Weighted Caching

a Primal-Dual Approach

Xiaoxu Guo

Outline

Outline

- Introduction
  - **Problem Definition**
  - Summary of Results
- Background
  - Duality in Linear Programming
  - An Example: “Ski Rental” via Primal-Dual
- A Primal-Dual Approach to Weighted Caching
- Conclusion

Weighted Caching

In Weighted Caching, we are given a cache of size k, i.e., the cache can hold up to k pages.

n pages $r_1, \ldots, r_n$ are requested.

A requested page r_i should be *loaded* into the cache. If the cache already contains k pages, another page r_j should be *evicted* from the cache, and we incur a positive cost $w(r_j)$.

Our goal is minimize the sum of **eviction cost**.

Competitiveness

For an algorithm $\mathscr{A}$, let $\mathscr{A}(k, \sigma)$ be the cost given a cache size of k and a sequence of requests $\sigma = (r_1, \ldots, r_n)$.

Assume that OPT is the **optimal offline** algorithm, and $\mathscr{A}$ is an **online** algorithm.

For fixed $k \geq h$, the algorithm $\mathscr{A}$ is $c(h, k)$-competitive if for any sequence of requests σ, $\mathscr{A}(k, \sigma) \leq c(h, k) \cdot \mathrm{OPT}(h, \sigma) + o(\mathrm{OPT}(h, \sigma))$

Outline

- Introduction
  - Problem Definition
  - **Summary of Results**
- Background
  - Duality in Linear Programming
  - An Example: “Ski Rental” via Primal-Dual
- A Primal-Dual Approach to Weighted Caching
- Conclusion

Summary of Results

- Deterministic
  - [Chrobak, 1991], *"Balance"*, k-competitive
  - [Young, 1994], *"GreedyDual"*, $\dfrac{k}{k-h+1}$ - competitive, **this talk**
- Randomized
  - [Blum, 1996], $O(\log^2 k)$-competitive, $n = k + 1$
  - [Irani, 2002], $O(\log k)$-competitive, $r_i = 1$ or $r_i = M$ (a constant)
  - [Fiat, 2003], $O(\log k)$-competitive, $n = k + c$ (a constant)
  - [Bansal, 2007], $O(\log \dfrac{k}{k-h+1})$-competitive

Outline

- Introduction
  - Problem Definition
  - Summary of Results
- Background
  - **Duality in Linear Programming**
  - An Example: “Ski Rental” via Primal-Dual
- A Primal-Dual Approach to Weighted Caching
- Conclusion

Duality in Linear Programming

Primal Programming P

$$\begin{array}{lccc} & a_{1,1}x_1 + \ldots + a_{1,n}x_n & \geq & b_1 \\ & \vdots & \geq & \vdots \\ & a_{m,1}x_1 + \ldots + a_{m,n}x_n & \geq & b_m \\ & x_1, \ldots, x_n & \geq & 0 \\ \min & c_1x_1 + \ldots + c_nx_n & & \end{array}$$

Dual Programming D

$$\begin{array}{lccc} & a_{1,1}y_1 + \ldots + a_{m,1}y_m & \leq & c_1 \\ & \vdots & \leq & \vdots \\ & a_{1,n}y_1 + \ldots + a_{m,n}y_n & \leq & c_n \\ & y_1, \ldots, y_m & \geq & 0 \\ \max & b_1y_1 + \ldots + b_ny_m & & \end{array}$$

The primal variable x_i corresponds to a dual constraint $\sum_j a_{j,i}y_j \geq c_i$

A primal contraint $\sum_i a_{j,i}x_i \leq b_j$ corresponds to a dual variable y_j

Weak & Strong Duality

Let $(x_1, \ldots, x_n)$ be a finite feasible solution to the primal programming P, $(y_1, \ldots, y_m)$ be a finite feasible solution to the dual programming D.

Weak Duality states that $\sum_i c_i x_i \geq \sum_j b_j y_j$,

i.e., the objective of $P \geq$ the objective of D.

Strong Duality states that $\sum_i c_i x_i = \sum_j b_j y_j$

i.e., The objective of $P \geq$ the objective of D, if $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_m)$ are **optimal**.

Complementary Slackness

Let $(x_1, \ldots, x_n)$ be an **optimal** solution to the primal programming P, $(y_1, \ldots, y_m)$ be an optimal solution to the dual programming D.

Primal Slackness: $\sum_i a_{j,i} x_i = b_j$ or $y_j = 0$, i.e., either *the j-th primal constraint is tight* or *the corresponding dual variable is zero*.

Dual Slackness: $\sum_j a_{j,i} y_j = c_i$ or $x_i = 0$, i.e., either *the i-th dual constraint is tight* or *the corresponding primal variable is zero*.

Linear Programming in Online Tasks

How can Linear Programming help?

When new input arrives

- New constraints (and new vars) are added in LP
- Update the feasible solution

Restriction: cannot withdraw decision

- Impose additional **monotonicity** constraint
- Only increase (or decrease) variables

Primal-Dual Approach

Maintain a primal feasible solution **x** and a dual feasible solution **y**

When new input arrives

- Increase variables in **y** until some dual constraints are tight
- Set corresponding primal variable to non-zero

For competitive ratio

- Bound with dual objective ≤ optimal dual objective = optimal primal objective

Outline

- Introduction
  - Problem Definition
  - Summary of Results
- Background
  - Duality in Linear Programming
  - **An Example: “Ski Rental” via Primal-Dual**
- A Primal-Dual Approach to Weighted Caching
- Conclusion

Ski Rental

At a ski resort, renting costs \$1 per day, and buying costs $\$B$

2-competitive online algorithm

Rent for $(B - 1)$ days and buy on the B-th day

Let's see how the Primal-Dual approach produces the same algorithm.

Ski Rental

Linear Programming Formulation

Formulate into an Integer Programming

- $x_0 \in \{0,1\}$ indicates whether to buy (1 = to buy, 0 = not to buy)
- $x_i \in \{0,1\}$ indicates whether to rent on the i-th day ($i \geq 1$)
- Constraints: $x_0 + x_i \geq 1$ for $1 \leq i \leq n$
- Objective: min $Bx_0 + x_1 + \ldots + x_n$

Relax the Integer Programming to a Linear Programming

- Change $x_i \in \{0,1\}$ to $x_i \geq 0$
- No need for the constraint $x_i \leq 1$

Ski Rental

Duality

Primal Programming P

$$
\begin{array}{rcl}
x_0 + x_1 & \geq & 1 \\
\vdots & \geq & \vdots \\
x_0 + x_n & \geq & 1 \\
x_0, \ldots, x_n & \geq & 0 \\
\min \quad Bx_0 + x_1 + \ldots + x_n & &
\end{array}
$$

Dual Programming D

$$
\begin{array}{rcl}
y_1 + \ldots + y_n & \leq & B \\
y_1 & \leq & 1 \\
\vdots & \leq & \vdots \\
y_n & \leq & 1 \\
y_1, \ldots, y_n & \geq & 0 \\
\max \quad y_1 + \ldots + y_n & &
\end{array}
$$

Ski Rental

Feasible Solution Update

On the n-th day, a new primal constraint $x_0 + x_n \geq 1$ is added.

We increase the dual variable y_n.

- If $n < B$, the process stops reaching $y_n = 1$. We set the corresponding primal variable x_n to 1, i.e., rent.
- If $n = B$, the process stops reaching $y_1 + \ldots + y_n = B$. We set x_0 to 1, i.e., buy.

Primal Programming P

$$
\begin{array}{rcl}
x_0 + x_1 & \geq & 1 \\
\vdots & \geq & \vdots \\
x_0 + x_n & \geq & 1 \\
x_0, \ldots, x_n & \geq & 0
\end{array}
$$

$$\min \quad Bx_0 + x_1 + \ldots + x_n$$

Dual Programming D

$$
\begin{array}{rcl}
y_1 + \ldots + y_n & \leq & B \\
y_1 & \leq & 1 \\
\vdots & \leq & \vdots \\
y_n & \leq & 1 \\
y_1, \ldots, y_n & \geq & 0
\end{array}
$$

$$\max \quad y_1 + \ldots + y_n$$

Outline

- Introduction
  - Problem Definition
  - Summary of Results
- Background
  - Duality in Linear Programming
  - An Example: “Ski Rental” via Primal-Dual
- **A Primal-Dual Approach to Weighted Caching**
- Conclusion

Weighted Caching

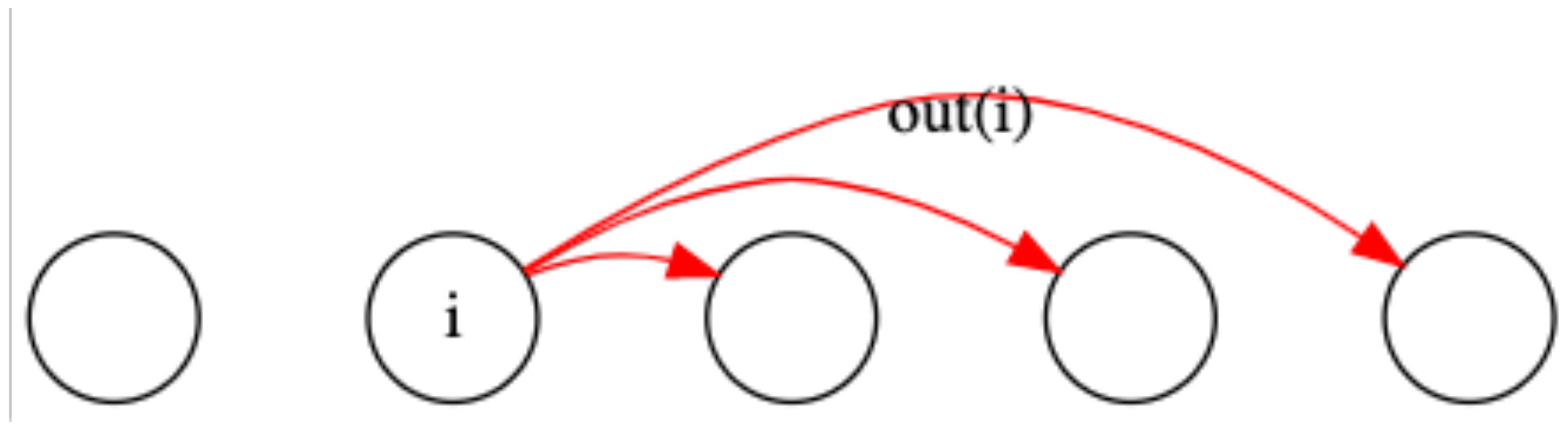
Linear Programming Formulation

Assume that the cache is occupied by k pages requested by r_0 and $w(r_0) = 0$.

$x_{i,j} = 1$ $(0 \leq i < j \leq n)$ if to load r_j into the cache, we have to evict the page requested by r_i. Or $x_{i,j} = 0$ otherwise.

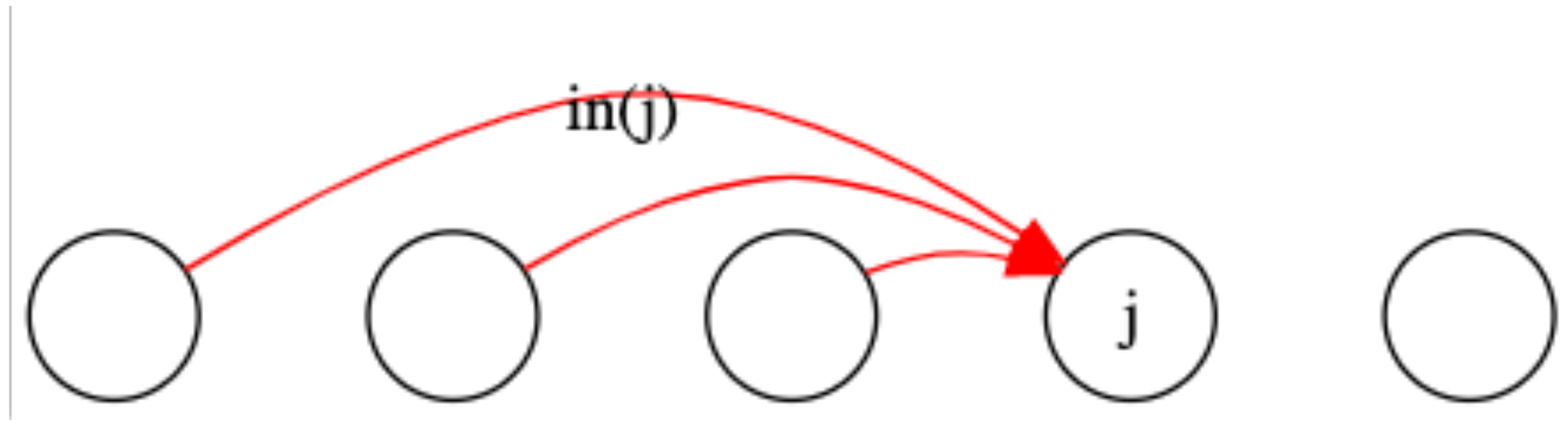
Constraints

$$\text{out}(0) = \sum_{j=1}^{n} x_{0,j} \leq k : r_0 \text{ can be evicted for at most } k \text{ times}$$



$$\text{out}(i) = \sum_{j=i+1}^{n} x_{i,j} \leq 1 \text{ for } i \geq 1\text{: } r_i \text{ can be evicted for at most once}$$

$$\text{in}(j) = \sum_{i=0}^{j-1} x_{i,j} = 1 \text{ for } j \geq 1\text{: } r_j \text{ should be loaded}$$



$$\text{Objective: min} \sum_{0 \leq i < j \leq n} \delta(r_i, r_j) x_{i,j} \text{ where } \delta(r_i, r_j) = \begin{cases} 0 & \text{if } r_i = r_j \\ w(r_i) & \text{if } r_i \neq r_j \end{cases}$$

Weighted Caching

An example

Cache size $k = 2$

$\sigma = (1,2,3,2,1)$

A feasible solution (may not optimal)

$x_{0,1} = x_{0,2} = x_{1,5} = x_{2,3} = x_{3,4} = 1$

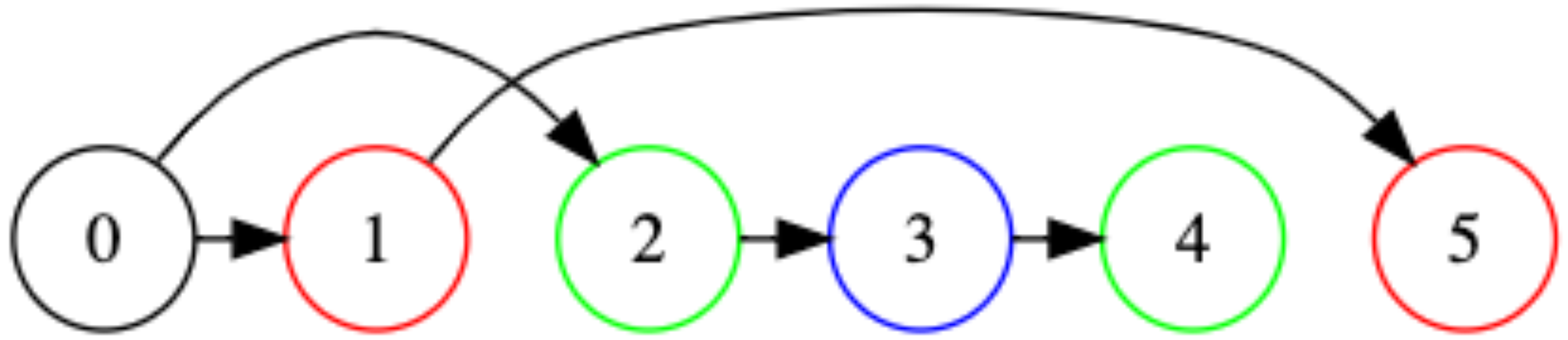
Other $x_{i,j} = 0$

Eviction cost

$w(r_0) + w(r_0) + 0 + w(r_2) + w(w_3) = w(2) + w(3)$

NOTE: As $r_1 = r_5, \sigma(r_1, r_5) = 0$

No.	Request	Cache	
1	1	$\{r_0, \mathbf{r_1}\}$	Evict r_0
2	2	$\{r_1, \mathbf{r_2}\}$	Evict r_0
3	3	$\{r_1, \mathbf{r_3}\}$	Evict r_2
4	2	$\{r_1, \mathbf{r_4}\}$	Evict r_3
5	1	$\{r_4, \mathbf{r_5}\}$	Evict r_1



An edge $i \to j$ is drawn iff $x_{i,j} = 1$
Different colors - different pages

Weighted Caching

Duality

Primal Programming P_k

$$\begin{array}{rcccll} -\text{out}(0) & = & -\sum_{j=1}^{n} x_{0,j} & \geq & -k & \\ -\text{out}(i) & = & -\sum_{j=i+1}^{n} x_{1,j} & \geq & -1 & (1 \leq i \leq n) \\ \text{in}(j) & = & \sum_{i=0}^{j-1} x_{i,j} & = & 1 & (1 \leq j \leq n) \\ & & x_{i,j} & \geq & 0 & (0 \leq i < j \leq n) \\ & \min & \sum_{0 \leq i < j \leq n} \delta(r_i, r_j) x_{i,j} & & & \end{array}$$

Dual Programming D_k

$$\begin{array}{lccl} b_j - a_i & \leq & \delta(r_i, r_j) & (0 \leq i < j \leq n) \\ a_0, \ldots, a_n & \geq & 0 & \\ \max \quad -ka_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j & & & \end{array}$$

We add a subscript k in P_k and D_k because we want to deal with different cache sizes for online and optimal offline algorithms .

a_i $(0 \leq i \leq n)$ is the dual variable for the primal constraint $\text{out}(i)$

b_j $(1 \leq j \leq n)$ is the dual variable for the primal constraint $\text{in}(j)$

Weighted Caching

Update Feasible Solution

Let S be the **multiset** of indices of requests in the cache. Initially, S contains k copies of 0.

a_i and b_j are initially 0.

When the request n arrives,

- if the page is already in the cache ($\exists i \in S, r_i = r_n$), we set $x_{i,n} = 1$ and $S \leftarrow S \backslash \{i\} \cup \{n\}$.
- Otherwise, increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \leq i \leq n\}$ by $\Delta \geq 0$ until some dual constraints are tight.

Primal Programming P

$$\begin{array}{rcl} -\sum_{j=1}^{n} x_{0,j} & \geq & -k \\ -\sum_{j=i+1}^{n} x_{1,j} & \geq & -1 \\ \sum_{i=0}^{j-1} x_{i,j} & = & 1 \\ x_{i,j} & \geq & 0 \end{array}$$

$$\min \quad \sum_{0 \leq i < j \leq n} \delta(r_i, r_j) x_{i,j}$$

Dual Programming D

$$\begin{array}{rcl} b_j - a_i & \leq & \delta(r_i, r_j) \\ a_0, \ldots, a_n & \geq & 0 \end{array}$$

$$\max \quad -ka_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$$

Weighted Caching

Update Feasible Solution

Increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \leq i \leq n\}$ by $\Delta \geq 0$ until some dual constraints are tight

Observation

- $i \in S \implies a_i = 0$. Proof: a_i won't change until r_i is evicted ($i \notin S$).
- $b_1 \geq \ldots \geq b_n$.

Primal Programming P

$$
\begin{array}{rcr}
-\sum_{j=1}^{n} x_{0,j} & \geq & -k \\
-\sum_{j=i+1}^{n} x_{1,j} & \geq & -1 \\
\sum_{i=0}^{j-1} x_{i,j} & = & 1 \\
x_{i,j} & \geq & 0
\end{array}
$$

$$\min \quad \sum_{0 \leq i < j \leq n} \delta(r_i, r_j) x_{i,j}$$

Dual Programming D

$$
\begin{array}{rcr}
b_j - a_i & \leq & \delta(r_i, r_j) \\
a_0, \ldots, a_n & \geq & 0
\end{array}
$$

$$\max \quad -ka_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$$

Weighted Caching

Update Feasible Solution

Increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \leq i \leq n\}$ by $\Delta \geq 0$ until **some dual constraints** are tight.

What's "some dual constraints"?

To evict a page, we want a constraint $b_j - a_i \leq \delta(r_i, r_j)$ where $i \in S$ to become tight.

Because $i \in S \implies a_i = 0$, the constraint becomes $b_j \leq \delta(r_i, r_j)$.

Because $r_i \neq r_n$, the constraint becomes $b_j \leq w(r_i)$.

Because $b_{i+1} \geq b_{i+2} \geq \ldots$, we can focus on the constraint $b_{i+1} \leq w(r_i)$.

Primal Programming P

$$\begin{aligned} -\textstyle\sum_{j=1}^{n} x_{0,j} &\geq -k \\ -\textstyle\sum_{j=i+1}^{n} x_{1,j} &\geq -1 \\ \textstyle\sum_{i=0}^{j-1} x_{i,j} &= 1 \\ x_{i,j} &\geq 0 \\ \min \quad \textstyle\sum_{0 \leq i < j \leq n} \delta(r_i, r_j) x_{i,j} & \end{aligned}$$

Dual Programming D

$$\begin{aligned} b_j - a_i &\leq \delta(r_i, r_j) \\ a_0, \ldots, a_n &\geq 0 \\ \max \quad -ka_0 - \textstyle\sum_{i=1}^{n} a_i + \textstyle\sum_{j=1}^{n} b_j & \end{aligned}$$

Observation

- $i \in S \implies a_i = 0$
- $b_1 \geq \ldots \geq b_n$.

Weighted Caching

Update Feasible Solution

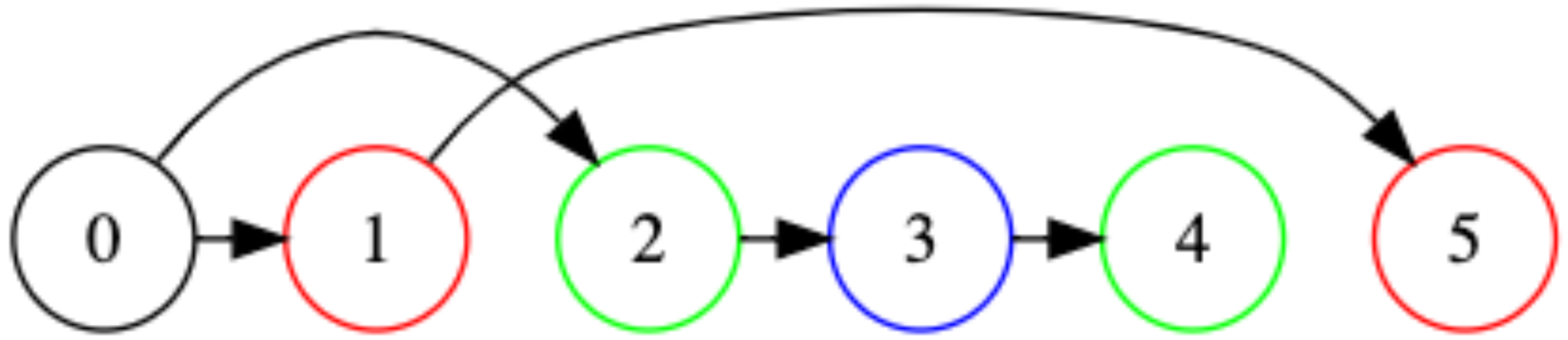
Increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \le i \le n\}$ by $\Delta \ge 0$ until **some dual constraints** are tight.

Focus on the constraint $b_{i+1} \le w(r_i)$

Let i^- be the request which loads page r_i.

Example:

$4^- = 4, 1^- = 5^- = 1.$



Weighted Caching

Update Feasible Solution

Increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \le i \le n\}$ by $\Delta \ge 0$ until **some dual constraints** are tight.

Focus on the constraint $b_{i+1} \le w(r_i)$

Let i^- be the request which loads page r_i.

We increase dual variables until $\exists i \in S, b_{i^-+1} = w(r_{i^-}) = w(r_i)$, set $x_{i,n} = 1$ and $S \leftarrow S \backslash \{i\} \cup \{n\}$.

Primal Programming P

$$
\begin{array}{rcl}
-\sum_{j=1}^{n} x_{0,j} & \ge & -k \\
-\sum_{j=i+1}^{n} x_{1,j} & \ge & -1 \\
\sum_{i=0}^{j-1} x_{i,j} & = & 1 \\
x_{i,j} & \ge & 0
\end{array}
$$

$$\min \quad \sum_{0 \le i < j \le n} \delta(r_i, r_j) x_{i,j}$$

Dual Programming D

$$
\begin{array}{rcl}
b_j - a_i & \le & \delta(r_i, r_j) \\
a_0, \ldots, a_n & \ge & 0
\end{array}
$$

$$\max \quad -k a_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$$

Weighted Caching

Update Feasible Solution

Increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \leq i \leq n\}$ by $\Delta \geq 0$ until **some dual constraints** are tight.

Increase dual variables until $\exists i \in S, b_{i^-+1} = w(r_{i^-}) = w(r_i)$.

Why the solution is still feasible?

For $i \notin S, (b_j + \Delta) - (a_i + \Delta) = b_j - a_i \leq \delta(r_i, r_j)$.

For $i \in S, b_j - a_i \leq b_{i^-+1} - a_i = w(r_i) - 0 = \delta(r_i, r_j)$ as $j \geq i + 1 \geq i^- + 1 \implies b_j \leq b_{i^-+1}$.

Primal Programming P

$$
\begin{array}{rcr}
-\sum_{j=1}^{n} x_{0,j} & \geq & -k \\
-\sum_{j=i+1}^{n} x_{1,j} & \geq & -1 \\
\sum_{i=0}^{j-1} x_{i,j} & = & 1 \\
x_{i,j} & \geq & 0
\end{array}
$$

$$\min \quad \sum_{0 \leq i < j \leq n} \delta(r_i, r_j) x_{i,j}$$

Dual Programming D

$$
\begin{array}{rcr}
b_j - a_i & \leq & \delta(r_i, r_j) \\
a_0, \ldots, a_n & \geq & 0
\end{array}
$$

$$\max \quad -k a_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$$

Observation

- $i \in S \implies a_i = 0$
- $b_1 \geq \ldots \geq b_n$.

Weighted Caching

Algorithm Recap

Maintain

- Primal solution $x_{i,j}$, dual solution $a_0, \ldots, a_n, b_1, \ldots, b_n$
- A set $S = \{0,\ldots,0\}$ (k copies) initially

When a request r_n arrives,

- Either $\exists i \in S, r_i = r_n$.
- Or we increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \le i \le n\}$ until $\exists i \in S, b_{i^-+1} = w(r_i)$.

- Set $x_{i,n} = 1$ and $S \leftarrow S \backslash \{i\} \cup \{n\}$.

Primal Programming P

$$
\begin{array}{rcl}
-\sum_{j=1}^{n} x_{0,j} & \ge & -k \\
-\sum_{j=i+1}^{n} x_{1,j} & \ge & -1 \\
\sum_{i=0}^{j-1} x_{i,j} & = & 1 \\
x_{i,j} & \ge & 0
\end{array}
$$

$$\min \quad \sum_{0 \le i < j \le n} \delta(r_i, r_j) x_{i,j}$$

Dual Programming D

$$
\begin{array}{rcl}
b_j - a_i & \le & \delta(r_i, r_j) \\
a_0, \ldots, a_n & \ge & 0
\end{array}
$$

$$\max \quad -ka_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$$

Observation

- $i \in S \implies a_i = 0$
- $b_1 \ge \ldots \ge b_n$.

Weighted Caching

Competitiveness

Let $\|\mathbf{a}, \mathbf{b}\|_t = -ta_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$

Claim: Given a cache size k and requests σ, our algorithm $\mathscr{A}$ has

$$\mathscr{A}(k, \sigma) \le \frac{k}{k-h+1}\|\mathbf{a}, \mathbf{b}\|_h - \sum_{j \in S} b_{j^- +1}$$

$(\mathbf{a}, \mathbf{b})$ is feasible in $D_k \implies (\mathbf{a}, \mathbf{b})$ is feasible in D_h (because k does not occur in constraints)

$\|\mathbf{a}, \mathbf{b}\|_h \le$ optimum of $D_h =$ optimum of $P_h = \mathrm{OPT}(h, \sigma)$

where OPT is the **optimal offline** algorithm

Plus $b_j \ge 0$, we have $\mathscr{A}(k, \sigma) \le \dfrac{k}{k-h+1}\mathrm{OPT}(h, \sigma)$.

Primal Programming P_k

$$\begin{array}{rcl}
-\sum_{j=1}^{n} x_{0,j} & \ge & -k \\
-\sum_{j=i+1}^{n} x_{1,j} & \ge & -1 \\
\sum_{i=0}^{j-1} x_{i,j} & = & 1 \\
x_{i,j} & \ge & 0 \\
\end{array}$$

$$\min \quad \sum_{0 \le i < j \le n} \delta(r_i, r_j) x_{i,j}$$

Dual Programming D_k

$$\begin{array}{rcl}
b_j - a_i & \le & \delta(r_i, r_j) \\
a_0, \ldots, a_n & \ge & 0 \\
\end{array}$$

$$\max \quad -ka_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$$

Observation

- $i \in S \implies a_i = 0$
- $b_1 \ge \ldots \ge b_n$.

Weighted Caching

Proof of the claim

Let C be our eviction cost, and

$\mathcal{U} = \frac{k}{k-h+1}\|\mathbf{a},\mathbf{b}\|_h - \sum_{j\in S} b_{j^-+1}$. We want to prove $C \le \mathcal{U}$.

Initially, $C = \mathcal{U} = 0$.

If $\exists i \in S, r_i = r_n$, C is unchanged.
Also, $n^- = i^- \implies \sum_{j\in S\backslash\{i\}\cup\{n\}} b_{j^-+1} = \sum_{j\in S} b_{j^-+1}$

Algorithm Recap

When a request r_n arrives,

- Either $\exists i \in S, r_i = r_n$.
- Or we increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \le i \le n\}$ until $\exists i \in S, b_{i^-+1} = w(r_i)$.

- Set $x_{i,n} = 1$ and $S \leftarrow S\backslash\{i\} \cup \{n\}$.

Let $\|\mathbf{a},\mathbf{b}\|_t = -ta_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$

Claim: $\mathcal{A}(k,\sigma) \le \frac{k}{k-h+1}\|\mathbf{a},\mathbf{b}\|_h - \sum_{j\in S} b_{j^-+1}$

Weighted Caching

Proof of the claim

Otherwise, C is increased by $w(r_i)$.

Claim': $\mathcal{U}$ does not change if we increase dual variables by Δ.

Proof

If $0 \in S$, it's easy to verify that $\Delta = 0$ and $i = 0$.

If $0 \notin S$, ...

Algorithm Recap

When a request r_n arrives,

- Either $\exists i \in S, r_i = r_n$.
- Or we increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \leq i \leq n\}$ until $\exists i \in S, b_{i^-+1} = w(r_i)$.

- Set $x_{i,n} = 1$ and $S \leftarrow S \backslash \{i\} \cup \{n\}$.

Let $\|\mathbf{a}, \mathbf{b}\|_t = -t a_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$

Claim: $\mathscr{A}(k, \sigma) \leq \frac{k}{k-h+1} \|\mathbf{a}, \mathbf{b}\|_h - \sum_{j \in S} b_{j^-+1}$

Weighted Caching

Proof of the claim

Otherwise, C is increased by $w(r_i)$.

If $0 \notin S$

The first term

$$
\begin{aligned}
&\|\mathbf{a}', \mathbf{b}'\|_h \\
&= -h(a_0 + \Delta) - \sum_{i \notin S} (a_i + \Delta) - \sum_{i \in S} a_i + \sum_{j} (b_j + \Delta) \\
&= \|\mathbf{a}, \mathbf{b}\|_h + (k - h + 1) \cdot \Delta
\end{aligned}
$$

The second term

$$\sum_{j \in S} (b_{j^-+1} + \Delta) = \left(\sum_{j \in S} b_{j^-+1} \right) + |S| \cdot \Delta = \left(\sum_{j \in S} b_{j^-+1} \right) + k \cdot \Delta$$

Thus, $\mathscr{U}$ remains unchanged increasing dual variables.

Algorithm Recap

When a request r_n arrives,

- Either $\exists i \in S, r_i = r_n$.
- Or we increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \leq i \leq n\}$ until $\exists i \in S, b_{i^-+1} = w(r_i)$.
- Set $x_{i,n} = 1$ and $S \leftarrow S \backslash \{i\} \cup \{n\}$.

Let $\|\mathbf{a}, \mathbf{b}\|_t = -ta_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$

Claim: $\mathscr{A}(k, \sigma) \leq \frac{k}{k - h + 1} \|\mathbf{a}, \mathbf{b}\|_h - \sum_{j \in S} b_{j^-+1}$

Weighted Caching

Proof of the claim

Otherwise, C is increased by $w(r_i)$.

Claim': $\mathscr{U}$ does not change if we increase dual variables by Δ.

And we have $n^- = n \implies b_{n^-+1} = 0$.

$$\sum_{j \in S \backslash \{i\} \cup \{n\}} b_{j^-+1}$$

$$= \left(\sum_{j \in S} b_{j^-+1} \right) - b_{i^-+1} + b_{n^-+1}$$

$$= \left(\sum_{j \in S} b_{j^-+1} \right) - w(r_i) + 0$$

Thus, $\mathscr{U}$ is also increased by $w(r_i)$. The claim is proved.

Algorithm Recap

When a request r_n arrives,

- Either $\exists i \in S, r_i = r_n$.
- Or we increase dual variables $\{a_i : i \notin S\} \cup \{b_i : 1 \leq i \leq n\}$ until $\exists i \in S, b_{i^-+1} = w(r_i)$.
- Set $x_{i,n} = 1$ and $S \leftarrow S \backslash \{i\} \cup \{n\}$.

Let $\|\mathbf{a}, \mathbf{b}\|_t = -t a_0 - \sum_{i=1}^{n} a_i + \sum_{j=1}^{n} b_j$

Claim: $\mathscr{A}(k, \sigma) \leq \frac{k}{k-h+1} \|\mathbf{a}, \mathbf{b}\|_h - \sum_{j \in S} b_{j^-+1}$

Outline

- Introduction
  - Problem Definition
  - Summary of Results
- Background
  - Duality in Linear Programming
  - An Example: “Ski Rental” via Primal-Dual
- A Primal-Dual Approach to Weighted Caching
- **Conclusion**

Conclusion

Primal-Dual Approach

We have seen two online tasks solved via a Primal-Dual Approach - a toy example “Ski Rental” and Weight Caching.

Steps to design an online algorithms using Primal-Dual

1. Formulate with Linear Programming and its dual
2. Monotonically change dual variables tightening dual constraints, and update the corresponding primal variables
3. Bound the primal objective by dual objective, and use Duality Theorem to complete the competitiveness proof

Thanks for Listening!

Reference

- Buchbinder, Niv, and Joseph Naor. The design of competitive online algorithms via a primal-dual approach. Now Publishers Inc, 2009.
- Borodin, Pankratov. Online and Other Myopic Algorithms, Working Draft, 2021.