Introduction to online algorithms and the multi-level aggregation problem

Christoph Dürr
CNRS, LIP6, Université Pierre et Marie Curie, Paris 6

Chalmers, 12/2015
My favorite story about how the world was created
The online setting

- The input is revealed to the algorithm in form of a request sequence (≠ offline algorithm: input-compute-output)
- Each request has to be served with an irrevocable decision (≠ dynamic data structure)
- Some objective has to be minimized
- Performance is measured by competitive ratio. Algorithm A is c-competitive if for all request sequences σ
  \[ A(\sigma) \leq c \cdot OPT(\sigma) + \text{constant} \]
  (≠ price of not knowing the future)
- Game between algorithm (make decisions to keep cost small) and adversary (generate request sequence to make cost big)
ski rental problem

- Request sequence = ski,ski,...,ski
  (length n unknown in advance)

- decision: rent (1€) or buy (b€)

- \( \text{OPT} = \min(n,b) \)

- deterministic \( \text{ALG} = \text{rent until day } t-1, \text{on day } t \text{ buy} \)

- worst input length is \( n=t \), giving ratio
  \[
  \frac{t - 1 + b}{\min(t,b)} = \frac{\min(t,b) + \max(t,b) - 1}{\min(t,b)} = 1 + \frac{\max(t,b) - 1}{\min(t,b)}
  \]

- best parameter \( t = b \), giving ratio \( 2-1/b \)

- randomized algorithm achieves \( e/(e-1) < 1.582 \)
secretary problem: definition

- Input is a sequence of $n$ distinct integers $n$ is known, order is chosen uniformly at random
- On request $x$, algorithm can either reject or accept (and game ends)
- Goal: accept the minimum integer with high probability
- ALG: reject first $n/e$ entries, then accept first entry that is smaller than anything seen so far

Claim: probability of success is $1/e > 0.36$
secretary problem: analysis

- Input (ranks) is a permutation $\sigma$ on 1,...,n
- set $t=n/e$
- Probability algorithm succeeds is

$$\sum_{j=t+1}^{n} \mathbb{P}[\sigma[j] = 1 \text{ and minimum of } \sigma[1], \ldots, \sigma[j-1] \text{ is in } \sigma[1], \ldots, \sigma[t]]$$

which is maximized by $t=n/e$ evaluating to $1/e$
cow path problem:
definition

- there is juicy grass on the other side of the fence
- cow is at position 0,
  fence has an opening at position \(x\) with \(|x| \geq 1\)
  (\(\text{sign}(x)\) is unknown to the cow)
- doubling ALG: walk to \(+1,-2,+4,-8,+16,-32,\ldots\)
- competitive ratio: = distance walked to opening / \(|x|\)
cow path problem: analysis

• worst case: $|x| = 2^i + \varepsilon$

• cost of ALG:
  
  \[ 2(1 + 2 + 4 + \ldots + 2^{i+1}) + 2^i + \varepsilon \]
  
  \[ = 2(2^{i+2} - 1) + 2^i + \varepsilon \]
  
  \[ < 8\cdot 2^i + 2^i + \varepsilon \]
  
  \[ < 9\cdot \text{OPT} \]
Multi-Level Aggregation

Marcin Bienkowski
Martin Böhm
Jaroslaw Byrka
Marek Chrobak
C.D.
Łukasz Folwarczný
Łukasz Jeż
Jiří Sgall
Nguyễn Kim Thăng
Pavel Veselý
1-level aggregation: TCP acknowledgement

- Minimize number of acknowledgements + total waiting cost
- Offline: optimal dynamic programming solution in time $O(n \log n)$
- Online: deterministic ratio = 2, randomized ratio = $1/e$
  (similar to ski rental: send acknowledgement as soon as total waiting time reaches 1)
- The deadline variant is trivial: acknowledge as soon as a deadline of a packet expires

studied since 1950’s as lot sizing
requests arrive at leafs

**general model:** a request comes with an increasing waiting function of the serving time, generally just - arrival time

**deadline model:** a request comes with a strict deadline and issues no waiting cost

they are served by buying a **subtree** containing them

NP-hard, even APX-hard
A simple 2-competitive algorithm:
As soon as some request reaches its deadline, serve at the same time a set of most urgent requests $S$ with $\sum_{i \in S} C_i \simeq C$

- this is optimal
L-decreasing instances

Definition: instance $I$ is L-decreasing if for every vertex $v$ and its directed ancestor $u$ we have $C_u \geq L \cdot C_v$.

- We can construct an instance $I'$ which is L-decreasing by replacing edges with shortcuts to first heavy enough ancestors (or to root if none).

Lemma: Every R-competitive algorithm on $I'$ is DLR-competitive on $I$. 

$C_u, C_v$
Definition: instance $I$ is $L$-decreasing if for every vertex $v$ and its directed ancestor $u$ we have $C_u \geq L \cdot C_v$

We can construct an instance $I'$ which is $L$-decreasing by replacing edges with shortcuts to closest heavy enough ancestors (or to root if none).

Lemma: Every $R$-competitive algorithm on $I'$ is DLR-competitive on $I$
Deadline model: a $D^22^D$-competitive algorithm

- Generalizing from 2-level trees, we add to service tree level 2 nodes of total weight $\approx C$. 

\[ C_u \quad C_v \]

- Answer: do both
Deadline model: a $D^22^D$-competitive algorithm

- Generalizing from 2-level trees, we add to service tree level 2 nodes of total weight $\approx C$. But then should we

1. add for each of these nodes $u$ a set of most urgent leafs from subtree rooted at $u$ of cost $\approx Cu$

2. or add a set of most urgent leafs from those subtrees with total weight $C$?
Deadline model: a $D^22^D$-competitive algorithm

- Generalizing from 2-level trees, we add to service tree level 2 nodes of total weight $\approx C$. But then should we

  1. add for each of these nodes $u$ a set of most urgent leafs from subtree rooted at $u$ of cost $\approx Cu$

  2. or add a set of most urgent leafs from those subtrees with total weight $C$?

- Answer: do both.
Deadline model:
a $D^22^D$-competitive algorithm

- Generalizing from 2-level trees, we add to service tree level 2 nodes of total weight $\approx C$. But then should we
  1. add for each of these nodes $u$ a set of most urgent leafs from subtree rooted at $u$ of cost $\approx Cu$
  2. or add a set of most urgent leafs from those subtrees with total weight $C$?
- Answer: do both.
Deadline model: a $D^22^D$-competitive algorithm

- **Urgency** of a vertex $u$ = smallest deadline among requests in subtree rooted at $u$

- **Urgent**$(S,C)$ = smallest set of most urgent vertices from $S$ with cost at least $C$ (or $S$)

- **ALG:**
  
  init $X = \{\text{root } r, \text{ descendant of root } q\}$

  for each level $i=2, \ldots, D$:

  $Z^i = \text{all children of nodes in } X^{i-1}$

  for each $v$ in $X^{<i}$:

  add Urgent$\left(Z^i, C_v\right)$ to $X$
Deadline model: a $D^22^D$-competitive algorithm

- Analysis: show by induction that for $i=2,\ldots,D$:
  \[\text{cost}(X \leq i) \leq (2+1/L)^{i-1} C_q\]

- Then the algorithm is $(2+1/L)^{D-1}$ competitive on $L$-decreasing trees

- Choosing $L=D/2$, the algorithm is $D^22^D$ competitive on general trees
**competitive ratio:**

**our results**

<table>
<thead>
<tr>
<th></th>
<th>linear or general waiting cost</th>
<th>deadline variant</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>upper</td>
<td>lower</td>
</tr>
<tr>
<td>depth 1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>depth 2</td>
<td>3</td>
<td>2.754</td>
</tr>
<tr>
<td>depth $D \geq 2$</td>
<td>$O(D^4 2^D)$</td>
<td>2.754</td>
</tr>
<tr>
<td>paths of arbitrary depth</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>
Thank you