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

Primal-Dual Algorithms for Online Optimization: Lecture 1

Seffi Naor

Computer Science Dept.

Technion

Haifa, Israel

Introduction

- Online algorithms and competitive analysis
  - Deterministic
  - Randomized

- Review of important techniques
  - Randomized rounding
  - Dual fitting

What is an Online Algorithm?

- Input is given "in pieces" over time, each piece is called a "request"
- Request sequence: $\{\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n, \ldots\}$
- Upon arrival of request σ_i :
  - Online algorithm has to serve the request
  - Previous decisions for requests $\sigma_1, \ldots, \sigma_{i-1}$ cannot be changed

Performance Evaluation: Competitive Factor

- How to evaluate performance of online algorithm A?
- For every request sequence $\sigma = \sigma_1, \ldots, \sigma_n$:

  compare cost of A to the cost of an optimal offline algorithm that "knows" the request sequence <u>in advance</u>
- Competitive factor of online algorithm A is α if

  For every request sequence $\sigma = \sigma_1, \ldots, \sigma_n$:

$$\frac{A(\sigma_1, \ldots, \sigma_n)}{\mathrm{OPT}(\sigma_1, \ldots, \sigma_n)} \leq \alpha$$

Example 1:The Ski Rental Problem

- Buying costs \$B
- Renting costs \$1 per day

Problem:

Number of ski days is not known in advance – each ski day is a request served by buying or renting

Goal: Minimize the total cost.

Ski Rental: Analysis

- Online Algorithm: rent for m days and then buy

- What is the optimal choice of m? m=B

  - If # of ski days $\leq$ B, cost(online) = OPT
  - If # of ski days > B, cost(online) $\leq$ 2OPT

- $\Rightarrow$ Competitive ratio = 2

Example 2: The Paging Problem

Universe of 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?

Paging: Analysis

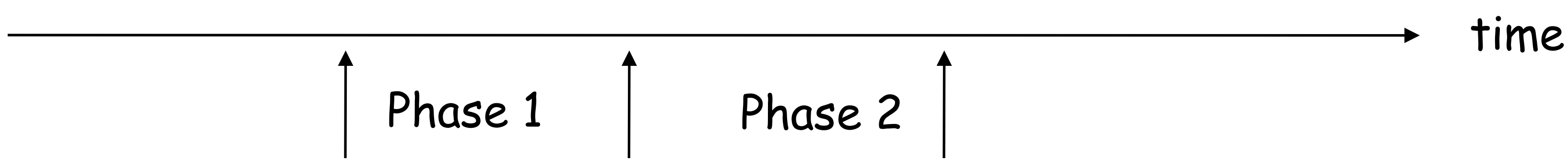
- Online Algorithm LRU (Least Recently Used):

  Upon cache miss, evict the page whose last access was earliest (least recently used page).

Theorem 1: The competitive factor of LRU is k.

Theorem 2: The competitive factor of any (deterministic) paging algorithm $\geq$ k.

Proof of Theorem 1



In each phase: LRU has precisely k misses

(OPT and LRU start from the same initial configuration)

Claim: Each phase has requests to k different pages

Proof: If the first miss in a phase is due to page x, then all k-1 remaining pages in the cache will be evicted before x, since x has a higher priority. □

Proof of Theorem 1 (contd.)

x - page that caused first miss in phase

S - pages that caused rest of cache misses in phase + page that caused first miss in next phase

By claim, $|S - \{x\}| \geq k$

Since page $x \in$ OPT's cache, OPT has a cache miss

p - number of full phases

cost of OPT $\geq p$

cost of LRU $\leq k(p+1)$ $\square$

Randomization

- Online algorithm A uses random bits $r=r_1, r_2, \ldots$
- Expected cost of A on σ: $\text{Exp}_r[A(\sigma,r)]$
- A is α-competitive if, $\forall\sigma$: $\frac{\text{Exp}_r A(\sigma, r)}{\text{OPT}(\sigma)} \leq \alpha$
- Oblivious adversary: knows online algorithm A, request sequence σ, but not the outcome of the random bits r

Example: Randomized Paging Algorithm

- Marking Algorithm:
  - Each requested page is marked
  - Page miss: evict one of the unmarked pages, chosen uniformly at random
  - When all pages are marked, unmark them

Theorem 1: The competitive factor of the Marking Algorithm is $2H_k$

Theorem 2: The competitive factor of any randomized paging algorithm is $H_k=\Omega(\log k)$

Set Cover

- Elements: U ={1,2, … ,n}
- Sets: $S_1,\ldots,S_m$ (each $S_i \subseteq$ {1,2, … ,n})
- Each set S_i has cost c_i
- Goal: find a min cost collection of sets that cover U

set cover can be formulated as integer/linear program:

x_i - indicator variable for choosing set S_i

$$\text{minimize} \quad \sum_{i=1}^{m} c_i x_i$$

$$\text{for every element } j\text{:} \quad \sum_{i \mid j \in S_i} x_i \geq 1$$

$$x_i \in \{0,1\}$$

Relaxation: $0 \leq x_i \leq 1$

- LP can be solved in poly time
- LP provides a lower bound on optimal integral solution!!!

Rounding a Fractional Solution (1)

- For each S_i: $0 \leq x_i \leq 1$
- For each element j: $\sum_i x_i \geq 1$ (summed over x_i, $j \in S_i$)

Randomized Rounding:

- For each set S_i: pick it to the cover with probability x_i

Analysis:

- Exp[cost of cover] = $\sum_i c_i x_i$ = LP cost
- Pr[element j is not covered] =

$$= \prod_{\ell | j \in S_\ell} (1 - x_\ell) \leq \left(1 - \frac{\sum_\ell x_\ell}{k}\right)^k \leq \left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}$$

Conclusion: probability of covering element j is at least a constant!

Rounding a Fractional Solution (2)

Amplify probability of success:

- Repeat experiment clogn times so that

$$\Pr[\text{element j is not covered}] \le \left(\frac{1}{e}\right)^{c\log n} \le \frac{1}{2n}$$

Analysis:

- $\Pr[\text{some element is not covered}] \le n \cdot \frac{1}{2n} \le \frac{1}{2}$
- Exp[cost of cover] = O(logn) $\sum_i c_i x_i$ = O(logn)(LP cost)

Conclusion: approximation factor is O(logn)

Set Cover: Greedy Algorithm

- Initially: C is empty
- While there is an uncovered element:
  - Add to C the set S_i minimizing

    c_i/(# new elements covered)

Analysis: via dual fitting

$$\text{minimize} \quad \sum_{i=1}^{m} c_i x_i \qquad\qquad \text{maximize} \quad \sum_{j=1}^{n} y_i$$

$$\text{for every element } j\text{:} \sum_{i|j \in S_i} x_i \geq 1 \qquad\qquad \text{for every set } S_i\text{:} \sum_{j \in S_i} y_j \leq c_i$$

$$x_i \geq 0 \qquad\qquad y_j \geq 0$$

Primal: covering $\geq$ Dual: packing

Dual Fitting (1)

- Primal solution is feasible
- Dual solution: if element j is covered by set S_i then

  $y_j = c_i/(\#$ new elements covered by $S_i)$
- In the iteration in which S_i is picked:

  Δ primal = Δ dual = c_i

  since cost of S_i is "shared" between the new elements
- Thus, cost of primal solution = cost of dual solution
- Is the dual solution feasible? Almost, but not quite …

Dual Fitting (2)

- For set S: suppose the order in which elements in S are covered is $e_1, \dots, e_k$
- When element e_i is covered, $y_{e_i} \leq \frac{c(S)}{k-i+1}$
- Thus, $\sum_{i=1}^{k} y_{e_i} \leq \sum_{i=1}^{k} \frac{c(S)}{k-i+1} \leq c(S) \cdot \sum_{i=1}^{k} \frac{1}{i} \leq c(S) \cdot H(k)$
- Dividing dual variables by $H(n) \approx \log n$ yields a feasible solution
- Greedy algorithm is an $O(\log n)$-approximation:

  primal $\leq$ dual $\times$ $H(n)$

The Online Primal-Dual Framework

- Introduction: covering
  - The ski problem
  - The online set cover problem

Back to Ski Rental

- Buying costs $B
- Renting costs $1 per day

Problem:

Number of ski days is not known in advance

Goal: Minimize the total cost.

Ski Rental – Integer Program

$$x = \begin{cases} 1 \text{ - Buy} \\ 0 \text{ - Don't Buy} \end{cases} \qquad z_i = \begin{cases} 1 \text{ - Rent on day i} \\ 0 \text{ - Don't rent on day i} \end{cases}$$

$$\min Bx + \sum_{i=1}^{k} z_i$$

Subject to:

For each day i: $x + z_i \geq 1$

$$x, z_i \in \{0,1\}$$

Ski Rental – Relaxation

P: Primal Covering

$$\min Bx + \sum_{i=1}^{k} z_i$$

For each day i: $x + z_i \geq 1$

$$x, z_i \geq 0$$

D: Dual Packing

$$\max \sum_{i=1}^{k} y_i$$

For each day i: $y_i \leq 1$

$$\sum_{i=1}^{k} y_i \leq B$$

Online setting:

- Primal: New constraints arrive one by one.
- Requirement: Upon arrival, constraints should be satisfied.
- Monotonicity: Variables can only be increased.

Ski Rental – Algorithm

P: Primal Covering

$$\min Bx + \sum_{i=1}^{k} z_i$$

For each day i: $x + z_i \geq 1$

$$x, z_i \geq 0$$

D: Dual Packing

$$\max \sum_{i=1}^{k} y_i$$

For each day i: $y_i \leq 1$

$$\sum_{i=1}^{k} y_i \leq B$$

Initially x← 0
Each new day (new constraint):
  if x<1:
- z_i ← 1-x
- x ← x(1+ 1/B) + 1/(c*B) - 'c' later.
- y_i ← 1

Analysis of Online Algorithm

Proof of competitive factor:

1. Primal solution is feasible.
2. In each iteration, $\Delta P \le (1 + 1/c)\Delta D$.
3. Dual is feasible.

Conclusion: Algorithm is (1+ 1/c)-competitive

Initially $x \leftarrow 0$

Each new day (new constraint):

if $x<1$:

- $z_i \leftarrow 1-x$
- $x \leftarrow x(1+ 1/B) + 1/(c*B)$ - 'c' later.
- $y_i \leftarrow 1$

Analysis of Online Algorithm

1. Primal solution is feasible.

   If $x \geq 1$ the solution is feasible.

   Otherwise set: $z_i \leftarrow 1-x$.

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

   If $x \geq 1$, $\Delta P = \Delta D = 0$

Otherwise:

- Change in dual: 1
- Change in primal:

  $B\Delta x + z_i = x + 1/c + 1 - x = 1 + 1/c$

Algorithm:

When new constraint arrives, if $x<1$:

$z_i \leftarrow 1-x$

$x \leftarrow x(1+ 1/B) + 1/c*B$

$y_i \leftarrow 1$

Analysis of Online Algorithm

3. Dual is feasible:

Need to prove: $\sum_{i=1}^{k} y_i \le B$

We prove that after B days x≥1

x is a sum of geometric sequence

$a_1 = 1/(cB)$, $q = 1+1/B$

$$x = \frac{1}{cB} \cdot \frac{\left(1+\frac{1}{B}\right)^B - 1}{\left(1+\frac{1}{B}\right) - 1} = \frac{\left(1+\frac{1}{B}\right)^B - 1}{c}$$

$$c \le \left(1+\frac{1}{B}\right)^B - 1 \approx e - 1$$

$$1 + \frac{1}{c} \approx \frac{e}{e-1}$$

Algorithm:

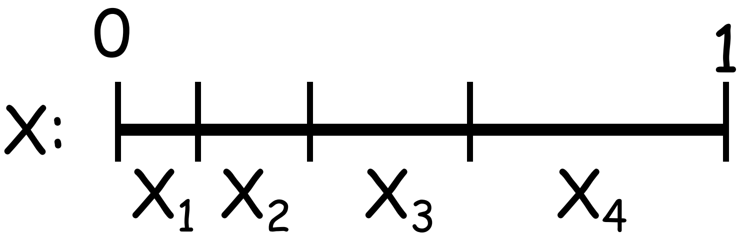
When new constraint arrives, if x<1:

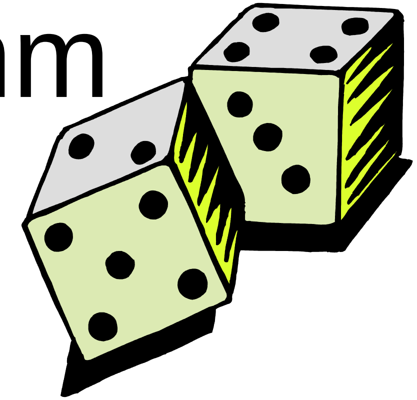
$z_i \leftarrow 1-x$

$x \leftarrow x(1+1/B) + 1/c*B$

$y_i \leftarrow 1$

Randomized Algorithm





- Choose *d* uniformly in [0,1]
- Buy on the day corresponding to the “bin” *d* falls in
- Rent up to that day

Analysis:

- Probability of buying on the *i*-th day is x_i
- Probability of renting on the *i*-th day is at most z_i

Going Beyond the Ski Problem

- Ski problem: coefficients in the constraint matrix belong to {0,1}
- What can be said about general constraint matrices with coefficients from {0,1}?

The Online Set-Cover Problem

- elements: $e_1, e_2, \ldots, e_n$
- set system: $s_1, s_2, \ldots s_m$
- costs: $c(s_1), c(s_2), \ldots c(s_m)$

Online Setting:

- Elements arrive one by one.
- Upon arrival elements need to be covered.
- Sets that are chosen cannot be “unchosen”.

Goal: Minimize the cost of the chosen sets.

Online Set-Cover: Lower Bound

- elements: 1, … ,*n*
- sets: all $\binom{n}{\sqrt{n}}$ subsets of cardinality √*n*
- cost: unit cost

Adversary's strategy:

- While possible: pick an element that is not covered
  (# of elements offered ≥ √n)

Competitive ratio: √n (cost of online: = √n, cost of OPT = 1)

But, … $\sqrt{n} \approx \log \binom{n}{\sqrt{n}}$. So polylog(*m*,*n*) is not ruled out.

Set Cover – Linear Program

P: Primal Covering

$$\min \sum_{s \in S} c(s)x(s)$$

$$\forall e \in E \quad \sum_{s | e \in s} x(s) \geq 1$$

D: Dual Packing

$$\max \sum_{e \in E} y(e)$$

$$\forall s \in S \quad \sum_{e \in s} y(e) \leq c(s)$$

Online setting:

- Primal: constraints arrive one by one.
- Requirement: each constraint is satisfied.
- Monotonicity: variables can only be increased.

Primal-Dual Algorithms

We will see two algorithms:

- “Discrete” algorithm – generalizing ideas from the ski problem
- “Continuous” algorithm

Set Cover – Discrete Algorithm

P: Primal Covering

$$\min \sum_{s \in S} c(s)x(s)$$

$$\forall e \in E \quad \sum_{s|e \in s} x(s) \geq 1$$

D: Dual Packing

$$\max \sum_{e \in E} y(e)$$

$$\forall s \in S \quad \sum_{e \in s} y(e) \leq c(s)$$

Initially $x(s) \leftarrow 0$

When new element arrives, while $\sum_{s|e \in s} x(s) < 1$:

- $y(e) \leftarrow y(e)+1$

$$\forall s | e \in s \quad x(s) \leftarrow x(s)\left(1+1/c(s)\right)+1/\left[m \cdot c(s)\right]$$

Analysis of Online Algorithm

Proof of competitive factor:

1. Primal solution is feasible.
2. In each iteration, $\Delta P \le 2\Delta D$.
3. Dual is (almost) feasible.

Conclusion: We will see later.

Initially $x(S) \leftarrow 0$

When new element e arrives, while $\sum_{s|e\in s} x(s) < 1$:

- $y(e) \leftarrow y(e)+1$

  $\forall s|e\in s \quad x(s) \leftarrow x(s)\left(1+1/c(s)\right)+1/\left[m\cdot c(s)\right]$

Analysis of Online Algorithm

1. Primal solution is feasible.

   We increase the primal variables until the constraint is feasible.

Initially x(S)← 0

When new element e arrives, while $\sum_{s|e\in s} x(s) < 1$:

- y(e) ← y(e)+1

  $\forall s|e \in s \quad x(s) \leftarrow x(s)\left(1+1/c(s)\right)+1/\left[m\cdot c(s)\right]$

Analysis of Online Algorithm

2. In each iteration, **$\Delta P \le 2\Delta D$.**

In each iteration:

- $\Delta D = 1$

$$\Delta P = \sum_{s|e\in s} c(s)\cdot \Delta x(s) = \sum_{s|e\in s} c(s)\cdot\left[\frac{x(s)}{c(s)} + \frac{1}{m\cdot c(s)}\right]$$

$$= \sum_{s|e\in s} x(s) + \sum_{s|e\in s} 1/m \le 2$$

Initially x(S)← 0

When new element e arrives, while $\sum_{s|e\in s} x(s) < 1$:

- y(e) ← y(e)+1

  $\forall s \,|\, e\in s \quad x(s) \leftarrow x(s)\left(1+1/c(s)\right) + 1/\left[m\cdot c(s)\right]$

Analysis of Online Algorithm

3. Dual is (almost) feasible:

- We prove that: $\forall s \in S, \quad \sum_{e \in S} y(e) \leq c(s) O(\log m)$
- If y(e) increases, then **x(s) increases** (for e in S).
- x(s) is a sum of a **geometric series**:

$a_1 = 1/[mc(s)]$, $q = (1 + 1/c(s))$

Initially $x(S) \leftarrow 0$

When new element e arrives, while $\sum_{s|e \in s} x(s) < 1$:

- $y(e) \leftarrow y(e)+1$

$\forall s | e \in s \quad x(s) \leftarrow x(s)\left(1 + 1/c(s)\right) + 1/\left[m \cdot c(s)\right]$

Analysis of Online Algorithm

- ➔ After c(s)O(log m) rounds:

$$x(s) = \frac{1}{m \cdot c(s)} \frac{\left(\left[1+1/c(s)\right]^{c(s)O(\log m)} - 1\right)}{\left[1+1/c(s)\right] - 1}$$

$$= \frac{\left(\left[1+1/c(s)\right]^{c(s)O(\log m)} - 1\right)}{m} \geq 1$$

We **never increase a variable x(s)>1!**

Initially x(S)← 0

When new element e arrives, while $\sum_{s|e\in s} x(s) < 1$:

- y(e) ← y(e)+1

  $\forall \mathrm{s} | \mathrm{e} \in \mathrm{s} \quad x(s) \leftarrow x(s)\left(1+1/c(s)\right) + 1/\left[m \cdot c(s)\right]$

Conclusions

- The dual is feasible with cost 1/O(log m) of the primal.

➔ The algorithm produces a fractional set cover that is O(log m)-competitive.

- **Remark:** no online algorithm can perform better (in the worst case).

Set Cover – Continuous Algorithm

Initially x(s)← 0, y(e)← 0

When new element e arrives:

While $\sum_{s|e\in s} x(s) < 1$:

- Increase variable y(e) continuously
- For each $s|e \in s$,

$$x(s) \leftarrow \frac{1}{m} \cdot \left[\exp\left(\frac{\ln(1+m)}{c(s)} \cdot \sum_{e' \in s} y(e') \right) - 1 \right]$$

Analysis of Online Algorithm

Proof of competitive factor:

1. Primal solution is feasible

2. In the iteration corresponding to element e:

$$\frac{\partial P}{\partial y(e)} \cdot \quad 2\ln(1+m) \cdot \frac{\partial D}{\partial y(e)}$$

3. Dual solution feasible

Analysis of Online Algorithm

1. Primal solution is feasible.

   We increase the primal variables until the constraint is feasible.

Analysis of Online Algorithm

2. In each iteration:

$$\frac{\partial P}{\partial y(e)} \cdot \quad 2\ln(1+m)\cdot\frac{\partial D}{\partial y(e)}$$

- Dual change: $\frac{\partial D}{\partial y(e)} = 1$

(variable y(e) is increased continuously)

Primal Change

$$\begin{aligned}
\frac{\partial P}{\partial y(e)} &= \sum_{s|e\in s} c(s)\frac{\partial x(s)}{\partial y(e)} \\
&= \sum_{s|e\in s} c(s)\left(\frac{\ln(1+m)}{c(s)}\cdot\frac{1}{m}\left[\exp\left(\frac{\ln(1+m)}{c(s)}\cdot\sum_{e'\in s} y(e')\right)\right]\right) \\
&= \ln(1+m)\cdot\sum_{s|e\in s}\left(\underbrace{\frac{1}{m}\left[\exp\left(\frac{\ln(1+m)}{c(s)}\cdot\sum_{e'\in s} y(e')\right)-1\right]}_{x(s)}+\frac{1}{m}\right) \\
&= \ln(1+m)\cdot\sum_{s|e\in s}\left(x(s)+\frac{1}{m}\right)\leq 2\ln(1+m).
\end{aligned}$$

$$\text{updating } s: \quad x(s)\leftarrow\frac{1}{m}\left[\exp\left(\frac{\ln(1+m)}{c(s)}\cdot\sum_{e'\in s} y(e')\right)-1\right]$$

Analysis of Online Algorithm

3. Dual is feasible:

- We prove that $\forall s: \quad \sum_{e' \in s} y(e') \leq c(s)$
- x(s) ≤ 1 (otherwise s satisfies the violated constraint for e)
- Hence,

$$x(s) = \frac{1}{m}\left[\exp\left(\frac{\ln(1+m)}{c(s)} \cdot \sum_{e' \in s} y(e')\right) - 1\right] \leq 1$$

$$\Rightarrow \quad \sum_{e' \in s} y(e') \leq c(s)$$

Conclusions

- The primal is feasible
- The dual is feasible
- The ratio between primal change and dual change is 1/O(log m)

➔ The algorithm produces a fractional set cover which is O(log m)-competitive.

Discrete vs. Continuous

- Both algorithms are essentially the same:

$$\left(1+\frac{1}{c(s)}\right) \approx \exp\left(\frac{1}{c(s)}\right)$$

as long as c(s) is not too small. (c(s) ≥1)

- Description of discrete algorithm is simpler

- Analysis of continuous algorithm is simpler

Summary: Key Idea for Primal-Dual Update

Primal: Min $\sum_i c_i x_i$

Dual: Max $\sum_t b_t y_t$

Step t, new constraint:

$a_1x_1 + a_2x_2 + \ldots + a_jx_j \geq b_t$

New variable y_t

$+ b_t y_t$ in dual objective

$x_i \leftarrow (1+ a_i/c_i)\, x_i$ (mult. update)

$y_t \leftarrow y_t + 1$ (additive update)

$$\Delta \text{ primal cost} = \sum_i c_i(\Delta x_i) = \sum_i c_i \left(\frac{a_i x_i}{c_i}\right)$$

$$= \sum_i a_i x_i \leq b_t = \Delta \text{ Dual Cost}$$

Online Randomized Rounding

What about an integral solution?

- Round fractional solution:
  - For set s, choose it with probability $\Delta x(s)$ when incrementing variable $x(s)$
  - Repeat $O(\log n)$ times to amplify success probability

- Competitive ratio is $O(\log m \log n)$

- Can be done deterministically online [AAABN03].