CS 294-128: Algorithms and Uncertainty	Lecture 4	Date: September 11, 2016
Instructor: Nikhil Bansal		Scribe: Daniel Li

In this lecture we will apply the online primal-dual method to the paging problem and introduce the topic of Dual-Fitting for online problems.

1 Paging Problem With Primal-Dual Framework

In the online paging problem there is a cache of size k and an input sequence $p_1, p_2, \ldots$, from a set of pages $p \in [n]$. If the currently requested page p_t is already in the size-k cache no cost is incurred, otherwise we fetch this page, possibly evicting another one. The goal is to minimize the number of page evictions.

1.1 Primal Formulation

We first consider a covering formulation for the problem. The LP is not strictly covering and it has box constraints (of the type $x_i \leq 1$), but as we shall see this will not cause any problems with the primal dual framework. For more on the PD method with box constraints we refer to [4]

We first give some notation. Let p_t denote the page requested at time t. Wlog we assume that the cache has pages $1, \ldots, k$ initially. Let $x(p, i)$ be a variable in $[0, 1]$ that indicates whether page p is evicted from our cache between the i-th and $i+1$-th time it is requested. As we will consider fractional solutions, $x(p, i)$ could be viewed as the probability that page p was evicted during that period. Let $t(p, i)$ denote the time for the i-th request for p. Let $r(p, t)$ denote the number of times page p is requested until time t, inclusive of t.

In the online setting, we will think of a new variable $x(p, i)$ arriving at time $t(p, i)$. Initially it has value 0, which rises until it reaches 1 or until time $t(p, i+1)$. Let T the time of the final request.

We can thus write the objective function as

$$\min \sum_{p \in [n]} \sum_{i=1}^{r(p,T)} x(p, i)$$

which can be viewed as the total number of evictions over the input sequence. Next we have the constraints. At any time t, our cache can only hold k-items. One of these must be p_t, so there is only room for another $k-1$ pages. So at least $(n-1)-(k-1) = n-k$ pages from $[n] \setminus \{p_t\}$ must not be in the cache at time t. We can write this as follows.

$$\sum_{p \neq p_t} x(p, r(p, t)) \geq n - k \quad \forall t \in T \tag{1}$$

We also enforce that $0 \leq x(p, i) \leq 1$, as it does not make sense to evict a page to extent more than 1. These are called box-constraints. Unlike in set cover, these box constraints $x(p, i)$ are important here to ensure that the LP cannot cheat.

1.2 Dual Formulation

The dual is given by the following. There is a variable y_t for each time t, and a variable $z(p,i)$ for each box constraint.

$$\text{Dual:} \quad \max \sum_t y(t)(n-k) - \sum_p \sum_{i=1}^{r(p,T)} z(p,i)$$

Corresponding to the primal variable $x(p,i)$, we have the following constraint.

$$\sum_{t \in I(p,i)} y_t - z(p,i) \leq 1 \qquad \forall p, i \in [r(p,t)]$$

where $I(p,i)$ is the interval $(t(p,i), t(p,i+1))$.

1.3 The Fractional primal dual algorithm

We now define the algorithm. It is convenient to define it in a continuous way. The framework is exactly as before. When request for p_t arrives, we create a new variable $x(p, r(p,t))$ and get the new covering constraint (1) at time t. Let $y(t)$ be the corresponding dual variable $y(t)$. We raise the primal variables and $y(t)$ until the primal constraint at time t is satisfied. If some variable $x(p, r(p,t))$ reaches 1, we freeze it at 1 and increase the corresponding variable $z(p, r(p,t))$ at the same rate $y(t)$.

Formally, while (1) is unsatisfied:
Raise $y(t)$ by dy.

(a) If $x(p, r(p,t)) < 1$, increase $x(p, r(p,t))$ as:

$$\frac{d(x(p, r(p,t)))}{dy} = x(p, r(p,t)) + \eta$$

where η will be chosen to be $1/k$.

(b) If $x(p, r(p,t)) = 1$, it does not increase anymore and we increase $z(p, r(p,t))$ by $dz(p, r(p,t)) = dy$.

1.4 Algorithm Analysis

We will show that for $\eta = 1/k$, this algorithm is $O(\log k)$-competitive. As previously, it suffices to show the following.

1. The primal solution is *feasible*.
2. The increase in primal and dual cost satisfies $\Delta 2 \leq c \Delta D$ where P is the primal and D is the dual,
3. Dual solution is feasible up to a factor of $O(\log k)$.

The first condition follows by design as the variables are increased until the constraints is satisfied.

Lemma 1 *At each infinitesimal step, $\Delta P \leq 2\Delta D$ where P is the primal and D is the dual.*

Proof: Consider some infinitesimal step and let S be the set of pages p where $x(p, r(p,t)) < 1$, exclusive of the current page, p_t. For convenience, we drop $r(p,t)$ from the notation as time is fixed to t.

As $x(p_t) = 0$ throughout time t, the number of pages p with $x(p) = 1$ is $|[n] \backslash S \backslash p_t| = n - |S| - 1$. For each of these variables $z(p)$ also rises by dy. So

$$\Delta D = (n-k)dy - (n - |S| - 1)dy = (|S| - k + 1)dy$$

Similarly, ΔP is given by

$$\begin{aligned}\Delta P = \sum_{p \in S} dx(p) = \sum_{p \in S} (x(p) + \eta)dy = (\sum_{p \in [n] \backslash p_t} x(p) - \sum_{p \in [n] \backslash p_t \backslash S} x(p))dy + (|S|\eta)dy \\ < (n-k)dy - (n - |S| - 1)dy + (|S|\eta)dy = (|S| - k + 1)dy + (|S|/k)dy \leq 2(|S| - k + 1)dy\end{aligned}$$

The first inequality follows as the primal constraint is still unsatisfied and so

$$\sum_{p \in [n] \backslash p_t} x(p) < n - k,$$

and as

$$\sum_{p \in [n] \backslash p_t \backslash S} x(p) = n - 1 - |S|$$

as $x(p) = 1$ for each of these pages, by the definition of S. The final step follows using the inequaltiy $a/k \leq a - k + 1$ for any $a \geq k$, and noting that $|S| \geq k$ (otherwise more than $n - k$ are already evicted), □

We finally show the dual is $O(\log k)$-infeasible.

Claim 2 *The dual is feasible up to a factor of $O(\log k)$.*

Proof: By our multiplicative update rule, $\frac{dx}{dy} = (1 + \frac{1}{k})$ as $\eta = \frac{1}{k}$. Solving the differential equation, we get

$$\int dy = [\ln x + \frac{1}{k}]_{x_{\text{initial}}}^{x_{\text{final}}}.$$

The left hand side ranges over all increments of y during $I(p, r(p,t))$, which is precisely the dual constraint for $x(p, r(p,t))$. As x lies in the range $[0, 1]$, the right hand side is at most $= \ln(k+1)$. □

1.5 Online Rounding

We obtained an $O(\log k)$ competitive fractional algorithm for paging, where the x variables indicate how much a page is eviced, and the LP cost is measured fractionally (evicting ϵ amount of a page costs ϵ). However, to specify a randomized algorithm for paging we must give a probability distribution on valid cache states (where each cache has k pages and contains p_t at time t). Let Q and Q' denote two distributions on cache states. The distance between these states is the defined as least cost to move transform Q into Q' (this is usual earthmover distance between two probability distributions defined on a metric space).

Fix a time t, and let $x_p = x_{p,t}$ (dropping t for convenience). It will be convenient to work with $y_p = 1 - x_p$, the amount of p in the cache, so that $\sum_p y_p = k$ and $y_{p_t,t} = 1$. To construct the randomized algorithm, we will show how to map Y to some distribution over states Q, with the following two properties:

1. *Consistency:* Each page p appears in y_p fraction of the caches in Q. That is, Q is consistent with the distribution on the marginals given by Y.

2. *Bounded Updates:* If Y changes to Y' incurring some fractional cost c, then Q can be modified to some Q' consistent with Y', with cost αc. The factor α is the loss in the rounding.

We will refer to such a procedure as online rounding. Note that unlike (the one shot) rounding in approximation algorithms, the difference here is that we need to maintain and update the rounded solution, without paying much more than the fractional update costs.

For paging, such a rounding was given by [2]. The rounding for weighted paging is more subtle and was considered in [3]. We sketch the proof and refer to [2] (section 4.2) for more details.

Theorem 3 *For unweighted paging, there is an online rounding procedure with alpha* $= 2$.

Proof: Given some fractional solution Y, let Q be any distribution on caches consistent with Y. Such a Q clearly exists. In paricular, starting with a distribution over empty caches, for each p place p into an arbitrary y_p fraction of the caches with the fewest pages. It is easily seen that no cache will exceed k.

We now show the online rounding step. If Y changes to Y', first we break it into elementary moves, where some page y_p increases by ϵ and some y_q decreases by ϵ. This pairing can be done so that cost of elementary moves is no more than fractional cost between Y and Y'. So it suffices to consider elementary moves. We now show how to move from Q to Q', while paying at most 2ϵ in eviction cost. We remove p from an arbitrary ϵ fraction of caches containing it. We add q to an arbitrary ϵ fraction of caches in Q that do not already contain q. Now at most $\epsilon' = \epsilon$ fraction of caches may be become of size $k+1$, but note that the fraction of caches with $k-1$ pages is also exactly ϵ'. So we pair the caches with $k+1$ and $k-1$ pages arbitrarily (in a measure preserving way), and for each such pair, move some page $\tilde{p}$ from a cache C with $k+1$ pages to a cache C' with $k-1$ pages, such that $\tilde{p} \notin C'$. Note that such a $\tilde{p}$ always exists. □

2 Dual Fitting

While Dual Fitting is a quite widely used technique in approximation algorithms it was first used for online algorithms by [1]. Unlike the primal-dual technique, here usually the algorithm is combinatorial or some natural greedy algorithm, but the duality is used for analyzing its performance. As a warm up we analyze the greedy algorithm for (offline) set-cover problem using dual-fitting.

2.0.1 Greedy Set Cover

Recall the offline set cover problem: Given a universe set U composed of n elements, a collection of sets $\{S_1, S_2, ..., S_k \ : S_i \subseteq U\}$, and a cost function $c(s) \mapsto R^+$, our task is to find the smallest cost collection of sets that can cover U.

The *Greedy Algorithm* starts with an empty collection, and at any step t, adds a set S with the minimum ratio of cost $c(S)$ to the number of remaining elements covered. It is well-known that greedy gives a $\ln n$ approximation. We analyze it dual fitting.

Consider the following (primal) linear program:

$$\min \sum_{s \in \mathbf{S}} c_s x_s \quad s.t. \sum_{S: e \in S} x_s \geq 1 \quad \forall e$$

The dual of this LP is the following:

$$\max \sum_{e \in \mathbf{U}} y_e \quad s.t. \sum_{S: e \in \mathbf{S}} y_e \leq c_s \quad \forall S$$

The idea will be to assign values of the dual variables y_e in some clever way based on the behavior of Greedy, so that the following two properties are satisfied.

1. The Greedy algorithm has cost at most $\sum_e y_e$.
2. y_e's give a $O(\log n)$-infeasible solution. That is, $y'_e = y_e / \log n$ for each e, forms a feasible dual solution.

By standard duality, this will directly imply a $\log n$ approximation.

Assigning the dual values: Fix some element e, and consider the first time t it is covered by some set S in Greedy. Let n_S be the number of elements at time t that are covered by S. We set $y_e = c_S / n_S$.

Claim 4 *$\sum_e y_e$ is exactly equal to the Greedy cost.*

Proof: When S is picked the cost is c_S, and each of the n_S elements that are covered for the first time pay c_S / n_S each. □

We now try to bound the violation of the dual constraints.

Claim 5 *For every S, $\sum_{e \in S} y_e \leq H_n \cdot c_S$.*

Proof: Fix a set S. Let $e_1, ..., e_k$ be the order (breaking ties arbitrarily) of elements of S in which Greedy covers them. The crucial observation is that for any $i \in [k]$, $y_{e_i} \leq c_S/(k - i + 1)$. This is because just before e_i was covered, S was still an option to pick for Greedy, and it would have covered the remaining elements $e_i, ..., e_k$ and had cost to coverage ratio $c_S/(k - i + 1)$. So Greedy could have only picking something better. Thus,

$$\sum_{i=1}^{k} y_{e_i} \leq \frac{\text{cost}(S)}{H_n}\left(\frac{1}{k} + \frac{1}{k-1} + ... + \frac{1}{1}\right) = \frac{H_k}{H_n}\text{cost}(S) \leq \text{cost}(S)$$

□

3 Flow Time Minimization on Unrelated Machines

In the flow time minimization on unrelated machines, we are given a set of machines and a set of jobs where p_{ij} is the processing time of job j on machine i, r_j is the release time/arrival time. Given a schedule if C_j is the completion time of job j, then its flow $F_j = C_j - r_j$ is the defined as its completion time minus its arrival time.

In the online version, we only know about the jobs information after it is released. We are not allowed job migration (can't switch jobs between machines) but we are allowed preemption which means we can interrupt a job and resume it later. The goal is to find a schedule that minimizes $\sum_j F_j$. We assume wlog that r_j, p_{ij} are all integers.

We begin with a simple observation.

Theorem 6 *The SRPT (Shortest Remaining Processing Time) algorithm is optimal on a single machine if preemption is allowed. This algorithm always executes the job with the least remaining work left.*

Proof: Suppose we have two jobs under an optimal schedule S where a job j is being processed at t while there is an unfinished job j' with strictly less processing p' remaining. Consider the time slots after t where j and j' are executed. Consider a new schedule S' that executes j' first in these slots and then does j. The rest of the jobs in the schedule are unchanged. It is easily checked the sum of completion times of j and j' in S' are strictly less than than in S: j' finishes strictly before either j or j' finishes in S and j finishes no later than either j or j' in S. □

We also note the following fact.

Theorem 7 *Let $n(t)$ denote the number of unfinished jobs in the system at time t. The total flow time $\sum_j f_j = \sum_t n(t)$, that is, the sum over the number of unfinished jobs at each time t.*

Proof: Imagine each job contributes one unit per time step it is alive. So total payment is total flow time. On the other hand, the amount received at time t is exactly $n(t)$. □

Theorem 8 *Any online algorithm can have arbitrarily high competitive ratio.*

Proof: There are 3 machines. There are two types on jobs: type 1 jobs can only be executed on machines 1 or 2 and have size 1 there. Type 2 jobs are can be done by machines 2 and 3 only and have unit size. The adversary releases two jobs of each type at each time for n time units. At time $t = n + 1$, at least n jobs are unfinished. Wlog say at least $n/2$ of them are of type 1. Then the adversary will release two jobs of type 1 at each step $t = n + 1, n + 2, \ldots, T + n$, where $T \gg n$.

Now, the online will have total flow time at least $nT/2$, at it has at least $n/2$ pending jobs for T steps (and using Theorem 7). On the other hand, the offline could have just delayed the type 2 jobs during $[1, n]$ and finished all the type 1 jobs by time $n+1$. Then it can work on the stream of type 1 jobs as they arrive during $[n+1, T+n]$. For type 2 jobs, n of them will be pending at time $n + 1$, but this backlog can be depleted by time $2n$. So, it incurs a flow time of at most $2(T + n)$ for type 1 jobs, and at most $O(n^2)$ for type 2 jobs.

Making $T \gg n$, say $T = n^3$ and making n large enough, the competitive ratio can be made arbitrarily large. □

To get around this issue, we will consider the problem in the *resource augmentation* setting, where the online algorithm is given a $(1 + \epsilon)$ times faster processor, some arbitrarily small $\epsilon > 0$, but its peformance is compared with an offline optimum running on a speed 1 processor. Often resource augmentation allows us to overcome the overly pessimistic nature of worst-case nature of competitive analysis, and leads to the design of interesting algorithms.

References

[1] S. Anand, Naveen Garg and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. *SODA* 2012, 1228–1241.

[2] Avrim Blum, Carl Burch and Adam Kalai. Finely-competitive Paging. *FOCS* 99, 450–458.

[3] Bansal, Nikhil, Niv Buchbinder, and Joseph Seffi Naor. "A primal-dual randomized algorithm for weighted paging." *Journal of the ACM (JACM)* 59.4 (2012): 19.

[4] Niv Buchbinder and Joseph Naor. The Design of Competitive Online Algorithms via a Primal-Dual Approach, *Foundations and Trends in Theoretical Computer Science*, 3(2-3), 93–263, 2009.