. Note the routing problem presented here differs slightly from lecture. We receive a number of requests $r_i = (s_i, t_i)$ and must choose whether or not to service unitary flow along a simple path (s_i, t_i) path. The total flow along a single edge cannot exceed its capacity c_e and our goal is to service as many requests as possible. In the fractional version, we can send any amount of flow $f = [0, 1]$ along any number of paths. The LP for the version we present and its dual are given below on the right and left respectively.

$$P : \min \sum_{e \in E} c_e x_e + \sum_{r_i} z_{r_i}$$

$$\text{s.t.} \sum_{e \in p} x_e + z_{r_i} \geq 1 \qquad \forall p \in \mathcal{P}_{r_i} \ : \ \forall r_i$$

$$D : \max \sum_{r_i} \sum_{p \in \mathcal{P}_{r_i}} f(r_i, p)$$

$$\text{s.t.} \sum_{p \in \mathcal{P}_{r_i}} f(r_i, p) \leq 1 \qquad \forall r_i$$

$$\sum_{(r_i, p) : e \in p} f(r_i, p) \leq c_e \qquad \forall e \in E$$

$\mathcal{P}_{r_i}$ denotes the set of all paths from s_i to t_i. We use the convention that the primal is a minimization problem, so the actual routing LP is formulated as the dual. In the online version, we receive a new request r_i at each round, updating a column in the dual constraint matrix and a row in the primal. Each time, we must satisfy these new constraints and assign $f(r_i, p)$. Let n be the number of vertices. We present a dual-fitting algorithm that sustains an $O(\log n)$-competitive ratio.

Algorithm 4: On new route request $r_i = (s_i, t_i)$, while there exists $p \in \mathcal{P}_{r_i}$ such that $\sum_{e \in p} x(e) < 1$, do:

1. Route the request on p and update $f(r_i, p) \leftarrow 1$
2. $z_{r_i} \leftarrow 1$
3. For each $e \in p$: let $|p|$ be the length of path p. Update x_e according to:

$$x_e \leftarrow x_e \left(1 + \frac{1}{c_e}\right) + \frac{1}{c_e \cdot |p|}$$

This problem is a packing problem, as we want to pack as many services as possible given the constraints. Consequently, this algorithm is very similar to the previous algorithm for cover-pack. Even though this algorithm provides an explicit update to primal variable x_e, we note this is a purely cosmetic difference. One can still define an implicit update relative to $\frac{\partial x_e}{\partial f(r_i, p)}$ and solve a differential equation to derive this algorithm.

Theorem 3. *Algorithm 4 returns a dual solution that is $O(\log n)$-competitive.*

Proof. Again this proof will be very similar to that of offline set-cover. Let P' and D' be the primal and dual objective values the algorithm returns. We state three claims:

1. The primal is feasible.
2. For each request, $\Delta P' \leq 3\Delta D'$.
3. Each dual constraint is violated by an $O(\log n)$ factor.

Proof of 1: Follows by the algorithm's design.

Proof of 2: Consider the change in the primal objective, $\Delta P'$. We know the following.

$$\Delta P' = \sum_{e \in E} c_e \Delta x_e + \sum_{r_i} \Delta z_{r_i}$$

where $\Delta x_e = \frac{x_e}{c_e} + \frac{1}{c_e|p|}$ by the update. Plugging this into the left sum, we find below.

$$\sum_{e \in E} c_e \Delta x_e = \sum_{e \in p} c_e \left(\frac{x_e}{c_e} + \frac{1}{c_e |p|} \right) = \sum_{e \in p} \left(x_e + \frac{1}{|p|} \right) \leq 2$$

The last equality follows from the primal constraint being infeasible. Now we update a single z_{r_i} to 1, thus $\sum_{r_i} \Delta z_{r_i} \leq 1$. Putting everything together, we find $\Delta P' \leq 2 + 1 = 3$. Now consider $\Delta D'$: $f(r_i, p)$ is updated to 1, so the change in the dual objective is simply 1. We have as required $\Delta P' \leq 3 \Delta D'$.

Proof of 3: We first prove by induction that

$$x_e \geq \frac{1}{n} \left(\left[1 + \frac{1}{c_e} \right]^{\sum_{(r_i, p): e \in p} f(r_i, p)} - 1 \right)$$

Initially, this is true since all of our variables are zero. Next consider a single pass of our algorithm in which we set $f(r_i, p) \leftarrow 1$. Let x'_e denote the new value of x_e. We have as follows:

$$\begin{aligned} x'_e &= x_e \left(1 + \frac{1}{c_e} \right) + \frac{1}{c_e |p|} \\ &\geq \frac{1}{n} \left(\left[1 + \frac{1}{c_e} \right]^{\sum_{(r'_i, p'): e \in p', (r'_i, p') \neq (r_i, p)} f(r'_i, p')} - 1 \right) \left(1 + \frac{1}{c_e} \right) + \frac{1}{n c_e} \\ &= \frac{1}{n} \left(\left[1 + \frac{1}{c_e} \right]^{\sum_{(r'_i, p'): e \in p'} f(r'_i, p')} - 1 \right) \end{aligned}$$

where we use $\max_{p \in \mathcal{P}_{r_i}} |p| < n$ and an application of the inductive hypothesis in the second line. Consider, we never update any $x_e \geq 1$, since any constraint with x_e would then be satisfied. Furthermore, $c_e, |p| \geq 1$. Plugging in $x_e = c_e = |p| = 1$ into our update, we get:

$$1(1+1) + 1 = 3 \geq x_e \geq \frac{1}{n} \left(\left[1 + \frac{1}{c_e} \right]^{\sum_{(r_i, p): e \in p} f(r_i, p)} - 1 \right)$$

Some algebra yields the required.

$$\sum_{(r_i, p): e \in p} f(r_i, p) \leq c_e \log(3n + 1) = c_e \cdot O(\log n)$$

To tie the proof together, let P^* be the optimal primal objective and D be any feasible dual objectives. Note that this solution is not a feasible dual solution, and in particular, is not guaranteed to obey the capacity constraints. We can achieve a feasible dual solution $D = \frac{D'}{O(\log n)}$ by dividing each dual variable by $O(\log n)$ to make it feasible. Using weak duality and (2), we find this bound:

$$\frac{P^*}{O(\log n)} \leq \frac{D'}{O(\log n)} = D \leq P^*$$

Thus, the dual is a feasible $O(\log n)$-competitive solution. □

An issue is that we have an exponential number of paths so we must seemingly check an exponential number of constraints each round. However, this can be circumvented by finding the shortest path from s_i to t_i using Dijkstra's algorithm and using it to check primal constraints for every path in $\mathcal{P}_{r_i}$ simultaneously.