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 6, 2019

Combinatorial Optimization Problems

Maximization or minimization problems

Algorithm receives an input

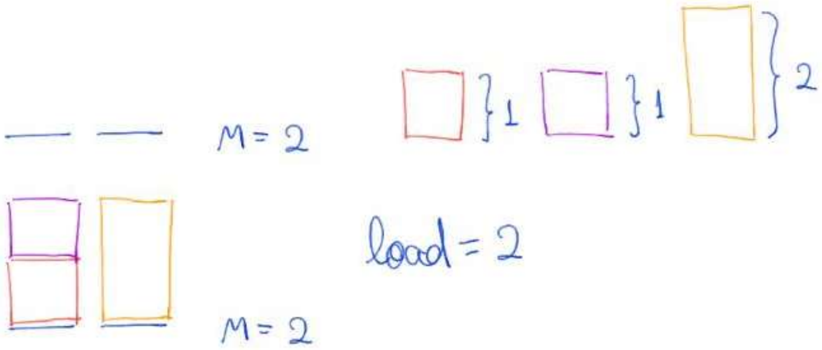
Returns a solution with a cost

As an example, lets take the Load Balancing problem

Load Balancing problem

Input: machines M, tasks D, sizes $s : D \to \mathbb{R}^+$

$$l(i) = \sum_{j \in D : a(j) = i} s(j) \qquad \min \max_{i=1}^{M} l(i)$$



Online Problems

Input parts arrive one at a time

Each part is served before next one arrives

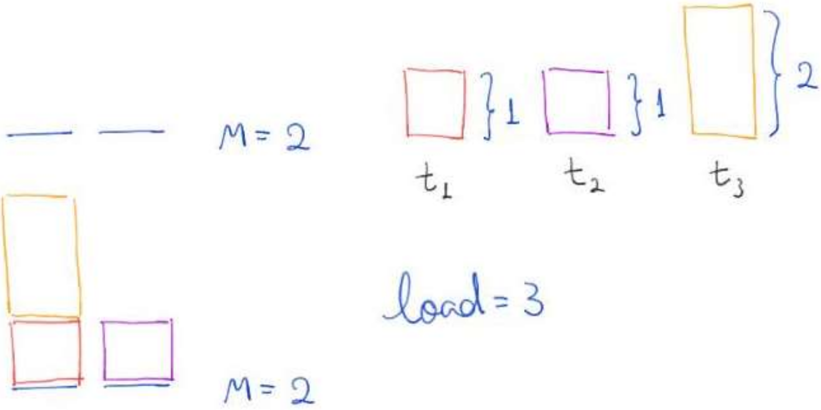
No decision can be changed in the future

As an example, lets take the Online Load Balancing (OLB) problem

Online Load Balancing problem

Itens arrive one at a time

$$\min \max_{i=1}^{M} l(i)$$



Competitive Analysis

Worst case analysis technique

For online algorithm ALG

Using offline optimal solution OPT

ALG is c-competitive if

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

for every input I

As an example, lets take a greedy online algorithm for the OLB

Greedy Online Load Balancing Algorithm

Always choose the machine with minimum load

Algorithm 1: OLB Algorithm

Input: M

For each machine $i = 1, \ldots, M$ set its load $l(i)$ to 0;

$i^* \leftarrow 1$;

while *a new task j arrives* **do**

    $a(j) \leftarrow i^*$;

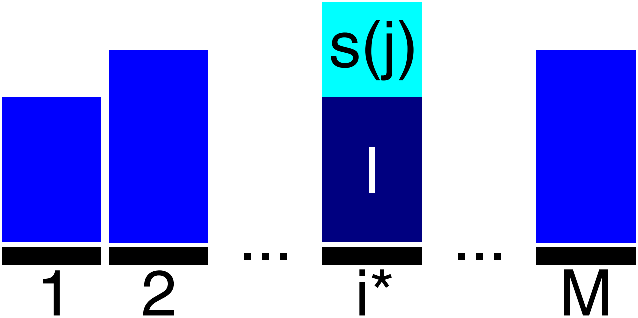
    $l(i^*) \leftarrow l(i^*) + s(j)$;

    choose machine with minimum load as new i^*;

return a;

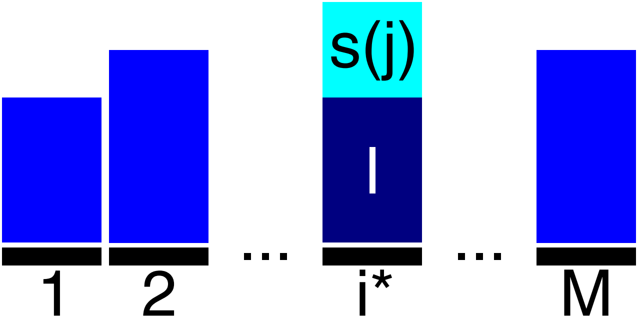
Analyzing Greedy Algorithm

Let i^* be the machine with maximum load, j be the last task assigned to i^*, and $l(i^*) = l + s(j)$



The ideia is to upper bound $s(j)$ and l using OPT

Auxiliary Results



Lemma 1: $\mathrm{OPT} \geq \max_{j' \in D} s(j') \geq s(j)$

Lemma 2: $\mathrm{OPT} \geq \frac{1}{M} \sum_{j' \in D} s(j') \geq \frac{1}{M} \sum_{j'=1}^{j} s(j')$
$= \frac{1}{M} \sum_{j'=1}^{j-1} s(j') + \frac{s(j)}{M} \geq l + \frac{s(j)}{M}$

Greedy Algorithm is $\left(2 - \frac{1}{M}\right)$-competitive

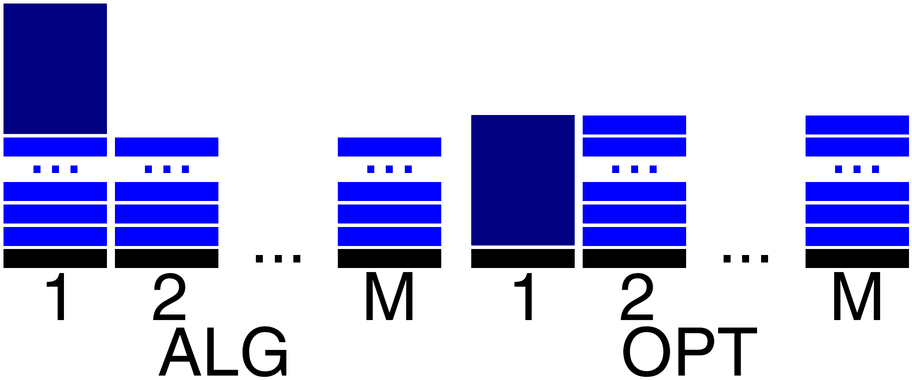
Since $\mathrm{OPT} \geq s(j)$ and $\mathrm{OPT} \geq l + \frac{s(j)}{M}$, we have

$$\begin{aligned} \mathrm{ALG} &= l + s(j) \\ &\leq \mathrm{OPT} - \frac{s(j)}{M} + s(j) \\ &\leq \mathrm{OPT} + \left(1 - \frac{1}{M}\right)\mathrm{OPT} \\ &= \left(2 - \frac{1}{M}\right)\mathrm{OPT} \end{aligned}$$

Thus, the greedy algorithm is 2-competitive

Lower Bound for Greedy Algorithm

List with $M(M-1)$ size 1 tasks followed by one size M task



We have $\mathrm{ALG} = 2M - 1$ and $\mathrm{OPT} = M$,

- i.e., $\frac{\mathrm{ALG}}{\mathrm{OPT}} = \frac{2M-1}{M} = 2 - \frac{1}{M}$

Two Interesting Special Cases

Since $\mathrm{OPT} \geq s(j)$ and $\mathrm{OPT} \geq l + \frac{s(j)}{M}$, we have

$$\mathrm{ALG} = l + s(j) \leq \left(2 - \frac{1}{M}\right) \mathrm{OPT}$$

Few machines special case: If $M = 2$ then ALG is $\frac{3}{2}$-competitive

Small items special case: If all items are smaller than $\alpha \, \mathrm{OPT}$ then

$$\mathrm{ALG} = l + s(j) \leq (1 + \alpha) \, \mathrm{OPT}$$

Areas of Interest

Online problems capture uncertainty over time

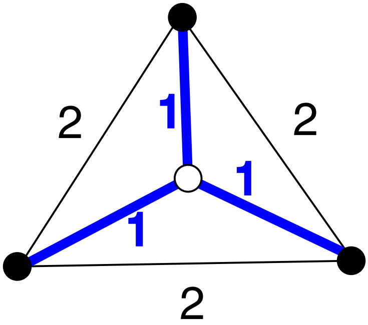
Common in operations research and computer science:

- Resource management: scheduling, packing, load balancing
- Dynamic data structures: list access problem
- Memory management: paging problem
- Sustainability: ski-rental problem
- Network design: Steiner tree, facility location

Greedy is a natural approach, since you are necessarily myopic

Steiner Tree Problem

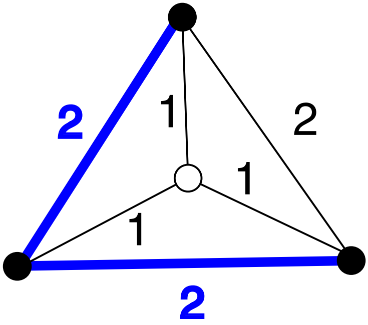
Input: $G = (V, E)$, $d : E \to \mathbb{R}^+$, terminals $D \subseteq V$



$$\min \sum_{e \in T} d(e) = 3$$

Online Steiner Tree Problem

Terminal nodes arrive one at a time



$$\min \sum_{e \in T} d(e) = 2 + 2 = 4$$

We are focusing on the metric completion special case.

- Why is this without loss of generality?

Greedy Online Steiner Tree Algorithm

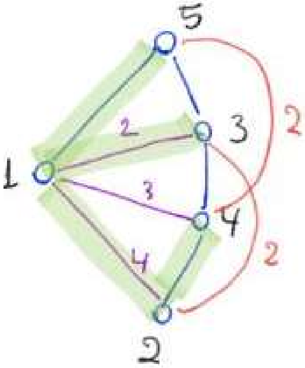
Connects the current terminal to the closest terminal

Algorithm 2: OST Algorithm

Input: (G, d)
$T \leftarrow (\emptyset, \emptyset)$;
while *a new terminal j arrives* **do**
    $T \leftarrow T \cup \{(j, V(T))\}$;
return T;

Lower Bound for Greedy Algorithm

Lets design a worst case example for the greedy algorithm.

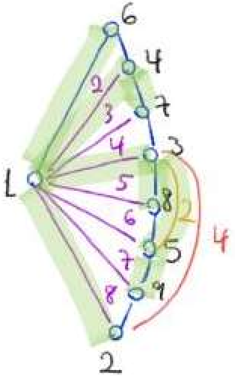


$$OPT = 4$$

$$ALG = 4 + 2 + \underbrace{2 * 1} = 8$$

$$OPT + \underbrace{\frac{OPT}{2} + \frac{OPT}{2}}_{\frac{OPT}{2} * \lg n}$$

Generalizing this worst case on a lager graph.



$$OPT = 8$$

$$ALG = 8 + 4 + \underbrace{2 * 2} + \underbrace{4 * 1} = 20$$

$$\underbrace{OPT + \frac{OPT}{2} + \frac{OPT}{2} + \frac{OPT}{2}}_{\frac{OPT}{2} * \lg n}$$

Greedy Algorithm is $(2 \ln k)$-competitive

Lemma 1: The i-th most expensive edge costs at most $\frac{2\mathrm{OPT}}{i}$

Using Lemma 1, we can prove the main result

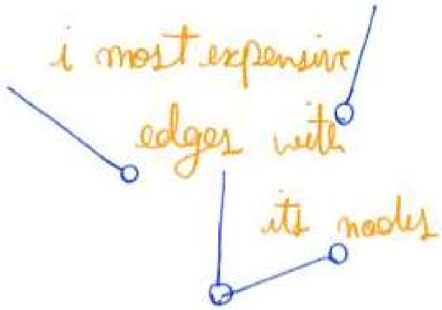
$$\begin{aligned}
\mathrm{ALG} &\leq 2\mathrm{OPT} + \mathrm{OPT} + \frac{2}{3}\mathrm{OPT} + \dots \\
&= \sum_{i=1}^{k-1} \frac{2\mathrm{OPT}}{i} \\
&= 2\mathrm{OPT} \sum_{i=1}^{k-1} \frac{1}{i} \\
&= 2\mathrm{OPT} H_{k-1} \\
&\leq 2 \ln k \mathrm{OPT}
\end{aligned}$$

Proving Auxiliary Result

Lemma 1: The i-th most expensive edge costs at most $\frac{2\mathrm{OPT}}{i}$

Associate each edge with the vertex it connected to the tree

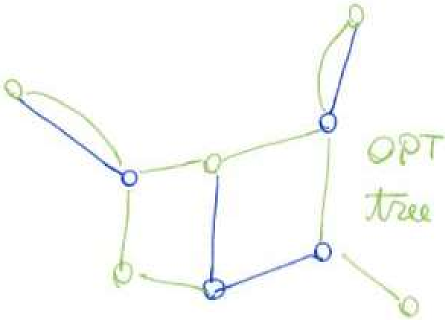
Take a set with the i "most expensive" vertices



Proving Auxiliary Result

Lemma 1: The i-th most expensive edge costs at most $\frac{2\mathrm{OPT}}{i}$

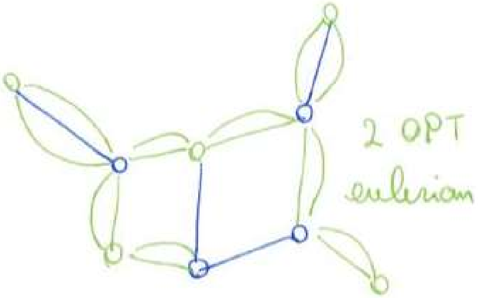
Consider the optimal tree which connects every terminal



Proving Auxiliary Result

Lemma 1: The i-th most expensive edge costs at most $\frac{2\mathrm{OPT}}{i}$

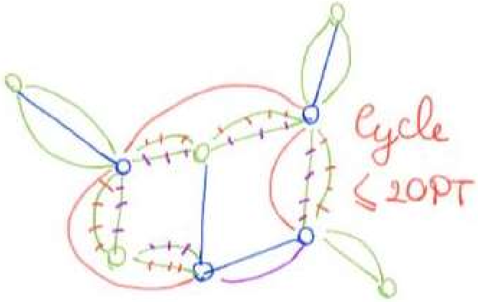
Two optimal trees form an eulerian graph



Proving Auxiliary Result

Lemma 1: The i-th most expensive edge costs at most $\frac{2\mathrm{OPT}}{i}$

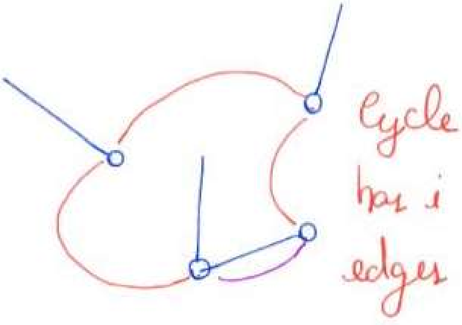
Since we are in the metric completion special case, 2 optimal trees pay for a cycle connecting these vertices



Proving Auxiliary Result

Lemma 1: The i-th most expensive edge costs at most $\frac{2\mathrm{OPT}}{i}$

Since the cycle has i vertices, it has i edges

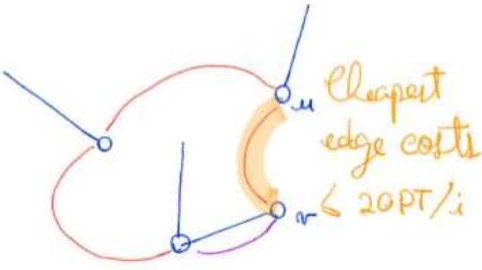


Proving Auxiliary Result

Lemma 1: The i-th most expensive edge costs at most $\frac{2\mathrm{OPT}}{i}$

Thus, the average edge cost in this cycle is at most $\frac{2\mathrm{OPT}}{i}$

The cost of the cheapest edge in the cycle is at most the average



Proving Auxiliary Result

Lemma 1: The i-th most expensive edge costs at most $\frac{2\mathrm{OPT}}{i}$

Now we show that, the cheapest edge was an option for the algorithm

Consider both endpoints u and v of the cheapest edge

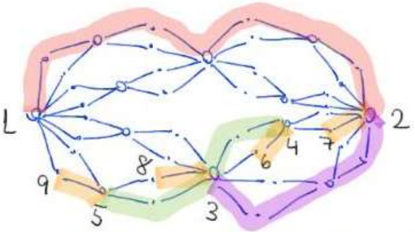
Suppose v arrived later, at a time u was already in the algorithm tree

Thus, the greedy algorithm bought and edge with cost at most $c(u, v)$ to connect v

This bought edge is in the set of the i most expensive edges

So, the i-th most expensive edge cost is at most $c(u, v) \leq \frac{2\mathrm{OPT}}{i}$

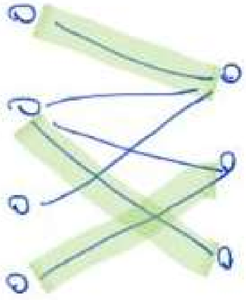
Lower Bound for Online Steiner Tree



$$ALG = 1 + \frac{1}{2} + 2 * \frac{1}{4} + 4 * \frac{1}{8} = \frac{5}{2}$$

$$OPT + \underbrace{\frac{OPT}{2} + \frac{OPT}{2} + \frac{OPT}{2}}_{\frac{OPT}{2} * \lg n}$$

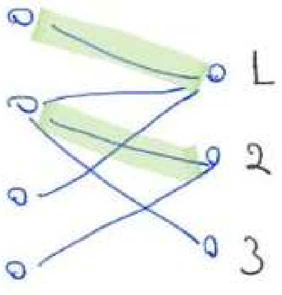
Bipartite Matching Problem



Online Bipartite Matching Problem

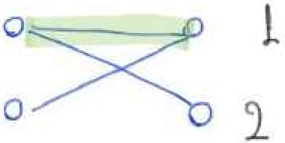
Greedy Online Bipartite Matching Algorithm

- Connects current r.h.s. node with arbitrary l.h.s. neighbor



Lower Bound for Online Bipartite Matching

Lets design a worst case example for any deterministic algorithm.



Any deterministic alg. is at most $\frac{1}{2}$-comp.

Analyzing Greedy Algorithm

Relying on Linear Programming

$$\max \sum_{e \in E} x_e$$

$$\text{s.t.} \quad \sum_{e \in \delta(v)} x_e \leq 1, \forall v \in V$$

$$x_e \geq 0, \forall e \in E$$

and Duality

$$\min \sum_{v \in V} y_v$$

$$\text{s.t. } \; y_u + y_v \geq 1, \forall e = (u, v) \in E$$

$$y_u \geq 0, \forall u \in V$$

For each chosen edge $e = (u, v)$, make $x_e = 1$ and $y_u = y_v = 1$.

Auxiliary Results

For each chosen edge $e = (u, v)$, make $x_e = 1$ and $y_u = y_v = 1$.

Idea is based on Primal-Dual relation, in which each constraint corresponds to a variable

Lemma 1: $\sum_{v \in V} y_v = 2 \sum_{e \in E} x_e$, since each edge has 2 vertices and no vertice has more than an edge in a matching

Lemma 2: The dual is feasible, i.e., $y_u + y_v \geq 1, \forall (u, v) \in E$

By contradiction, suppose there is an edge $e = (u, v)$ such that $y_u + y_v = 0$

Since both u and v are free, why the algorithm did not chose e when it arrived?

Greedy Algorithm is $\frac{1}{2}$-competitive

Lemma 1: $\sum_{v \in V} y_v = 2 \sum_{e \in E} x_e$

Lemma 2: The dual is feasible, i.e., $y_u + y_v \geq 1, \forall (u, v) \in E$

Back to the main result, since our primal is a maximization problem,

- by weak duality we have $\mathrm{OPT} \leq \sum_{u \in V} y_u$

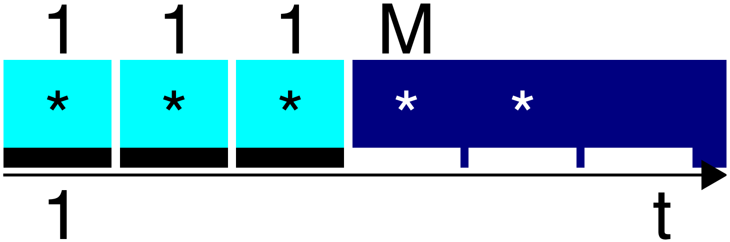
Thus,

$$
\begin{aligned}
|M| &= \sum_{e \in E} x_e \\
&= \frac{1}{2} \sum_{u \in V} y_v \\
&\geq \frac{1}{2} \mathrm{OPT}
\end{aligned}
$$

and the greedy algorithm is $\frac{1}{2}$-competitive

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)

Does a greedy algorithm solve this problem?

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]

Important both to theoretical and practical leasing problems