Online Combinatorial Optimization Problems

Mário César San Felice

Instituto de Matemática e Estatística - Universidade de São Paulo

felice@ic.unicamp.br

20 de maio de 2016
Combinatorial Optimization Problems

Maximization or minimization problems
Set of inputs and set of solutions
Cost associated with each pair (input, solution)
As an example, let's take the Steiner Tree Problem
Maximization or minimization problems
Set of inputs and set of solutions
Cost associated with each pair (input, solution)
Combinatorial Optimization Problems

Maximization or minimization problems
Set of inputs and set of solutions
Cost associated with each pair (input, solution)

As an example, let's take the Steiner Tree Problem
Steiner Tree Problem

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

Solution: tree $T$ connecting terminal nodes $D$

Cost: $\sum_{e \in T} d(e)$
Steiner Tree Problem

Minimization problem
Input: \( G = (V, E), d : E \rightarrow \mathbb{R}^+ \), \( D \subseteq V \)
Solution: tree \( T \) connecting terminal nodes \( D \)
Cost: \( \sum_{e \in T} d(e) \)
Steiner Tree Problem

Minimization problem
Input: \( G = (V, E), \ d : E \to \mathbb{R}^+, \ D \subseteq V \)
Solution: tree \( T \) connecting terminal nodes \( D \)
Cost: \( \sum_{e \in T} d(e) \)
Steiner Tree Problem

Minimization problem

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

Solution: tree \( T \) connecting terminal nodes \( D \)

Cost: \( \sum_{e \in T} d(e) \)
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, let's take the Online Steiner Tree problem
Input parts arrive one at a time
Each part is served before next one arrives
No decision can be changed in the future
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, let's take the Online Steiner Tree problem
Online Steiner Tree Problem

Similar to the Steiner Tree problem

Terminal nodes arrive one at a time

No edge used can be removed in the future
Online Steiner Tree Problem

Similar to the Steiner Tree problem
Terminal nodes arrive one at a time
No edge used can be removed in the future
Online Steiner Tree Problem

Similar to the Steiner Tree problem
Terminal nodes arrive one at a time
No edge used can be removed in the future
Online Steiner Tree Problem

Similar to the Steiner Tree problem
Terminal nodes arrive one at a time
No edge used can be removed in the future

![Graph diagram](image-url)
Online Steiner Tree Problem

Similar to the Steiner Tree problem
Terminal nodes arrive one at a time
No edge used can be removed in the future
Online Steiner Tree Problem

Similar to the Steiner Tree problem
Terminal nodes arrive one at a time
No edge used can be removed in the future
Online Steiner Tree Problem

Similar to the Steiner Tree problem
Terminal nodes arrive one at a time
No edge used can be removed in the future
Competitive Analysis

Worst case analysis technique for online algorithm ALG using offline optimal solution OPT.

ALG is $c$-competitive if $\text{ALG}(I) \leq c \cdot \text{OPT}(I)$ for every input $I$.

As an example, let's take a greedy online algorithm for the Online Steiner Tree problem.
Competitive Analysis

Worst case analysis technique
For online algorithm ALG
Using offline optimal solution OPT
Competitive Analysis

Worst case analysis technique
For online algorithm ALG
Using offline optimal solution OPT

ALG is $c$-competitive if

$$\text{ALG}(I) \leq c \text{OPT}(I)$$

for every input $I$
Competitive Analysis

Worst case analysis technique
For online algorithm ALG
Using offline optimal solution OPT

ALG is $c$-competitive if

$$ALG(I) \leq c \cdot OPT(I)$$

for every input $I$

As an example, let's take a greedy online algorithm for the Online Steiner Tree problem
Algorithm 1: OST Algorithm

Input: \((G, d)\)

\[ T \leftarrow (\emptyset, \emptyset); \]

while a new terminal \(j\) arrives do

\[ T \leftarrow T \cup \{ \text{path}(j, V(T)) \}; \]

end

return \(T\);

This algorithm is \(O(\log n)\)-competitive [Imase and Waxman 1991]

A \(\Omega(\log n)\) lower bound is known [Imase and Waxman 1991]
Algorithm 1: OST Algorithm

Input: \((G, d)\)

\[ T \leftarrow (\emptyset, \emptyset); \]

\begin{algorithmic}
\While {a new terminal \(j\) arrives}
\State \( T \leftarrow T \cup \{ \text{path}(j, V(T)) \}; \)
\EndWhile
\State return \( T; \)
\end{algorithmic}

This algorithm is \(O(\log n)\)-competitive [Imase and Waxman 1991]. An \(\Omega(\log n)\) lower bound is known [Imase and Waxman 1991].
**Algorithm 1: OST Algorithm**

**Input:** \((G, d)\)

\[ T \leftarrow (\emptyset, \emptyset); \]

**while a new terminal \(j\) arrives do**

\[ T \leftarrow T \cup \{\text{path}(j, V(T))\}; \]

end

**return** \(T\);

This algorithm is \(O(\log n)\)-competitive [Imase and Waxman 1991]
Greedy Online Steiner Tree Algorithm

Algorithm 1: OST Algorithm

Input: \((G, d)\)

\(T \leftarrow (\emptyset, \emptyset);\)

while a new terminal \(j\) arrives do

\(T \leftarrow T \cup \{\text{path}(j, V(T))\};\)

end

return \(T;\)

This algorithm is \(O(\log n)\)-competitive [Imase and Waxman 1991]

A \(\Omega(\log n)\) lower bound is known [Imase and Waxman 1991]
Areas of Interest

Online problems capture uncertainty, common in operations research and computer science:

- Resource management: scheduling, packing and load balancing problems
- Dynamic data structures: list access problem
- Memory management: paging problem
- Sustainability: ski-rental problem
- Network design: online versions of Steiner tree and facility location problems
Areas of Interest

Online problems capture uncertainty
Areas of Interest

Online problems capture uncertainty

Common in operations research and computer science:
Areas of Interest

Online problems capture uncertainty

Common in operations research and computer science:

- Resource management: scheduling, packing and load balancing problems
Areas of Interest

Online problems capture uncertainty

Common in operations research and computer science:

- Resource management: scheduling, packing and load balancing problems
- Dynamic data structures: list access problem
Areas of Interest

Online problems capture uncertainty

Common in operations research and computer science:

- Resource management: scheduling, packing and load balancing problems
- Dynamic data structures: list access problem
- Memory management: paging problem
Areas of Interest

Online problems capture uncertainty

Common in operations research and computer science:

- Resource management: scheduling, packing and load balancing problems
- Dynamic data structures: list access problem
- Memory management: paging problem
- Sustainability: ski-rental problem
Areas of Interest

Online problems capture uncertainty

Common in operations research and computer science:

- Resource management: scheduling, packing and load balancing problems
- Dynamic data structures: list access problem
- Memory management: paging problem
- Sustainability: ski-rental problem
- Network design: online versions of Steiner tree and facility location problems
Online Load Balancing problem

Minimization problem

Input: machines $M$, tasks $D$, sizes $s$:

$D \rightarrow R^+$

Solution: assignment of tasks to machines

Cost: max $M_i=1 l(i)$
Online Load Balancing problem

Minimization problem
Input: machines \( M \), tasks \( D \), sizes \( s : D \to \mathbb{R}^+ \)
Solution: assignment of tasks to machines
Cost: \( \max_{i=1}^{M} l(i) \)
Online Load Balancing problem

Minimization problem
Input: machines $M$, tasks $D$, sizes $s : D \to \mathbb{R}^+$
Solution: assignment of tasks to machines
Cost: $\max_{i=1}^M l(i)$
Online Load Balancing problem

Minimization problem
Input: machines $M$, tasks $D$, sizes $s : D \to \mathbb{R}^+$
Solution: assignment of tasks to machines
Cost: $\max_{i=1}^{M} l(i)$
Online Load Balancing problem

Minimization problem
Input: machines $M$, tasks $D$, sizes $s : D \rightarrow \mathbb{R}^+$
Solution: assignment of tasks to machines
Cost: $\max_{i=1}^{M} l(i)$
Online Load Balancing problem

Minimization problem
Input: machines $M$, tasks $D$, sizes $s : D \rightarrow \mathbb{R}^+$
Solution: assignment of tasks to machines
Cost: $\max_{i=1}^{M} l(i)$
Greedy Online Load Balancing Algorithm

Algorithm 2: OLB Algorithm

Input:

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 \( i^*; \)

end

return \( a; \)
Algorithm 2: 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 $i^*$;

end

return $a$;
OLB Algorithm is \((2 - \frac{1}{M})\)-competitive.

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

We have \(\text{OPT} \geq s(j)\) and \(\text{OPT} \geq l + s(j)\).
OLB Algorithm is \((2 - \frac{1}{M})\)-competitive

Let \(i^*\) be the machine with maximum load, \(j\) be the last task assigned to \(i^*\), and \(l(i^*) = l + s(j)\)
OLB Algorithm is \((2 - \frac{1}{M})\)-competitive

Let \(i^*\) be the machine with maximum load, \(j\) be the last task assigned to \(i^*\), and \(l(i^*) = l + s(j)\).
OLB Algorithm is \((2 - \frac{1}{M})\)-competitive

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

We have \(\text{OPT} \geq s(j)\) and \(\text{OPT} \geq l + \frac{s(j)}{M}\)
OLB Algorithm is \((2 - \frac{1}{M})\)-competitive
OLB Algorithm is \((2 - \frac{1}{M})\)-competitive

Since \(OPT \geq s(j)\) and \(OPT \geq l + \frac{s(j)}{M}\), we have
OLB Algorithm is \((2 - \frac{1}{M})\)-competitive

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

\[
\text{ALG} = l + s(j) \\
\leq \text{OPT} - \frac{s(j)}{M} + s(j) \\
\leq \text{OPT} + \left(1 - \frac{1}{M}\right) \text{OPT} \\
= \left(2 - \frac{1}{M}\right) \text{OPT}
\]
Lower Bound for OLB Algorithm

We have $\text{ALG} = 2^M - 1$ and $\text{OPT} = M$. 

Mário César San Felice (IME-USP)
Lower Bound for OLB Algorithm

List with $M(M - 1)$ size 1 tasks followed by one size $M$ task
Lower Bound for OLB Algorithm

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

![Diagram showing ALG and OPT algorithms with a lower bound comparison]
List with $M(M - 1)$ size 1 tasks followed by one size $M$ task.
List with $M(M - 1)$ size 1 tasks followed by one size $M$ task

We have $\text{ALG} = 2M - 1$ and $\text{OPT} = M$
Ski Rental Problem

Minimization problem

Input: skis price \( M \), list informing when snow melts

Solution: list informing when we rent or buy skis

Cost: 1 for each renting day plus \( M \) if we buy skis
Ski Rental Problem

Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis
Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis
Ski Rental Problem

Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis
Ski Rental Problem

Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis
Ski Rental Problem

Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis
Ski Rental Problem

Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis
Ski Rental Problem

Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis

1 1 1
* * *
1 1 1
Ski Rental Problem

Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis
Ski Rental Problem

Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis
Ski Rental Problem

Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis
Ski Rental Problem

Minimization problem
Input: skis price $M$, list informing when snow melts
Solution: list informing when we rent or buy skis
Cost: 1 for each renting day plus $M$ if we buy skis

\[ 1 \quad 1 \quad 1 \quad M \]
\[ 1 \quad 1 \quad 1 \quad 1 \quad \text{M} \quad \text{t} \]
Ski Rental Application and Generalization

Ski rental algorithms 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 to theoretical and practical leasing problems
Ski rental algorithms 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
Ski Rental Application and Generalization

Ski rental algorithms 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 to theoretical and practical leasing problems
Ski Rental Algorithm

Algorithm 3: 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 ← r + 1;
else
Buy skis if still don't have them;
end
j ← j + 1;
end

This algorithm is 2-competitive. Why?
Algorithm 3: 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 $r ← r + 1$;
  else
    Buy skis if still don’t have them;
  end
  $j ← j + 1$;
end

This algorithm is 2-competitive. Why?
Algorithm 3: 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;
  end

  $j \leftarrow j + 1$;

end

This algorithm is 2-competitive. Why?
Ski Rental LP Formulations

\[
\begin{align*}
\text{min} & \quad Mx + \sum_{j=1}^{n} y_j \\
\text{s.t.} & \quad x + y_j \geq 1 \text{ for } j = 1, \ldots, n \\
& \quad x \geq 0, \quad y_j \geq 0 \text{ for } j = 1, \ldots, n
\end{align*}
\]

and its dual

\[
\begin{align*}
\text{max} & \quad \sum_{j=1}^{n} \alpha_j \\
\text{s.t.} & \quad \sum_{j=1}^{n} \alpha_j \leq M \\
& \quad \alpha_j \leq 1 \text{ for } j = 1, \ldots, n \\
& \quad \alpha_j \geq 0 \text{ for } j = 1, \ldots, n
\end{align*}
\]
Ski Rental LP Formulations

Linear programming relaxation

\[
\begin{align*}
\text{min} & \quad Mx + \sum_{j=1}^{n} y_j \\
\text{s.t.} & \quad x + y_j \geq 1 \quad \text{for } j = 1, \ldots, n \\
& \quad x \geq 0, y_j \geq 0 \quad \text{for } j = 1, \ldots, n
\end{align*}
\]
Ski Rental LP Formulations

Linear programming relaxation

\[ \begin{align*}
\text{min} & \quad Mx + \sum_{j=1}^{n} y_j \\
\text{s.t.} & \quad x + y_j \geq 1 \quad \text{for } j = 1, \ldots, n \\
& \quad x \geq 0, y_j \geq 0 \quad \text{for } j = 1, \ldots, n
\end{align*} \]

and its dual

\[ \begin{align*}
\text{max} & \quad \sum_{j=1}^{n} \alpha_j \\
\text{s.t.} & \quad \sum_{j=1}^{n} \alpha_j \leq M \\
& \quad \alpha_j \leq 1 \quad \text{for } j = 1, \ldots, n \\
& \quad \alpha_j \geq 0 \quad \text{for } j = 1, \ldots, n
\end{align*} \]
Primal-Dual Ski Rental Algorithm

Algorithm 4: Primal-Dual SR Algorithm

Input:

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 $x_j = 1$ */

$j' \leftarrow j' + 1$;

end

Is it similar to the previous algorithm?
Algorithm 4: 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 $x = 1$ */

$j' \leftarrow j' + 1$;

end
Algorithm 4: 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 */
    $j' \leftarrow j' + 1$
end

Is it similar to the previous algorithm?
Primal-Dual SR Algorithm is 2-Competitive

Cost of any dual solution is at most $\text{OPT}$

So $\text{ALG} = Mx + n \sum_{j=1}^{n} y_j \leq n \sum_{j=1}^{n} \alpha_j + n \sum_{j=1}^{n} \alpha_j \leq 2 \text{OPT}$
Primal-Dual SR Algorithm is 2-Competitive

Cost of any dual solution is at most OPT
Primal-Dual SR Algorithm is 2-Competitive

Cost of any dual solution is at most $OPT$

So

$$ALG = Mx + \sum_{j=1}^{n} y_j$$

$$\leq \sum_{j=1}^{n} \alpha_j + \sum_{j=1}^{n} \alpha_j$$

$$\leq 2OPT$$
Online Facility Location Problem

\[ \text{Total cost} = 2 + 2 + 2 = 6. \]
Online Facility Location Problem

\[
\text{Total cost} = 2 + 2 + 2 = 6.
\]
Online Facility Location Problem

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

Total cost = 2 + 2 + 2 = 6.
Online Facility Location Problem

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

\[f = 2\]

Total cost = 2 + 2 + 2 = 6.
Online Facility Location Problem

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

Total cost = 2
min \sum_{i \in F^a} f(i) + \sum_{j \in D} d(j, a(j))

Total cost = 2
Online Facility Location Problem

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

Total cost = 2 + 2
Online Facility Location Problem

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

Total cost = 2 + 2
Online Facility Location Problem

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

Total cost = 2 + 2 + 2
Online Facility Location Problem

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

Total cost = 2 + 2 + 2 = 6.
Online Facility Location LP Formulation

\[
\begin{align*}
\text{min} & \quad \sum_{i \in F} f(i) y_i + \sum_{j \in D} \sum_{i \in F} d(j, i) x_{ji} \\
\text{s.t.} & \quad x_{ji} \leq y_i \quad \text{for } j \in D \text{ and } i \in F \\
& \quad \sum_{i \in F} x_{ji} \geq 1 \quad \text{for } j \in D \\
& \quad y_i \geq 0, \quad x_{ji} \geq 0 \quad \text{for } j \in D \text{ and } i \in F
\end{align*}
\]

and its dual

\[
\begin{align*}
\text{max} & \quad \sum_{j \in D} \alpha_j \\
\text{s.t.} & \quad \sum_{j \in D} (\alpha_j - d(j, i)) + \leq f(i) \quad \text{for } i \in F \\
& \quad \alpha_j \geq 0 \quad \text{for } j \in D
\end{align*}
\]
Online Facility Location LP Formulation

Linear programming relaxation

\[
\begin{align*}
\text{min} & \quad \sum_{i \in F} f(i)y_i + \sum_{j \in D} \sum_{i \in F} d(j, i)x_{ji} \\
\text{s.t.} & \quad x_{ji} \leq y_i \quad \text{for } j \in D \text{ and } i \in F \\
& \quad \sum_{i \in F} x_{ji} \geq 1 \quad \text{for } j \in D \\
& \quad y_i \geq 0, x_{ji} \geq 0 \quad \text{for } j \in D \text{ and } i \in F
\end{align*}
\]
Online Facility Location LP Formulation

Linear programming relaxation

\[ \min \sum_{i \in F} f(i)y_i + \sum_{j \in D} \sum_{i \in F} d(j, i)x_{ji} \]

s.t.
\[ x_{ji} \leq y_i \quad \text{for } j \in D \text{ and } i \in F \]
\[ \sum_{i \in F} x_{ji} \geq 1 \quad \text{for } j \in D \]
\[ y_i \geq 0, x_{ji} \geq 0 \quad \text{for } j \in D \text{ and } i \in F \]

and its dual

\[ \max \sum_{j \in D} \alpha_j \]

s.t.
\[ \sum_{j \in D} (\alpha_j - d(j, i))^+ \leq f(i) \quad \text{for } i \in F \]
\[ \alpha_j \geq 0 \quad \text{for } j \in D \]
Online Facility Location Algorithm

Algorithm 5: OFL Algorithm

Input: $(G, d, f, F)$

\[ \begin{align*}
& F_a \leftarrow \emptyset; \\
& D \leftarrow \emptyset; \\
\end{align*} \]

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 */

\[ F_a \leftarrow F_a \cup \{i\}; \\
D \leftarrow D \cup \{j'\}; \\
a(j') \leftarrow i; \]

end

return $(F_a, a)$;
Algorithm 5: 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 */

    $F^a \leftarrow F^a \cup \{i\}$; $D \leftarrow D \cup \{j'\}$; $a(j') \leftarrow i$;
end

return $(F^a, a)$;
Online Facility Location Results

The OFL has competitive ratio $\Theta\left(\log n \log \log n\right)$ [Fotakis 2008]. There are randomized and deterministic $O(\log n)$-competitive algorithms known for it [Meyerson 2001, Fotakis 2007]. Nagarajan and Williamson 2013 give a dual-fitting analysis for the algorithm by [Fotakis 2007].
The OFL has competitive ratio $\Theta\left(\frac{\log n}{\log \log n}\right)$ [Fotakis 2008]
The OFL has competitive ratio $\Theta\left(\frac{\log n}{\log \log n}\right)$ [Fotakis 2008].

There are randomized and deterministic $O(\log n)$-competitive algorithms known for it [Meyerson 2001, Fotakis 2007].
The OFL has competitive ratio $\Theta \left( \frac{\log n}{\log \log n} \right)$ [Fotakis 2008]

There are randomized and deterministic $O(\log n)$-competitive algorithms known for it [Meyerson 2001, Fotakis 2007]

[Nagarajan and Williamson 2013] give a dual-fitting analysis for the algorithm by [Fotakis 2007]
Acknowledgements

Thank you!

Questions?