9th Max-Planck Advanced Course on the Foundations of Computer Science (ADFOCS)

Primal-Dual Algorithms for Online Optimization: Lecture 3

Seffi Naor
Computer Science Dept.
Technion
Haifa, Israel
Contents

• The ad-auctions problem

• Caching
  • Relationship with k-server
  • Weighted paging
  • Web caching
What are Ad-Auctions?

You type in a query:
You get:

Algorithmic Search results

And … Ad-auctions
How do search engines sell ads?

- Each advertiser:
  - Sets a daily budget
  - Provides bids on interesting keywords

- Search Engine (on each keyword):
  - Selects ads
  - Advertiser pays bid if user clicks on ad.

Goal (of search engine):
Maximize Revenue
How much does it cost?

Buying keyword like “divorce lawyer” may cost as much as $40 per click.

Estimates are for September 30th 2007.

<table>
<thead>
<tr>
<th>Keywords</th>
<th>Estimated Avg. CPC</th>
<th>Estimated Ad Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>luxury vacation</td>
<td>$3.40</td>
<td>1 - 3</td>
</tr>
<tr>
<td>golf vacation</td>
<td>$3.23</td>
<td>1 - 3</td>
</tr>
<tr>
<td>hotel vacation</td>
<td>$2.66</td>
<td>1 - 3</td>
</tr>
<tr>
<td>scuba vacation</td>
<td>$2.62</td>
<td>1 - 3</td>
</tr>
<tr>
<td>vacation tours</td>
<td>$2.61</td>
<td>1 - 3</td>
</tr>
<tr>
<td>hotels vacation</td>
<td>$2.61</td>
<td>1 - 3</td>
</tr>
<tr>
<td>Tel Aviv vacation</td>
<td>$2.50</td>
<td></td>
</tr>
<tr>
<td>vacation rentals</td>
<td>$2.40</td>
<td>1 - 3</td>
</tr>
<tr>
<td>spa vacation</td>
<td>$2.39</td>
<td>1 - 3</td>
</tr>
<tr>
<td>Eilat vacation</td>
<td>$2.36</td>
<td></td>
</tr>
<tr>
<td>vacation in eilat</td>
<td>$2.24</td>
<td>1 - 3</td>
</tr>
<tr>
<td>hotels in eilat</td>
<td>$2.21</td>
<td>1 - 3</td>
</tr>
<tr>
<td>Red sea vacation</td>
<td>$2.15</td>
<td></td>
</tr>
<tr>
<td>beach vacation</td>
<td>$2.12</td>
<td>1 - 3</td>
</tr>
<tr>
<td>Jewish vacation</td>
<td>$2.05</td>
<td></td>
</tr>
<tr>
<td>sea vacation</td>
<td>$1.89</td>
<td>1 - 3</td>
</tr>
</tbody>
</table>
Mathematical Model

• Buyer i:
  – has a daily budget $B(i)$

• Online Setting:
  – items (keywords) arrive one-by-one.
  – buyers bid on the items (bid can be zero)

• Algorithm:
  – Assigns each item to an interested buyer.

Assumption:
Each bid is small compared to the daily budget.
Ad-auctions – Linear Program

I - Set of buyers.  B(i) – Budget of buyer i
J - Set of items.  b(i,j) – bid of buyer i on item j

\[ y(i, j) = 1 \Rightarrow j\text{-th adword is sold to buyer } i. \]

\[
\begin{align*}
\text{max} & \sum_{i \in I} \sum_{j \in J} b(i, j) y(i, j) \\
\text{s.t.:} & \quad \sum_{i \in I} y(i, j) \leq 1 \\
& \quad \sum_{j \in J} b(i, j) y(i, j) \leq B(i)
\end{align*}
\]

For each item j: \[ \sum_{i \in I} y(i, j) \leq 1 \]

For each buyer i: \[ \sum_{j \in J} b(i, j) y(i, j) \leq B(i) \]

Buyers do not exceed their budget
Ad-auctions: Primal and Dual

**P: Primal Covering**

$$\min \sum_{i \in I} B(i) x(i) + \sum_{j \in J} z(j)$$

For each item $j$ and buyer $i$: $b(i, j) x(i) + z(j) \geq b(i, j)$

---

**D: Dual Packing**

$$\max \sum_{i \in I} \sum_{j \in J} b(i, j) y(i, j)$$

For each item $j$: $\sum_{i \in I} y(i, j) \leq 1$

For each buyer $i$: $\sum_{j \in J} b(i, j) y(i, j) \leq B(i)$
The Primal-Dual Algorithm

- Initially: for each buyer $i$: $x(i) \leftarrow 0$
- When new item $j$ arrives:
  - Assign the item to the buyer $i$ that maximizes:

$$b(i, j)[1 - x(i)]$$

- if $x(i) \geq 1$ do nothing, otherwise:
  - $y(i, j) \leftarrow 1$
  - $z(j) \leftarrow b(i, j)[1 - x(i)]$
  - $x(i) \leftarrow x(i) \left[1 + \frac{b(i, j)}{B(i)}\right] + \frac{b(i, j)}{B(i)[c - 1]}$ - ‘c’ later.
Analysis of Online Algorithm

Proof of competitive factor:
1. Primal solution is feasible.
2. In each iteration, $\Delta P \leq (1 + 1/(c-1))\Delta D$.
3. Dual is feasible.

Conclusion:
Algorithm is $(1+ 1/(c-1))$-competitive
Analysis of Online Algorithm

1. **Primal solution is feasible.**

   For each item $j$ and buyer $i$:
   
   $$b(i, j)x(i) + z(j) \geq b(i, j)$$

   If $x(i) \geq 1$ the solution is feasible.

   Else, $z(j) \leftarrow \max_i \{ b(i,j)(1-x(i)) \} $, and the solution is feasible

   Increasing $x(i)$ in the future maintains feasibility
Analysis of Online Algorithm

2. In each iteration, $\Delta P \leq (1+ 1/(c-1))\Delta D$:
   
   If $x(i) \geq 1$, $\Delta P = \Delta D = 0$
   
   Otherwise:
   
   $\Delta D = b(i,j)$
   
   $\Delta P = B(i)\Delta x(i) + z(j)$
   
   $= B(i)\left[ \frac{b(i, j)x(i)}{B(i)} + \frac{b(i, j)}{B(i)[c-1]} \right] + b(i, j)[1-x(i)] = b(i, j)\left[ 1 + \frac{1}{(c-1)} \right]$
3. Dual is feasible:

- The “last” item assigned to a buyer may exceed his budget
- The online algorithm loses the revenue from such an item
- This where the assumption that each individual bid is small with respect to the budget is used
- The maximum ratio between a bid of any buyer and its total budget:

$$ R = \max_{i \in I, j \in M} \left\{ \frac{b(i, j)}{B(i)} \right\} $$
Analysis of Online Algorithm

It is easy to prove by induction that:

\[ 1 \geq x(i) \geq \frac{1}{c-1} \left[ \frac{\sum_{j} b(i,j) y(i,j)}{B(i)} - 1 \right] \]

- if \( x(i) \geq 1 \), primal constraints of buyer \( i \) are feasible.
- No more items are assigned to the buyer.
- simplifying the inequality we get that the dual is almost feasible (up to the “last” item)
Competitive Factor

- Setting

\[ c = (1 + R)^{\frac{1}{R}} \]

\[ c \to e \text{ when } R \to 0 \]

- The competitive factor is

\[ \left(1 - \frac{1}{c}\right) (1 - R) = \left(1 - \frac{1}{e}\right) \text{ if } R \to 0 \]

- Result obtained by [MSVV, FOCS 2005]
Extensions – Getting More Revenue

• Seller wants to sell several advertisements

• There are $\ell$ slots on each page

• Bidders provide bids on keywords which are slot dependent $b(i,j,k)$ – bid of buyer i on keyword j and slot k

• A slot can only be allocated to one advertiser
Linear Program

Dual (Packing)

Maximize:  \[ \sum_{j=1}^{m} \sum_{i=1}^{n} \sum_{\ell=1}^{k} b(i, j, \ell) y(i, j, \ell) \]
Subject to:  \[ \sum_{i=1}^{n} y(i, j, k) \leq 1 \]
\[ \forall 1 \leq i \leq n: \sum_{j=1}^{m} \sum_{k=1}^{\ell} b(i, j, k) y(i, j, k) \leq B(i) \]
\[ \forall 1 \leq j \leq m, 1 \leq i \leq n: \sum_{k=1}^{\ell} y(i, j, k) \leq 1 \]

Primal (Covering)

Minimize:  \[ \sum_{i=1}^{n} B(i) x(i) + \sum_{j=1}^{m} \sum_{k=1}^{\ell} z(j, k) + \sum_{i=1}^{n} \sum_{j=1}^{m} s(i, j) \]
Subject to:  \[ b(i, j, k) x(i) + z(j, k) + s(i, j) \geq b(i, j, k) \]
Initially, $\forall i$, $x(i) \leftarrow 0$.

Upon arrival of a new item $j$:

1. Generate a bipartite graph $H$: $n$ buyers on one side and $\ell$ slots on the other side. Edge $(i, k) \in H$ has weight $b(i, j, k)(1 - x(i))$.

2. Find a maximum weight (integral) matching in $H$, i.e., an assignment to the variables $y(i, j, k)$.

3. Charge buyer $i$ the minimum between $\sum_{k=1}^{\ell} b(i, j, k)y(i, j, k)$ and its remaining budget.

4. For each buyer $i$, if there exists slot $k$ for which $y(i, j, k) > 0$:

$$x(i) \leftarrow x(i) \left(1 + \frac{b(i, j, k)y(i, j, k)}{B(i)}\right) + \frac{b(i, j, k)y(i, j, k)}{(c - 1) \cdot B(i)}$$

Remark: If $\ell = 1$, the maximum weight matching is a single edge maximizing $b(i, j)(1 - x(i))$. 
Analysis of Online Algorithm

Proof of competitive factor:
1. Primal solution is feasible.
2. In each iteration, $\Delta P \leq (1 + 1/(c-1)) \Delta D$.
3. Dual is feasible.

Conclusion: Algorithm is $(1 + 1/(c-1))$-competitive.
Analysis: Crucial Fact

<table>
<thead>
<tr>
<th>Dual (Packing)</th>
<th>Primal (Covering)</th>
</tr>
</thead>
</table>
| \[
\max \sum_i \sum_k b(i, j, k) (1 - x(i)) y(i, j, k)
\] | \[
\min \sum_{i=1}^n s(i, j) + \sum_{k=1}^\ell z(j, k)
\] |
| Subject to: \[\forall 1 \leq k \leq \ell: \sum_{i=1}^n y(i, j, k) \leq 1\] | Subject to: \[\forall (i, k): s(i, j) + z(j, k) \geq b(i, j, k) (1 - x(i))\] |
| \[\forall 1 \leq i \leq n: \sum_{k=1}^\ell y(i, j, k) \leq 1\] | \[\forall i, k: s(i, j), z(j, k) \geq 0\] |
| \[y(i, j, k) \geq 0\] | \[
\sum_{i=1}^n s(i, j) + \sum_{k=1}^\ell z(j, k) = \sum_{i=1}^n b(i, j, k) (1 - x(i))
\] |

Figure 1: The LP for the matching problem solved for item \(j\)

- Primal variables are the same as in the allocation problem.
- There is an optimal primal solution and a dual \textit{integral} solution satisfying:

\[
\sum_{i=1}^n \sum_{k=1}^\ell b(i, j, k) (1 - x(i)) y(i, j, k) = \sum_{i=1}^n s(i, j) + \sum_{k=1}^\ell z(j, k).
\]
- This solution defines the assignment to the primal and dual variables
Analysis of Online Algorithm

1. Primal solution is feasible.

for each buyer $l$, item $j$, slot $k$:

$$b(i, j, k)x(i) + z(j, k) + s(i, j) \geq b(i, j, k).$$

this constraint is satisfied by the primal-dual solution to the weighted matching LP

Increasing $x(i)$ in the future maintains feasibility
Analysis of Online Algorithm

2. In each iteration, $\Delta P \leq (1 + 1/(c-1))\Delta D$:

$$
\Delta P = \sum_{i=1}^{n} z(j, i) + \sum_{k=1}^{\ell} s(i, j) + \sum_{i=1}^{n} B(i) \Delta x(i)
$$

$$
= \sum_{i=1}^{n} \sum_{k=1}^{\ell} b(i, j, k) (1 - x(i)) y(i, j, k)
+ \sum_{i=1}^{n} \sum_{k=1}^{\ell} B(i) \left( \frac{b(i, j, k) x(i) y(i, j, k)}{B(i)} + \frac{b(i, j, k) y(i, j, k)}{(c - 1) \cdot B(i)} \right)
$$

$$
= \sum_{i=1}^{n} \sum_{k=1}^{\ell} b(i, j, k) y(i, j, k) \left( 1 + \frac{1}{c-1} \right).
$$

Since $\Delta D = \sum_{i=1}^{n} \sum_{k=1}^{\ell} b(i, j, k) y(i, j, k)$, the claim follows.
Analysis of Online Algorithm

3. Dual is feasible:

• similar to the proof in the single slot case

• the competitive factor is

\[
\left(1 - \frac{1}{c}\right) (1 - R) = \left(1 - \frac{1}{e}\right) \quad \text{if } R \to 0
\]
Online Matching in Bipartite Graphs

Input: bipartite graph $H=(U,V,E)$

Goal: find a maximum matching in $H$

Online model:

- $V$ is known
- the vertices of $U$ arrive one by one and expose their neighbors in $V$ (upon arrival)
- for each $u \in U$, upon arrival, online algorithm decides whether to match $u$ to a vertex in $V$
Online Algorithms for Matching

• any algorithm that matches a vertex, if possible, achieves competitive ratio $\frac{1}{2}$ since 
  \[
  \text{(maximal matching)} \geq \frac{1}{2} \cdot \text{(maximum matching)}
  \]

• online algorithm of [KVV 1990]:
  – choose a random permutation $\pi$ on $V$
  – assign each vertex $u \in U$ to the minimum index vertex in $V$ with respect to $\pi$
  – competitive ratio: $1 - \frac{1}{e}$

• an online primal-dual algorithm can find a fractional matching with competitive ratio $1 - \frac{1}{e}$

• can an integral matching be computed via the primal-dual method?
The Paging/Caching Problem

- Relationship to the k-Server Problem
- Weighted paging
- Web caching
The Paging/Caching Problem (Reminder)

Universe of n pages
Cache of size $k \ll n$

Request sequence of pages: 1, 6, 4, 1, 4, 7, 6, 1, 3, ...

If requested page is in cache: no penalty.
Else, cache miss! load requested page into cache, evicting some other page.

Goal: minimize number of cache misses.

Question: which page to evict in case of a cache miss?
Known Results: Paging

Paging (Deterministic) [Sleator Tarjan 85]:
• Any online algorithm ≥ $k$-competitive.
• LRU is $k$-competitive (also other algorithms)
• LRU is $\frac{k}{(k-h+1)}$-competitive if optimal has cache of size $h \leq k$.

Paging (Randomized):
• Rand. Marking $O(\log k)$ [Fiat, Karp, Luby, McGeoch, Sleator, Young 91].
• Lower bound $H_k$ [Fiat et al. 91], tight results known.
• $O(\log(\frac{k}{k-h+1}))$-competitive algorithm if optimal has cache of size $h \leq k$ [Young 91].
The Weighted Paging Problem

One small change:

• Each page $i$ has a different fetching cost $w(i)$.
• Models scenarios where cost of loading pages into the cache is not uniform:
  
  Main memory, disk, internet ...

Goal

• Minimize the total cost of cache misses.
## Weighted Paging

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Randomized</th>
<th>Weighted Paging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower bound k</td>
<td>LRU k competitive</td>
<td>O(log k/(k-h+1))</td>
<td>k-competitive [Chrobak, Karloff, Payne, Vishwanathan 91]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>k/(k-h+1) [Young 94]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O(log k) for two distinct weights [Irani 02]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No o(k) algorithm known even for three distinct weights</td>
</tr>
</tbody>
</table>
The k-server Problem (1)

• k servers are placed in an n-point metric space

• requests arrive at points in the metric

• serving a request: move a server to request point

**Goal**: minimize total distance traveled by the servers.
The k-server Problem

• Paging = k-server on a uniform metric
  – every page is a point
  – A page is in the cache iff a server is at the point

• Weighted paging = k-server on a weighted star metric

Deterministic Results:
• General metric spaces: $(2k-1)$-competitive work function algorithm [Koutsoupias-Papadimitriou 95]

• Tree metric: $k$-competitive algorithm [Chrobak et al. 91]

Randomized Results:
• No $o(k)$ algorithm known (even for very simple spaces).
• Best lower bound $\Omega(\log k)$
Fractional Weighted Paging

Model:

- Fractions of pages are kept in cache: probability distribution over pages $p_1,\ldots,p_n$.
- The total sum of fractions of pages in the cache is at most $k$.
- If $p_i$ changes by $\varepsilon$, cost = $\varepsilon w(i)$.
Overview

High level idea:

1. Design a primal-dual $O(\log k)$-competitive algorithm for fractional weighted paging.

2. Obtain a randomized algorithm while losing only a constant factor.
We can only keep \( k \) pages out of the \( B(t) \) pages

\[
\text{Evict} \geq |B(t)| - 1 - (k - 1) = |B(t)| - k
\]

B(t): Set of pages requested until time t (including \( p_t \))
Weighted paging – Linear Program

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{r(i,t)} w(i) x(i, j)
\]

\[
\forall t \sum_{i \in B(t) \setminus \{p_t\}} x(i, r(i, t)) \geq |B(t)| - k
\]

\[
0 \leq x(i, j) \leq 1
\]

- Idea: charge for evicting pages instead of fetching pages

\(x(i,j)\) – indicator for the event that page \(i\) is evicted from the cache between the \(j\)-th and \((j+1)\)-st times it is requested

\(r(i,t)\) - number of times page \(i\) is requested till time \(t\), including \(t\)
Primal and Dual Programs

**P: Primal Covering**

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{r(i,t)} w(i) x(i, j)
\]

\[\forall t \sum_{i \in B(t) \setminus \{p_t\}} x(i, r(i, t)) \geq |B(t)| - k\]

\[0 \leq x(i, j) \leq 1\]

**D: Dual Packing**

\[
\max \sum_{t} \left( |B(t)| - k \right) y(t) - \sum_{i=1}^{n} \sum_{j=1}^{r(i,t)} z(i, j)
\]

For each page \(i\) and the \(j\)th time it was asked:

\[
\left( \sum_{t=t(i,j)+1}^{t(i,j+1)-1} y(t) \right) - z(i, j) \leq w(i)
\]
At time $t$, when page $p_t$ is requested:

- Set the new variable: $x(p_t, r(p_t, t)) \leftarrow 0$:
  - this guarantees that $p_t$ is in the cache at time $t$.
  - this variable can only be increased at times $t' > t$.

- If the primal constraint corresponding to time $t$ is satisfied, then do nothing.

- Else, increase variables $x(i, j)$ as a function of $y(t)$, details follow soon...
The growth function of $x(i,j)$

$$x(i, j)$$

- Dual is tight
- Dual violated by $O(\log k)$
- Page is “unmarked”
- Corresponding Dual constraint
- Page fully in memory (marked)
- Page fully evicted
• Else: increase primal and dual variables, till primal constraint corresponding to time $t$ is satisfied:

1. Increase variable $y(t)$ continuously; for each variable $x(p, j)$ that appears in the (yet unsatisfied) primal constraint that corresponds to time $t$:

2. If $x(p, j) = 1$, then increase $z(p, j)$ at the same rate as $y(t)$.

3. If $x(p, j) = 0$ and

$$\left( \sum_{t=t(p, j)+1}^{t(p, j+1)-1} y(t) \right) - z(p, j) = w(p),$$

then set $x(p, j) \leftarrow 1/k$.

4. If $1/k \leq x(p, j) < 1$, increase $x(p, j)$ by the following function:

$$\frac{1}{k} \cdot \exp \left( \frac{1}{w(p)} \left[ \left( \sum_{t=t(p, j)+1}^{t(p, j+1)-1} y(t) \right) - z(p, j) - w(p) \right] \right)$$
Analysis of Online Algorithm

Proof of competitive factor:
1. Primal solution is feasible.
2. Primal \leq 2 \cdot \text{Dual}
3. Dual is feasible up to a factor of O(log k)

Conclusion (weak duality):
Algorithm is $O(\log k)$-competitive
1. Primal solution is **feasible**. At time $t$:

- for page $p_t$, $x(p_t,r(p_t,t)) \leftarrow 0$, i.e., $p_t$ is in the cache

- primal variables $x(q,r(q,t))$ corresponding to other pages $q$ are increased till primal constraint is satisfied

- for each page $q$, by the algorithm, $x(q,r(q,t)) \leq 1$ (increase in $z$ balances out increase in $y$)
3. Dual is $O(\log k)$ feasible:

Consider any dual constraint. Since $x(i,j) \leq 1$:

$$1 \geq x(i, j) = \frac{1}{k}$$

Simplifying, we get that:

$$\left( \sum_{t=t(i,j)+1}^{t(i,j+1)-1} y(t) \right) - z(i, j) \leq w(i) \left[ 1 + \ln k \right]$$
Analysis of Online Algorithm

2. Primal $\leq 2 \cdot \text{Dual}$

This is done in two separate steps:

- $C_1$ - contribution to the primal cost of the variables $x(p,j)$ when increased from 0 to $1/k$

- $C_2$ - contribution to the primal cost of the variables $x(p,j)$ when increased from $1/k$ to (at most) 1, according to the exponential function

Each contribution is upper bounded separately by the dual
Bounding $C_1$

Define: $\tilde{x}(p, j) = \min(x(p, j), \frac{1}{k})$

Primal complementary slackness: if $\tilde{x}(p, j) > 0$,

$$\left( \sum_{t=t(p,j)+1}^{t(p,j+1)-1} y(t) \right) - z(p, j) \geq w(p)$$
Bounding $C_1$

- $B'(t)$ - set of pages $p \in B(t)$ for which $x(p, r(p, t)) = 1$

**Dual complementary slackness (1):** if $y(t)$ is being increased at time $t$ then:

$$\sum_{p \in B(t) \setminus (B'(t) \cup \{p_t\})} \tilde{x}(p, r(p, t)) \leq \frac{|B(t)| - 1 - |B'(t)|}{k} \leq |B(t)| - k - |B'(t)|$$

- $|B(t)| - |B'(t)| \geq k + 1$ (else $|B'(t)| \geq |B(t)| - k$, satisfying constraint)

- $\Rightarrow \frac{|B(t)| - 1 - |B'(t)|}{k} \leq |B(t)| - k - |B'(t)|$

**Dual complementary slackness (2):** if $z(p, j) > 0$, then $x(p, j) \geq 1$
Bounding $C_1$

$$\sum_{p=1}^{n} \sum_{j=1}^{r(p,t)} w(p) \tilde{x}(p, j) \leq$$

(by primal complementary slackness)

$$\leq \sum_{p=1}^{n} \sum_{j=1}^{r(p,t)} \left( \left( \sum_{t=t(p,j)+1}^{t(p,j+1)-1} y(t) \right) - z(p, j) \right) \tilde{x}(p, j) =$$

(changing order of summation)

$$= \sum_{t} \left( \sum_{i \in B(t) \setminus \{p_t\}} \tilde{x}(p, r(p, t)) \right) y(t) - \sum_{p=1}^{n} \sum_{j=1}^{r(p,t)} \tilde{x}(p, j) z(p, j)$$
Bounding $C_1$

\[
\sum_{t} \left( \sum_{i \in B(t) \setminus \{p_t\}} \tilde{x}(p, r(p, t)) \right) y(t) - \sum_{p=1}^{n} \sum_{j=1}^{r(p, t)} \tilde{x}(p, j) z(p, j)
\]

\[
\leq \sum_{t} (|B(t)| - k) y(t) - \sum_{p=1}^{n} \sum_{j=1}^{r(p, t)} z(p, j)
\]

- The derivative of the LHS is:

\[
\sum_{p \in B(t) \setminus (B'(t) \cup \{p_t\})} \tilde{x}(p, r(p, t)) \leq |B(t)| - k - |B'(t)|
\]

since $z(p, j)$ increases at the same rate as $y(t)$ when $x(p, r(p, t)) = 1$

- The derivative of the RHS is $|B(t)| - k - |B'(t)|$

Thus, $C_1$ is upper bounded by the dual solution
Bounding $C_2$

Reminder:

If $1/k \leq x(p, j) < 1$, increase $x(p, j)$ by the following function:

$$\frac{1}{k} \cdot \exp \left( \frac{1}{w(p)} \left[ \sum_{t = t(p, j) + 1}^{t(p, j+1) - 1} y(t) \right] - z(p, j) - w(p) \right)$$
Bounding $C_2$

Variables $y(t)$ and $z(p, j)$ are raised at rate 1 with respect to virtual variable $\tau$.

- $\frac{dy(t)}{d\tau} = 1$,
  $\frac{dx(p, j)}{dy(t)} = \frac{1}{w(p)} \cdot x(p, j)$

$$\frac{dC_2}{d\tau} = \sum_{p \in B(t) \setminus \{p_t\}, 1/k \leq x(p, j) < 1} w(p) \cdot \frac{dx(p, r(p, t))}{dy(t)} \cdot \frac{dy(t)}{d\tau}$$

$$= \sum_{p \in B(t) \setminus \{p_t\}, 1/k \leq x(p, j) < 1} x(p, r(p, t))$$

$$\leq (|B(t)| - k) \frac{dy(t)}{d\tau} - \sum_{p \in B(t) \setminus \{p_t\}, x(p, j) = 1} 1$$

$$\text{dual objective} = \sum_t (|B(t)| - k) y(t) - \sum_{p=1}^{n} \sum_{j=1}^{r(p, t)} z(p, j)$$
Conclusion

- $C_1$ is upper bounded by a dual solution
- $C_2$ is upper bounded by a dual solution

Thus, $\text{primal} \leq 2 \cdot \text{dual}$

The algorithm is $O(\log k)$-competitive
Rounding

Linear program provides a fractional view:

$$\text{Prob}[p \text{ is in cache at time } t] = 1 - x(p,r(p,t))$$

Randomized alg.: distribution on cache states

Example: pages A,B,C,D \( k=2 \)

LP state = \((1/2, 1/2, 1/2, 1/2)\)

Consistent distribution = \( \frac{1}{2} (A,B) + \frac{1}{2} (C,D) \)
Rounding – Need to be Careful

A, B have wt. 1, \quad C, D have wt. M

LP state = \( (1/2, 1/2, 1/2, 1/2) \)

Distribution = \( \frac{1}{2} \) (A,B) + \( \frac{1}{2} \) (C,D)

LP changes to (1,0,1/2,1/2)

LP cost = \( \frac{1}{2} \)

randomized algorithm: only consistent distribution = \( \frac{1}{2} \) (A,C) + \( \frac{1}{2} \) (A,D)

cost of randomized algorithm:

\( (\frac{1}{2} \) (A,B) + \( \frac{1}{2} \) (C,D)) \rightarrow (\frac{1}{2} \) (A,C) + \( \frac{1}{2} \) (A,D))

\( \Theta \) (M) – either C or D are (partly) evicted
Rounding – Main Ideas

- **Partition** the pages into weight classes:
  - class \( i \) pages with size \([2^i, 2^{i+1}]\)

- Define a **distribution** \( D \) on cache states
  - each cache state has *approximately* the same number of pages from each class.

- Show how to **update** the distribution on the cache states while paying at most 5 times the fractional cost.
Further Extensions of the Basic Model

First Extension:
• Pages have different fetching costs.
• Models scenarios in which the fetching cost is not uniform:
  Main memory, disk, internet …

Second (Orthogonal) Extension:
• Pages have different sizes.
• Models web-caching problems (Proxy Servers, local cache in browser)
# Caching Models

<table>
<thead>
<tr>
<th>Fetch cost</th>
<th>Uniform</th>
<th>Non-uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>Minimize number of times the user has to wait</td>
<td></td>
</tr>
<tr>
<td>Non-uniform</td>
<td>Weighted Caching</td>
<td></td>
</tr>
</tbody>
</table>

- **Bit Model** (fetching cost = size)
- **General Caching**

**Offline is NP-Hard** (Simple reduction from Knapsack/Partition)
## Deterministic Algorithms

<table>
<thead>
<tr>
<th>Fetch cost</th>
<th>Size</th>
<th>Uniform</th>
<th>Non-uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>Basic Caching</td>
<td>LRU is $k$-competitive (also other algorithms)</td>
<td>LRU is $k$-competitive [Irani]</td>
</tr>
</tbody>
</table>
| Non-uniform| Weighted Caching | $k$-competitive [Chrobak, Karloff, Payne, Vishwanathan] | 1. General Caching $k$-competitive [Irani, Cao], [Young]  
|            |           |                                  | 2. Bit Model LRU $k$-competitive [Irani] |

Any Algorithm $\geq k$-competitive
### Randomized Algorithms

<table>
<thead>
<tr>
<th>Fetch cost</th>
<th>Uniform</th>
<th>Non-uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>Basic Caching</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Randomized Marking</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$O(\log k)$-competitive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>[Fiat et al.]</td>
<td></td>
</tr>
<tr>
<td>Non-uniform</td>
<td>Weighted Caching</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$O(\log k)$-competitive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>algorithm [Bansal, Buchbinder, Naor]</td>
<td></td>
</tr>
</tbody>
</table>

- **Other algorithms that are optimal with constants**
- **Fault Model**
  - $O(\log^2 k)$-competitive algorithm [Irani]

1. **General Caching**
2. **Bit Model**
   - $O(\log^2 k)$-competitive algorithm [Irani]
## Improved Results

<table>
<thead>
<tr>
<th>Size</th>
<th>Uniform</th>
<th>Non-uniform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch cost</td>
<td><strong>Basic Caching</strong>&lt;br&gt;Randomized Marking&lt;br&gt;O(\log k)-competitive&lt;br&gt;[Fiat et al.]</td>
<td><strong>Fault Model</strong>&lt;br&gt;O(\log^2 k)-competitive&lt;br&gt;O(\log k)-competitive</td>
</tr>
<tr>
<td>Uniform</td>
<td>Weighted Caching&lt;br&gt;O(\log k)-competitive&lt;br&gt;algorithm [Bansal, Buchbinder, Naor]</td>
<td></td>
</tr>
<tr>
<td>Non-uniform</td>
<td></td>
<td>1. General Caching&lt;br&gt;O(\log^2 k)-competitive&lt;br&gt;O(\log k)-competitive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Bit Model&lt;br&gt;</td>
</tr>
</tbody>
</table>
Basic Definitions: Generalized Caching

- n pages
- Cache of size k
- Size of page \( p \): \( w_p \in [1,k] \)
- Fetching cost of page \( p \): \( c_p \) (arbitrary)

Fractional solution:

- Algorithm maintains fractions of pages as long as the total size does not exceed \( k \).
- Fetching \( \varepsilon \) fraction of page \( p \) costs \( \varepsilon c_p \)
High level approach

First step:
• General $O(\log k)$-competitive algorithm for the fractional generalized caching.
⇒ Maintains fractions on pages.

Second Step:
Transform \textbf{online} the fractional solution into Randomized algorithm:
• Maintain \textit{distribution on cache states} that is “consistent” with the fractional solution.
• Simulation procedure maps changes in fractions on pages to distribution on cache states (w/ similar cost).
• $O(1)$ simulation for Bit/Fault model
$O(\log k)$ simulation for the general model.
Generalized Caching – Linear Program

- Interval: Keep the page between the jth time it is requested and the (j+1) time it is requested.
- If interval present, no cache miss.
- At any time step t, **total size of intervals** (pages) is at most k.
Generalized Caching: 1st LP formulation

- \( x(p,j) \): How much of interval \((p,j)\) evicted thus far
- \( B(t) \): Set of pages requested until time \( t \).
- \( W(B(t)) \): total size of pages in \( B(t) \).
- \( r(p,t) \): number of times page \( p \) requested until time \( t \)

\[
\text{P: Primal Covering} \quad \min \sum_{p=1}^{n} \sum_{j=1}^{r(p,t)} c_p x(p, j)
\]

\[\forall t \quad \sum_{p \in B(t) \setminus \{p_t\}} w_p \cdot x(p, r(p,t)) \geq W(B(t)) - k\]

\[0 \leq x(p, j) \leq 1\]
Problem with LP formulation

The formulation has unbounded integrality gap …

Example:

• Two pages of size $k/2 + \varepsilon$ requested alternately.
• Integral solution: cache miss every turn
• Fractional solution:
  – Keeps almost one unit of each page.
  – Needs to fetch only $O(\varepsilon/k)$ page every turn

P: Primal Covering

$$
\min \sum_{p=1}^{n} \sum_{j=1}^{r(p,t)} c_p x(p, j) \\
\forall t \quad \sum_{p \in B(t) \setminus \{p_t\}} w_p \cdot x(p, r(p, t)) \geq W(B(t)) - k \\
0 \leq x(p, j) \leq 1
$$
Generalized Caching: 2nd LP formulation

Strengthening the LP:

**P: Primal Covering**

For any time $t$ and set $S$.

\[ \sum_{p \in B(t) \setminus \{p_t\}} \min \{ W(S)(p, k) \cdot w_p, Wx(S), r(k, t) \} \geq x(S) - k \]

\[ 0 \leq x(p, j) \leq 1 \]

**D: Dual Packing**

For each page $p$ and the $j$th time it is requested:

\[ \sum_{t=p(j+1)-1}^{t=p(j)+1} \min \{ W(s) - k, w_p \} \cdot y(t, S) \leq c_p \]

Observation: after strengthening, box constraints are redundant

These are called: knapsack inequalities
Sketch of Primal-Dual algorithm

- While there exists an **unsatisfied primal constraint** of set of pages $S$ and time $t$:
  - Increase the dual variable $y(t,S)$.
- When dual constraint of variable $x(p,j)$ is tight, $x(p,j) = 1/k$
  
  $$
  \sum_{t=t(p,j)+1}^{t(p,j+1)-1} \min\{W(s) - k, w_p\} \cdot y(t, S) = c_p
  $$

  From then on, increase $x(p,j)$ exponentially (until $x(p,j)=1$)

  $$
  x(p, j) = \left(\frac{1}{k}\right)^{t(p,j+1)-1} \exp \left[ \frac{1}{c_p} \sum_{t=t(p,j)+1}^{t(p,j+1)-1} \min\{W(s) - k, w_p\} \cdot y(t, S) \right] - 1
  $$
The growth function of $x(p,j)$

$$x(p,j)$$

Dual is tight

Dual violated by $O(\log k)$

Corresponding Dual constraint

Page fully in cache ("marked")

Page is "unmarked"

Page fully evacuated
Analysis of Online Algorithm

Proof of competitive factor:
1. Primal solution is feasible.
2. Primal ≤ 2 Dual.
3. Dual is feasible up to $O(\log k)$ factor

Conclusion (weak duality):
Algorithm is $O(\log k)$-competitive
Analysis - sketch

1. Primal solution is feasible.
   
   We increase $x(p,j)$’s until current primal constraint is feasible

2. Primal $\leq$ 2 Dual:
   
   a. Setting $x(p,j)$ to $1/k$ analyzed using **complementary slackness**
   
   b. During the **exponential** growth the **primal derivative is at most** dual derivative

3. Dual is $O(\log k)$ feasible:

   $$x(p, j) = \left(\frac{1}{k}\right)^{\exp\left[\frac{1}{c_p} \sum_{t=t(p,j)+1}^{t(p,j+1)-1} \min\{W(s) - k, w_p\} \cdot y(t, S)\right] - 1} \leq 1$$

   $$\sum_{t=t(p,j)+1}^{t(p,j+1)-1} \min\{W(s) - k, w_p\} \cdot y(t, S) \leq c_p \left(1 + \ln(k)\right)$$
Concluding Remarks

• Primal-dual approach gives simple unifying framework for caching.

Open questions:

1. Improving to $O(\log k)$ for the general model.
2. $o(k)$ randomized algorithms for k-server using primal-dual approach.
3. Extend primal-dual framework beyond packing/covering.