Designing Competitive Online Algorithms: Greediness and **Regret**

Mário César San Felice

Professor at Department of Computing - University of São Carlos

3rd Workshop Paulista on Optimization, Combinatorics and Algorithms

September 7, 2019

Online Problems and Competitive Analysis

Input parts arrive one at a time

Each part is served before next one arrives

No decision can be changed in the future

An online algorithm ALG is c-competitive if

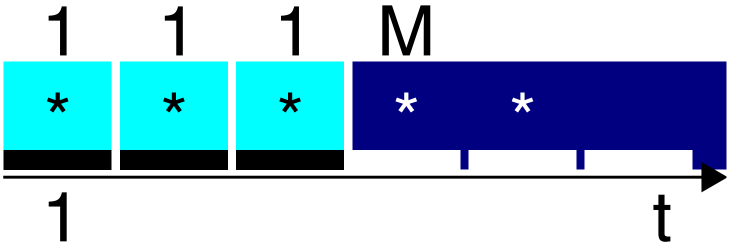
$$\mathrm{ALG}(I) \leq c\,\mathrm{OPT}(I)$$

for every input I

As an example, lets take the Ski Rental problem

Ski Rental Problem

Input: time horizon, skis buying price M (renting cost is 1 per day), list informing when snow melts



minimize sum of renting days plus M (if we decide to buy skis)

How to solve the offline version of this problem?

Does a greedy algorithm solve its online version?

Ski Rental Application and Generalization

Ski rental algorithms are useful to save energy

Help to decide when to turn off parts of a system

Like cores in a processor or computers in a cluster

Generalized into Parking Permit Problem [Meyerson 2005]

Quintessential both to theoretical and practical leasing problems,

- in which resources are leased instead of permanently acquired

Dealing with Regret: Change

When you realize that a course of actions was wrong,

take the better course in retrospect,

even if you have to pay a price for it

Ski Rental Algorithm

Rent for the first $M - 1$ days, buy in the M-th day

Algorithm 1: Intuitive SR Algorithm

Input: M

Set day j and total renting cost r to 0;

while *a new snow day happens* **do**

  if $r + 1 < M$ **then**

    Rent skis at day j and $r \leftarrow r + 1$;

  else

    Buy skis if still don't have them;

  $j \leftarrow j + 1$;

The algorithm chose greedily to rent, until buying being better

This algorithm is 2-competitive. Why?

Ski Rental LP Formulations

Linear programming relaxation

$$\begin{aligned} \min\ & Mx + \textstyle\sum_{j=1}^{n} y_j \\ \text{s.t.}\ & x + y_j \geq 1 \text{ for } j = 1, \ldots, n \\ & x \geq 0, y_j \geq 0 \text{ for } j = 1, \ldots, n \end{aligned}$$

(covering problem: constraints arrive online)

and its dual

$$\begin{aligned} \max\ & \textstyle\sum_{j=1}^{n} \alpha_j \\ \text{s.t.} & \textstyle\sum_{j=1}^{n} \alpha_j \leq M \\ & 0 \leq \alpha_j \leq 1 \text{ for } j = 1, \ldots, n \end{aligned}$$

(packing problem: variables arrive online)

Primal-Dual Ski Rental Algorithm

Algorithm 2: Primal-Dual SR Algorithm

Input: M

Set day j' to 0;

while *a new snow day happens* **do**

  increase $\alpha_{j'}$ until one of the following happens:

  (a) $\alpha_{j'} = 1$; /* rent skis setting $y_{j'} = 1$ */

  (b) $M = \alpha_{j'} + \sum_{j=1}^{j'-1} \alpha_j$; /* buy skis setting $x = 1$ */

  $j' \leftarrow j' + 1$;

Is it similar to the previous algorithm?

Primal-Dual SR Algorithm is 2-Competitive

Note that, $Mx \leq \sum_{j=1}^{n} \alpha_j$ and that $y_j \leq \alpha_j$ for any j

Moreover, our dual solution is feasible and,

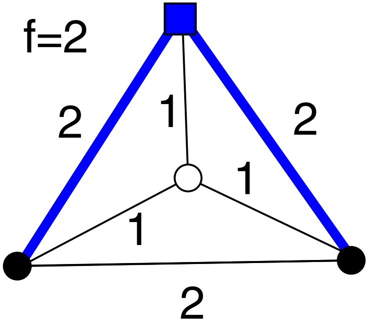
due to weak duality, any dual feasible solution costs at most OPT

Thus

$$
\begin{aligned}
ALG &= Mx + \sum_{j=1}^{n} y_j \\
&\leq \sum_{j=1}^{n} \alpha_j + \sum_{j=1}^{n} \alpha_j \\
&\leq 2\mathrm{OPT}
\end{aligned}
$$

Online Facility Location Problem

Input: $G = (V, E)$, $d : E \to \mathbb{R}^+$, $f : V \to \mathbb{R}^+$, clients $D \subseteq V$



$$\min \sum_{i \in F^a} f(i) + \sum_{j \in D} d(j, a(j))$$

Total cost $= 2 + 2 + 2 = 6$.

How a greedy algorithm would behave? What is its worst case?

Online Facility Location LP Formulation

Linear programming relaxation

$$
\begin{array}{rll}
\min & \sum_{i\in F} f(i)y_i + \sum_{j\in D}\sum_{i\in F} d(j,i)x_{ji} & \\
\text{s.t.} & x_{ji} \le y_i & \text{for } j \in D \text{ and } i \in F \\
& \sum_{i\in F} x_{ji} \ge 1 & \text{for } j \in D \\
& y_i \ge 0, x_{ji} \ge 0 & \text{for } j \in D \text{ and } i \in F
\end{array}
$$

and its dual

$$
\begin{array}{rll}
\max & \sum_{j\in D} \alpha_j & \\
\text{s.t.} & \sum_{j\in D}(\alpha_j - d(j,i))^+ \le f(i) & \text{for } i \in F \\
& \alpha_j \ge 0 & \text{for } j \in D
\end{array}
$$

Online Facility Location Algorithm

Algorithm 3: OFL Algorithm

Input: (G, d, f, F)
$F^a \leftarrow \emptyset$; $D \leftarrow \emptyset$;
while *a new client* j' *arrives* **do**
  increase $\alpha_{j'}$ until one of the following happens:
  (a) $\alpha_{j'} = d(j', i)$ for some $i \in F^a$; /* connect only */
  (b) $f(i) = (\alpha_{j'} - d(j', i)) + \sum_{j \in D}(d(j, F^a) - d(j, i))^+$ for some $i \in F \setminus F^a$; /* open and connect */
  $F^a \leftarrow F^a \cup \{i\}$; $D \leftarrow D \cup \{j'\}$; $a(j') \leftarrow i$;
return (F^a, a);

Algorithm is $(4 \ln n)$-competitive

Lemma 1: $\mathrm{ALG} \leq 2 \sum_{j \in D} \alpha_j$

Lemma 2: $\sum_{j \in D} \left(\frac{\alpha_j}{2H_{|D|}} - d(j,i) \right) \leq f_i$, for any $i \in F$

Using Lemmas 1 and 2, we can prove the main result

$$\begin{aligned} \mathrm{ALG} &\leq 2 \sum_{j \in D} \alpha_j \\ &= 4H_{|D|} \sum_{j \in D} \frac{\alpha_j}{2H_{|D|}} \\ &\leq 4H_{|D|} \mathrm{OPT} \\ &\leq 4 \ln n \, \mathrm{OPT} \end{aligned}$$

Result due to [Fotakis 2007] and [Nagarajan and Williamson 2013]

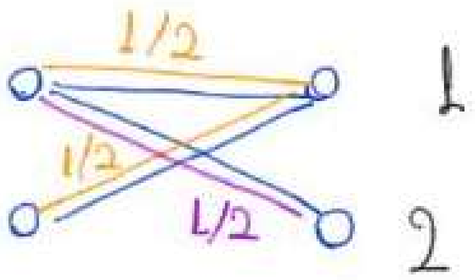
Dealing with Regret: Avoidance

When you don't know the best choice,

do not compromise,

by using continuous variables and randomness

As an example, consider the online bipartite matching worst case



Recalling the Ski Rental LP Formulations

Linear programming relaxation

$$\begin{aligned} \min\ & Mx + \textstyle\sum_{j=1}^{n} y_j \\ \text{s.t. } & x + y_j \geq 1 \text{ for } j = 1, \ldots, n \\ & x \geq 0, y_j \geq 0 \text{ for } j = 1, \ldots, n \end{aligned}$$

(covering problem: constraints arrive online)

and its dual

$$\begin{aligned} \max\ & \textstyle\sum_{j=1}^{n} \alpha_j \\ \text{s.t.} & \textstyle\sum_{j=1}^{n} \alpha_j \leq M \\ & 0 \leq \alpha_j \leq 1 \text{ for } j = 1, \ldots, n \end{aligned}$$

(packing problem: variables arrive online)

Fractional Ski Rental Algorithm

Constraint must be satisfied as they arrive

and variables can only increase in value

Algorithm 4: Fractional SR Algorithm

Input: M

while *a new snow day* j' *happens* **do**

  if $x < 1$ **then**

    $y_{j'} \leftarrow 1 - x$

    $x \leftarrow x\left(1 + \frac{1}{M}\right) + \frac{1}{cM}$

    $\alpha_{j'} \leftarrow 1$

Constant c will be define later

Analyzing Fractional Algorithm

Let P be the cost of the primal solution and D the dual one

Proof relies on the following three steps

- P is feasible
- In each iteration, $\Delta P \leq (1 + 1/c)\Delta D$
- D is feasible

Notice that, we are still relying on the primal-dual relation to obtain the bounds

Analyzing Fractional Algorithm

Since primal feasibility constraint is $x + y_j \geq 1$,

- P is feasible because either $x = 1$ or $y_j = 1 - x$

Since primal objective function is $Mx + \sum_{j=1}^{n} y_j$,

- $\Delta P = M\frac{x}{M} + M\frac{1}{cM} + 1 - x = 1 + \frac{1}{c}$

Since dual objective function is $\sum_{j=1}^{n} \alpha_j$,

- $\Delta D = 1$

Since dual feasibility constraint is $\sum_{j=1}^{n} \alpha_j \leq M$,

- We need to show that after M days $x \geq 1$

Analyzing Fractional Algorithm

Since dual feasibility constraint is $\sum_{j=1}^{n} \alpha_j \leq M$,

- we need to show that after M days $x \geq 1$

Since at each new day $x \leftarrow x\left(1 + \frac{1}{M}\right) + \frac{1}{cM}$,

- x value corresponds to the sum of a geometric progression
  $x_0 = \frac{1}{cM}$ $\quad x_1 = \frac{1}{cM}\left(1 + \frac{1}{M}\right) + \frac{1}{cM}$
  $x_2 = \frac{1}{cM}\left(1 + \frac{1}{M}\right)^2 + \frac{1}{cM}\left(1 + \frac{1}{M}\right) + \frac{1}{cM}$
- with initial term $\frac{1}{cM}$ and ratio $\left(1 + \frac{1}{M}\right)$

$$x = \frac{1}{cM}\frac{(1+1/M)^M - 1}{(1+1/M) - 1} = \frac{(1+1/M)^M - 1}{c} \geq 1$$

Since $(1+1/M)^M \simeq \mathrm{e}$ we have $c \leq \mathrm{e} - 1$ and $1 + \frac{1}{c} = \frac{\mathrm{e}}{\mathrm{e}-1}$

Analyzing Fractional Algorithm

Thus, we have a $\frac{e}{e-1}$-competitive algorithm,

- but it is for the fractional version of the problem

We use randomization to obtain an algorithm for the discrete problem

In particular, we use the increment of x on a day

- as the probability that the algorithm will buy at that day