R–LINE: An Online Algorithm for the 2-Server Problem on the Line with Improved Competitive Ratio

Lucas Bang

University of Nevada, Las Vegas

bang@unlv.nevada.edu

8 April 2013

Thesis Defense

Abstract

Thesis Result

Abstract

Thesis Result

- A randomized online algorithm for the 2-server problem on the line.

Abstract

Thesis Result

- A randomized online algorithm for the 2-server problem on the line.
- Competitiveness ≤ 1.901 against the oblivious adversary.

Abstract

Thesis Result

- A randomized online algorithm for the 2-server problem on the line.
- Competitiveness ≤ 1.901 against the oblivious adversary.
- Improves the previously best known competitiveness of $\frac{155}{78} \approx 1.987$.

Outline

1. Offline vs. online algorithms.
2. Competitive analysis and game theory.
3. The *k*-server problem.
4. Our 2-server algorithm, R–LINE (Randomized Line).
5. Future work.

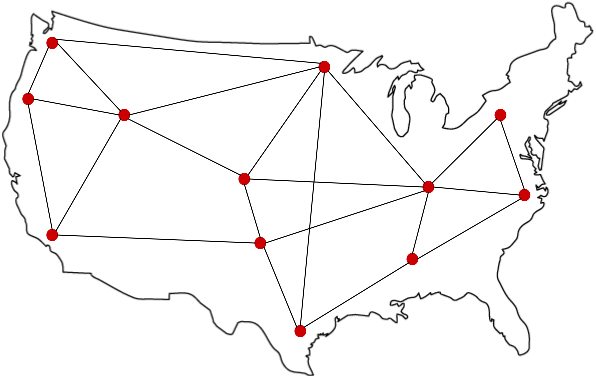
Offline Algorithms

Typically, one initially studies algorithms in the **offline** setting, where all data is available to the algorithm at start-up.

Offline Algorithms

Typically, one initially studies algorithms in the **offline** setting, where all data is available to the algorithm at start-up.

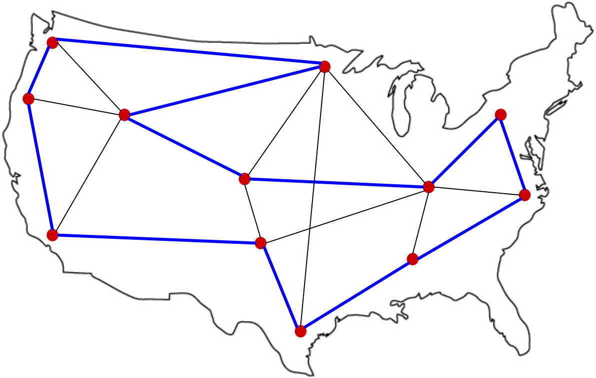
Example: Traveling Salesman Problem.



Offline Algorithms

Typically, one initially studies algorithms in the **offline** setting, where all data is available to the algorithm at start-up.

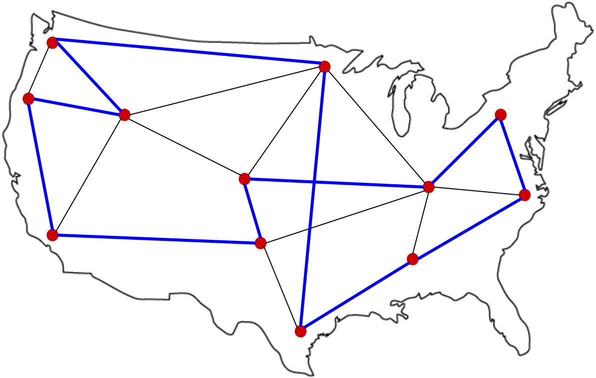
Example: Traveling Salesman Problem.



Offline Algorithms

Typically, one initially studies algorithms in the **offline** setting, where all data is available to the algorithm at start-up.

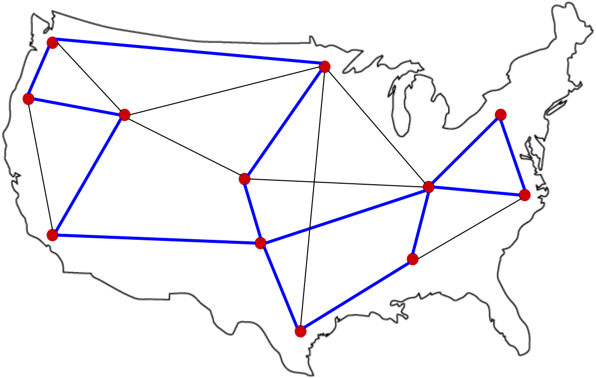
Example: Traveling Salesman Problem.



Offline Algorithms

Typically, one initially studies algorithms in the **offline** setting, where all data is available to the algorithm at start-up.

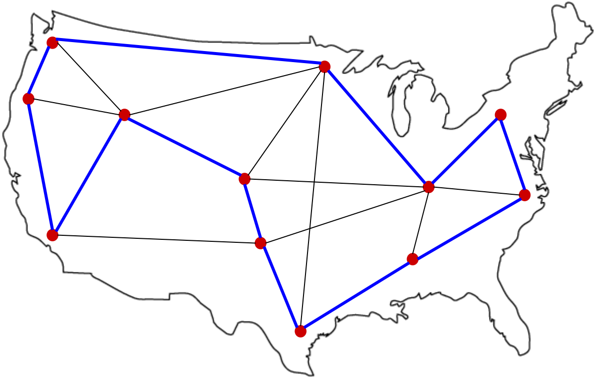
Example: Traveling Salesman Problem.



Offline Algorithms

Typically, one initially studies algorithms in the **offline** setting, where all data is available to the algorithm at start-up.

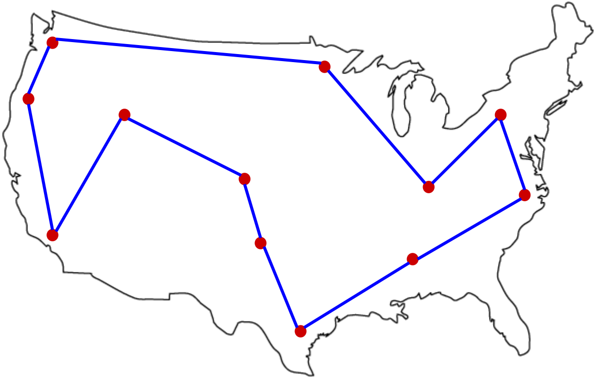
Example: Traveling Salesman Problem.



Offline Algorithms

Typically, one initially studies algorithms in the **offline** setting, where all data is available to the algorithm at start-up.

Example: Traveling Salesman Problem.



Online Algorithms

An **online** algorithm must make decisions with only partial information.

Online Algorithms

An **online** algorithm must make decisions with only partial information.

Example: Canadian Traveler's Problem

Online Algorithms

An **online** algorithm must make decisions with only partial information.

Example: Canadian Traveler's Problem

Online Algorithms

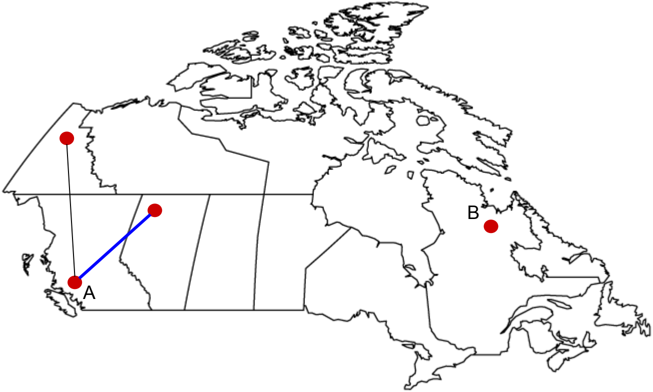
An **online** algorithm must make decisions with only partial information.

Example: Canadian Traveler's Problem

Online Algorithms

An **online** algorithm must make decisions with only partial information.

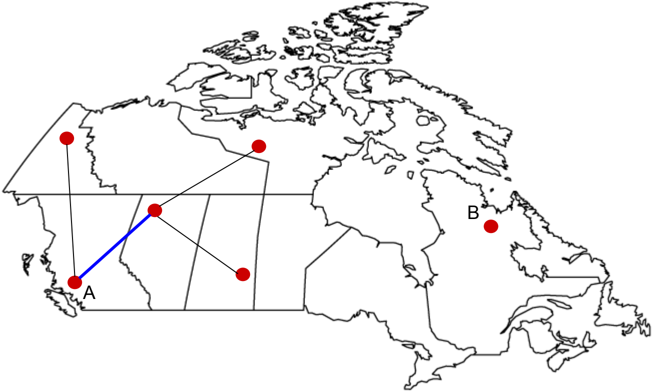
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

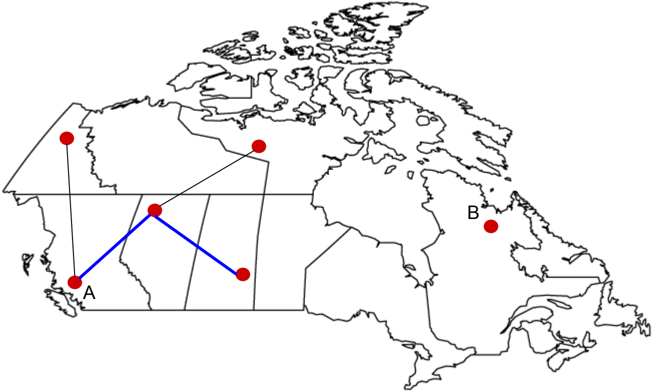
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

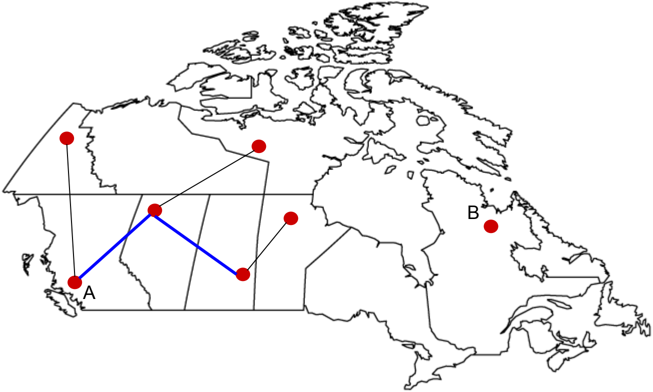
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

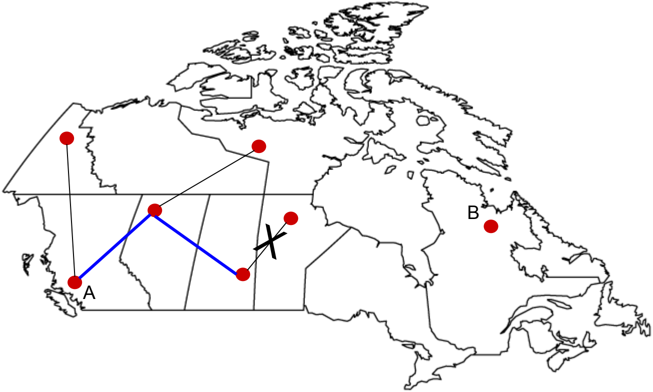
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

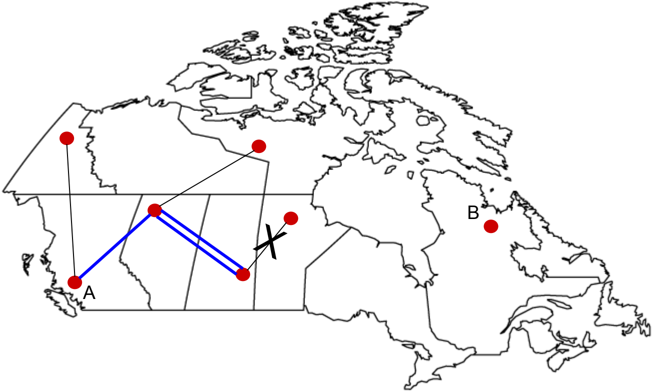
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

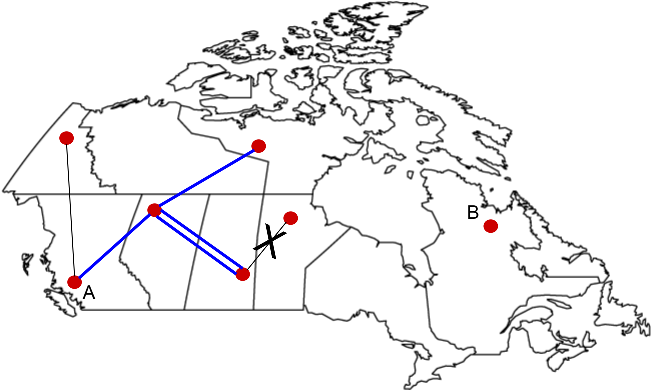
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

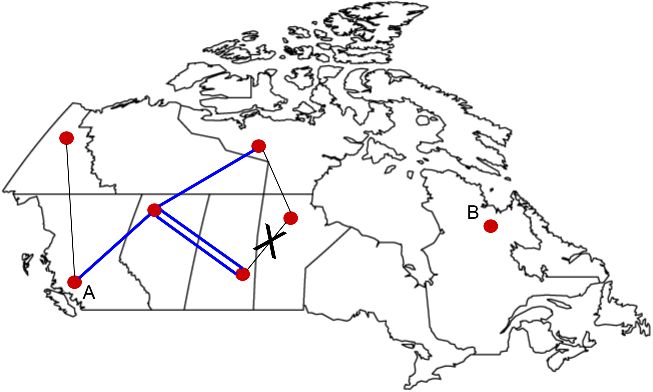
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

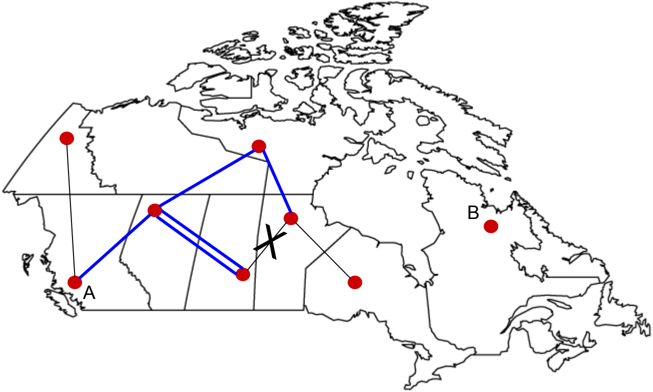
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

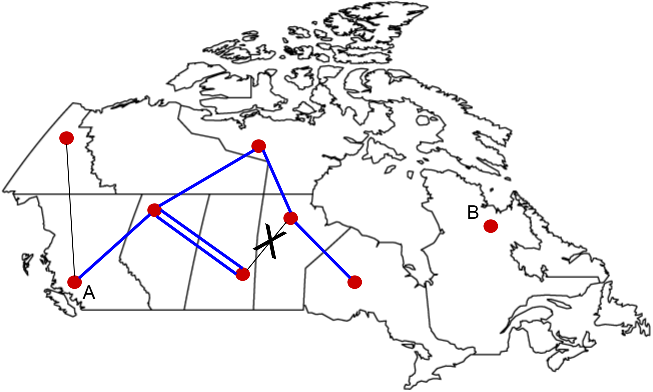
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

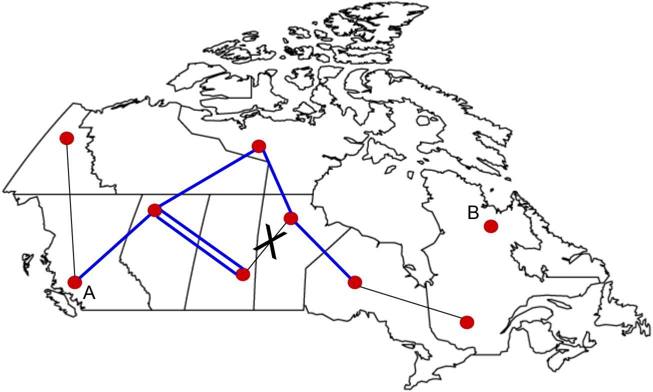
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

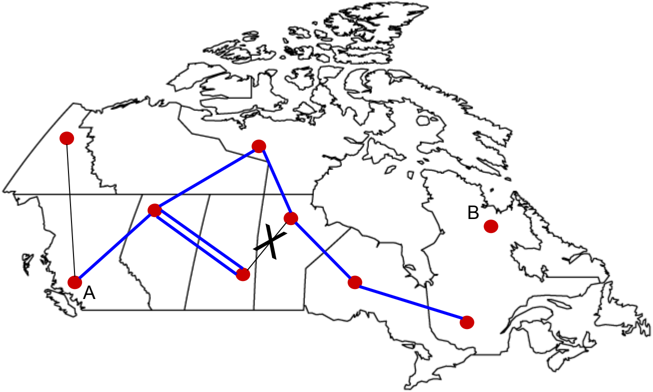
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

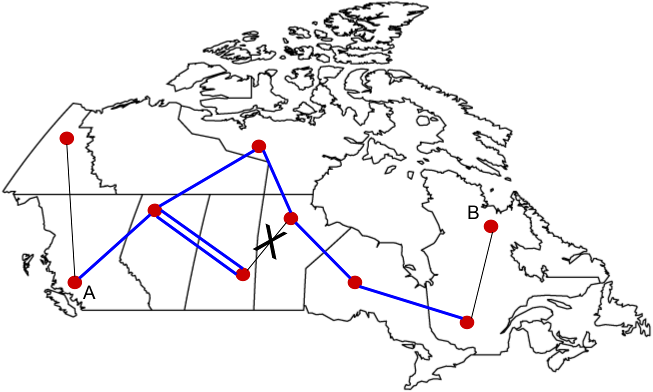
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

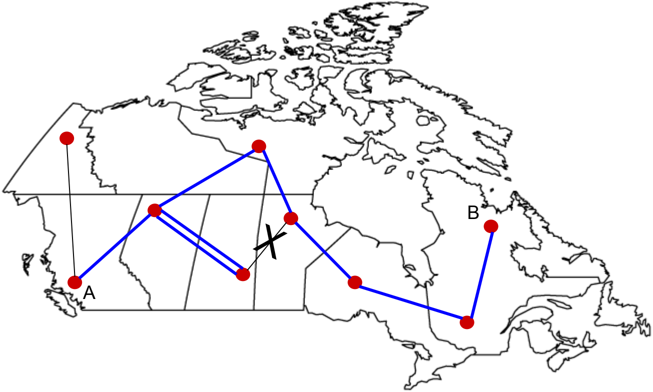
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

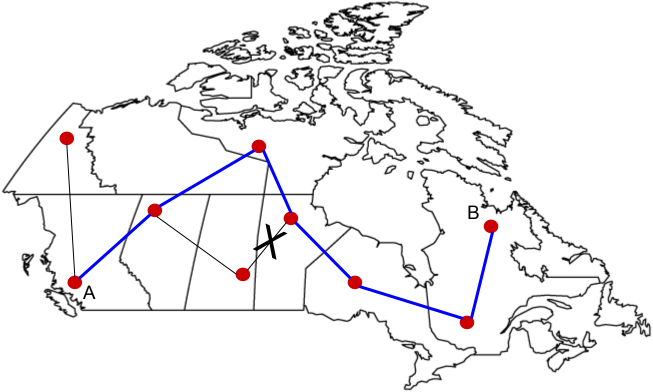
Example: Canadian Traveler's Problem



Online Algorithms

An **online** algorithm must make decisions with only partial information.

Example: Canadian Traveler's Problem



Online Algorithms

Many important real-world problems are online.

Examples

1. Investment decisions, as in algorithmic stock trading.
2. Job scheduling, as in multi-core computing.
3. Memory cache page management.
4. and more....

Online Algorithms

Online algorithms accept input one piece at a time and must produce an output before more information is given.

Online Algorithms

Online algorithms accept input one piece at a time and must produce an output before more information is given.

1. Input $I = I_1, I_2, \ldots, I_n$

Online Algorithms

Online algorithms accept input one piece at a time and must produce an output before more information is given.

1. Input $I = I_1, I_2, \ldots, I_n$
2. Receive an input I_i, and produces an output O_i.

Online Algorithms

Online algorithms accept input one piece at a time and must produce an output before more information is given.

1. Input $I = I_1, I_2, \ldots, I_n$
2. Receive an input I_i, and produces an output O_i.
3. Each input has an associated cost(I_i).

Online Algorithms

Online algorithms accept input one piece at a time and must produce an output before more information is given.

1. Input $I = I_1, I_2, \ldots, I_n$
2. Receive an input I_i, and produces an output O_i.
3. Each input has an associated $\text{cost}(I_i)$.
4. We wish to minimize the total $\text{cost}(I) = \sum \text{cost}(I_i)$.

Competitive Analysis

We measure the performance of an online algorithm A with the

Competitive Ratio

Competitive Analysis

We measure the performance of an online algorithm A with the

Competitive Ratio

- For any request sequence I the competitive ratio C satisfies

Competitive Analysis

We measure the performance of an online algorithm A with the

Competitive Ratio

- For any request sequence I the competitive ratio C satisfies

$$cost_A(I) \leq C \cdot cost_{\mathcal{OPT}}(I) + K$$

Competitive Analysis

We measure the performance of an online algorithm A with the Competitive Ratio

- For any request sequence I the competitive ratio C satisfies

$$cost_A(I) \leq C \cdot cost_{\mathcal{OPT}}(I) + K$$

- If A is randomized then

$$E[cost_A(I)] \leq C \cdot cost_{\mathcal{OPT}}(I) + K$$

Background

The k-server Problem

- Introduced by Manasse, McGeoch, and Sleator, 1990.

Background

The k-server Problem

- Introduced by Manasse, McGeoch, and Sleator, 1990.
- Given k mobile servers in a metric space M.

Background

The k-server Problem

- Introduced by Manasse, McGeoch, and Sleator, 1990.
- Given k mobile servers in a metric space M.
- Serve requests online in the metric space.

Background

The k-server Problem

- Introduced by Manasse, McGeoch, and Sleator, 1990.
- Given k mobile servers in a metric space M.
- Serve requests online in the metric space.
- Move a single server to the request point.

Background

The k-server Problem

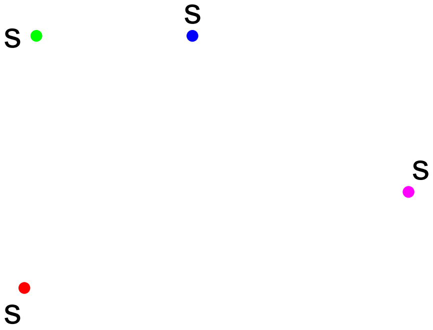
- Introduced by Manasse, McGeoch, and Sleator, 1990.
- Given k mobile servers in a metric space M.
- Serve requests online in the metric space.
- Move a single server to the request point.
- Goal: minimize total distance moved.

Background

4-server Problem

Background

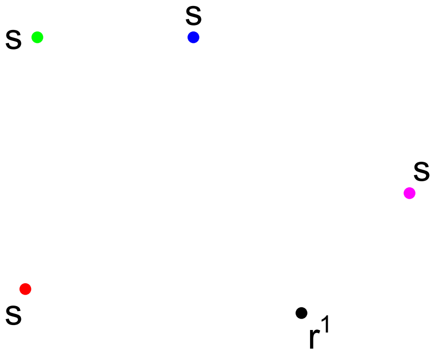
4-server Problem



$cost = \ldots$

Background

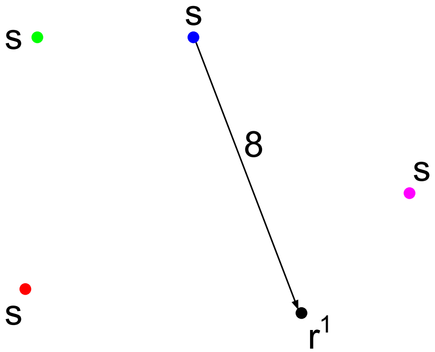
4-server Problem



$cost = \ldots$

Background

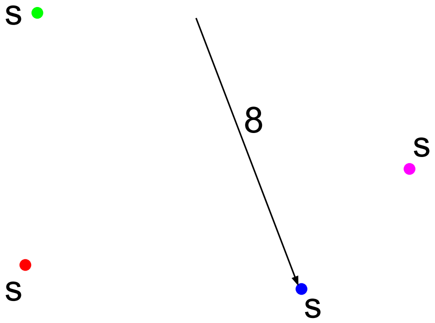
4-server Problem



$cost = 8 + \ldots$

Background

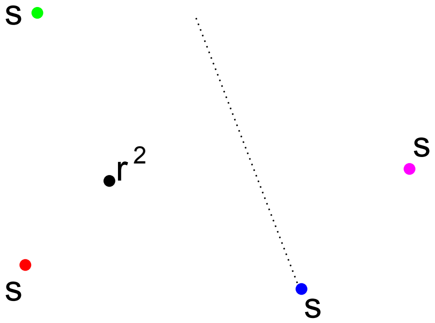
4-server Problem



$cost = 8 + \ldots$

Background

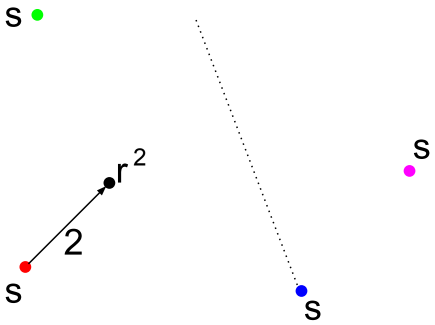
4-server Problem



$cost = 8 + \ldots$

Background

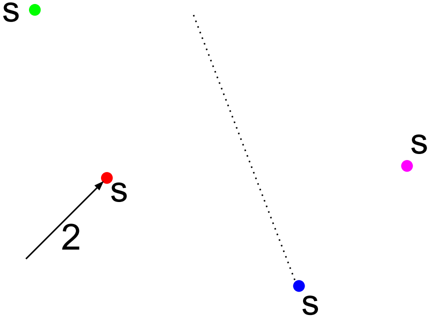
4-server Problem



$cost = 8 + 2 + \ldots$

Background

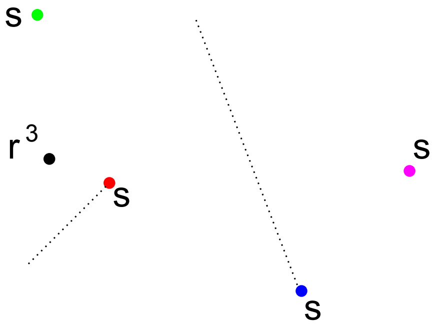
4-server Problem



$cost = 8 + 2 + \ldots$

Background

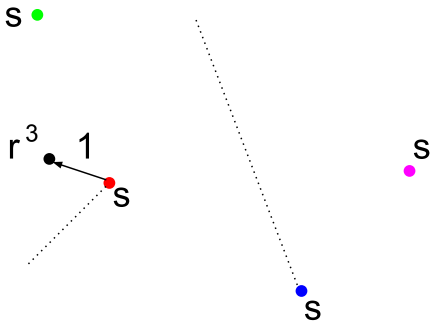
4-server Problem



$cost = 8 + 2 + \ldots$

Background

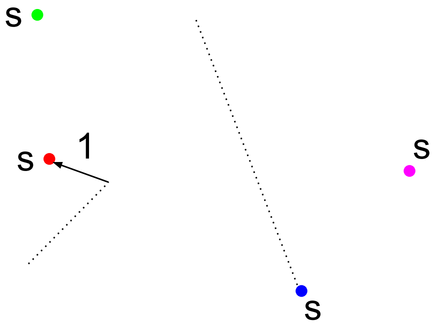
4-server Problem



$cost = 8 + 2 + 1 + \ldots$

Background

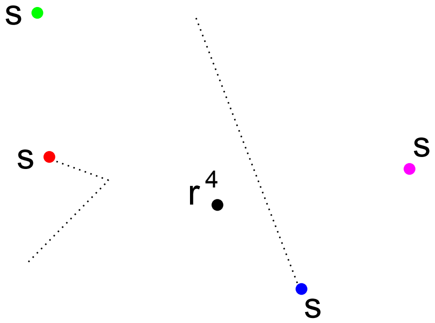
4-server Problem



$cost = 8 + 2 + 1 + \dots$

Background

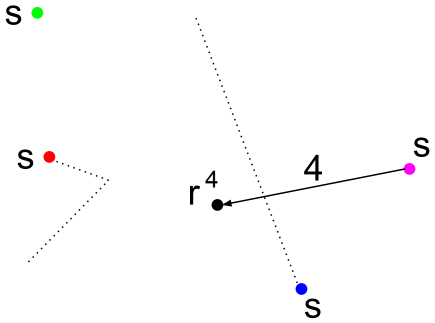
4-server Problem



$cost = 8 + 2 + 1 + \dots$

Background

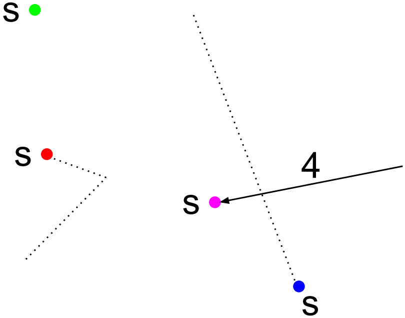
4-server Problem



$cost = 8 + 2 + 1 + 4 + \dots$

Background

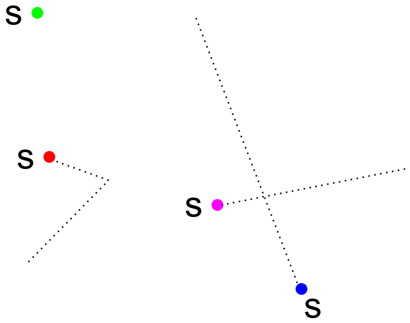
4-server Problem



$cost = 8 + 2 + 1 + 4 + \ldots$

Background

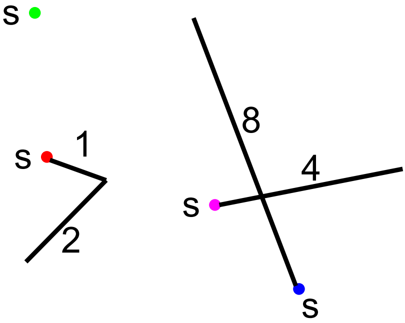
4-server Problem



$cost = 8 + 2 + 1 + 4$

Background

4-server Problem



$cost = 8 + 2 + 1 + 4 = 15$

Background

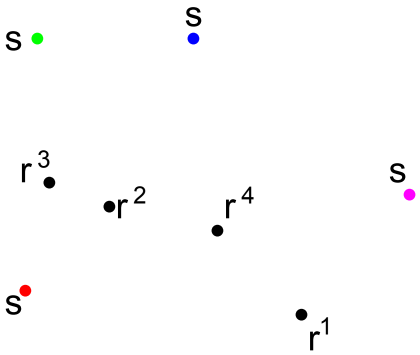
4-server Problem

Background

4-server Problem

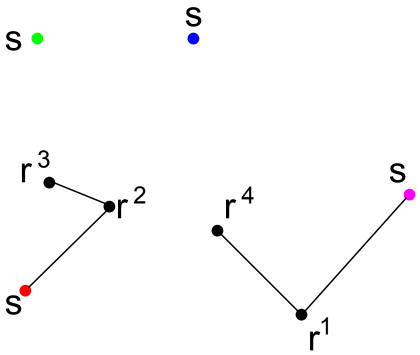
Background

4-server Problem



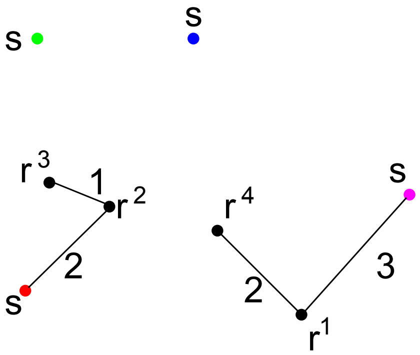
Background

4-server Problem



Background

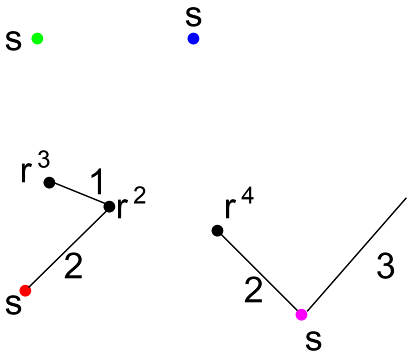
4-server Problem



optimal cost $= 3 + 2 + 1 + 2 = 8$

Background

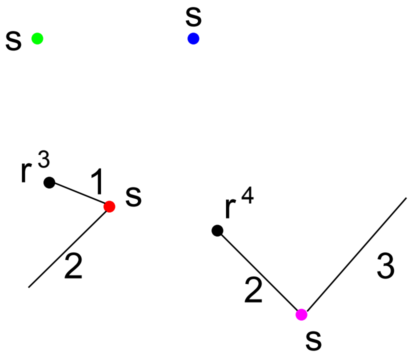
4-server Problem



optimal cost $= 3 + 2 + 1 + 2 = 8$

Background

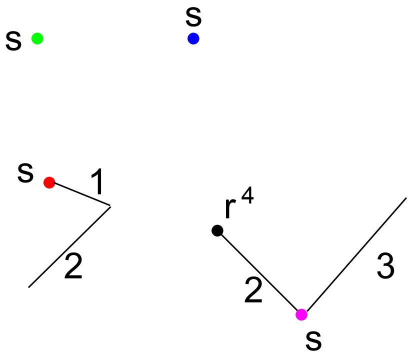
4-server Problem



$optimal\ cost = 3 + 2 + 1 + 2 = 8$

Background

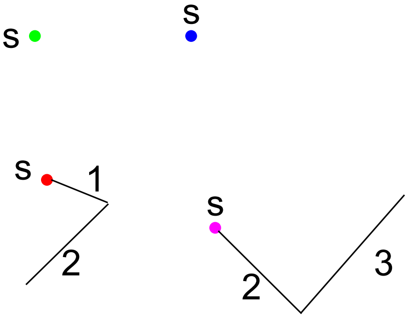
4-server Problem



$optimal\ cost = 3 + 2 + 1 + 2 = 8$

Background

4-server Problem



$optimal\ cost = 3 + 2 + 1 + 2 = 8$

Cost ratio for this particular input sequence:

- $cost = 15$

Cost ratio for this particular input sequence:

- $cost = 15$
- $optimal\ cost = 8$

Cost ratio for this particular input sequence:

- $cost = 15$
- $optimal\ cost = 8$
- $cost \leq 2 \cdot (optimal\ cost) + 1$

Cost ratio for this particular input sequence:

- $cost = 15$
- $optimal\ cost = 8$
- $cost \leq 2 \cdot (optimal\ cost) + 1$
- $15 \leq 2 \cdot 8 + 1$

Cost ratio for this particular input sequence:

- $cost = 15$
- $optimal\ cost = 8$
- $cost \leq 2 \cdot (optimal\ cost) + 1$
- $15 \leq 2 \cdot 8 + 1$

Compare with definition of competitive ratio:

Cost ratio for this particular input sequence:

- $cost = 15$
- $optimal\ cost = 8$
- $cost \leq 2 \cdot (optimal\ cost) + 1$
- $15 \leq 2 \cdot 8 + 1$

Compare with definition of competitive ratio:

$$cost_A(I) \leq C \cdot cost_{\mathcal{OPT}}(I) + K$$

Cost ratio for this particular input sequence:

- $cost = 15$
- $optimal\ cost = 8$
- $cost \leq 2 \cdot (optimal\ cost) + 1$
- $15 \leq 2 \cdot 8 + 1$

Compare with definition of competitive ratio:

$$cost_A(I) \leq C \cdot cost_{\mathcal{OPT}}(I) + K$$

$$cost \leq 2 \cdot (optimal\ cost) + 1$$

Cost ratio for this particular input sequence:

- $cost = 15$
- $optimal\ cost = 8$
- $cost \leq 2 \cdot (optimal\ cost) + 1$
- $15 \leq 2 \cdot 8 + 1$

Compare with definition of competitive ratio:

$$cost_A(I) \leq C \cdot cost_{\mathcal{OPT}}(I) + K$$

$$cost \leq 2 \cdot (optimal\ cost) + 1$$

If this holds for **all possible** input sequences, we could claim that A is 2-competitive.

The Potential Method

A common method for proving the competitiveness of an online algorithm. For the server problem, define a potential function

$$\phi : \textit{Server Locations} \rightarrow \mathbb{R}.$$

The Potential Method

A common method for proving the competitiveness of an online algorithm. For the server problem, define a potential function

$$\phi : \textit{Server Locations} \rightarrow \mathbb{R}.$$

Show that at any step i,

$$cost_A(r_i) \leq C \cdot cost_{OPT}(r_i) - (\phi_i - \phi_{i-1})$$

The Potential Method

A common method for proving the competitiveness of an online algorithm. For the server problem, define a potential function

$$\phi : \textit{Server Locations} \rightarrow \mathbb{R}.$$

Show that at any step i,

$$cost_A(r_i) \leq C \cdot cost_{OPT}(r_i) - (\phi_i - \phi_{i-1})$$

Then, summing over the request sequence:

$$\sum_{i=1}^{n} cost_A(r_i) \leq \sum_{i=1}^{n} C \cdot cost_{OPT}(r_i) - \sum_{i=1}^{n} \Delta\phi_i.$$

The Potential Method

A common method for proving the competitiveness of an online algorithm. For the server problem, define a potential function

$$\phi : \textit{Server Locations} \rightarrow \mathbb{R}.$$

Show that at any step i,

$$cost_A(r_i) \leq C \cdot cost_{OPT}(r_i) - (\phi_i - \phi_{i-1})$$

Then, summing over the request sequence:

$$\sum_{i=1}^{n} cost_A(r_i) \leq \sum_{i=1}^{n} C \cdot cost_{OPT}(r_i) - \sum_{i=1}^{n} \Delta\phi_i.$$

$$cost_A(R) \leq C \cdot cost_{OPT}(R) + K.$$

The Optimal Adversary Algorithm

How to model the optimal algorithm?

- We think of the optimal algorithm as a malevolent adversary.
- Adversary generates the input sequence $I_1, I_2, \ldots, I_n$
- Adversary must also use its servers to satisfy requests.
- Adversary tries to maximize C by simultaneously making its cost low and our cost high.

Game Theory

We can now think of the server problem as a game between our algorithm and the adversary algorithm. Consider the payoff matrix for a two-person zero-sum game G.

	Adversary Strategy 1	Adversary Strategy 2
Server Strategy 1	a_{11}	a_{12}
Server Strategy 2	a_{21}	a_{22}

Game Theory

We can now think of the server problem as a game between our algorithm and the adversary algorithm. Consider the payoff matrix for a two-person zero-sum game G.

	Adversary Strategy 1	Adversary Strategy 2
Server Strategy 1	a_{11}	a_{12}
Server Strategy 2	a_{21}	a_{22}

$$v(G) = \frac{\det A}{a_{11} - a_{12} - a_{21} + a_{22}}$$

Game Theory

	Adversary Strategy 1	Adversary Strategy 2
Server Strategy 1	a_{11}	a_{12}
Server Strategy 2	a_{21}	a_{22}

Optimum **row player** strategy:

Play row 1 with $p_1 = \frac{a_{22} - a_{21}}{a_{11} - a_{12} - a_{21} + a_{22}}$

Play row 2 with $p_2 = \frac{a_{11} - a_{12}}{a_{11} - a_{12} - a_{21} + a_{22}}$

Optimum **column player** strategy:

Play column 1 with $p_1 = \frac{a_{22} - a_{12}}{a_{11} - a_{12} - a_{21} + a_{22}}$

Play column 2 with $p_2 = \frac{a_{11} - a_{21}}{a_{11} - a_{12} - a_{21} + a_{22}}$

Generalization of the k-server problem

The (m, n)-server Problem

Generalization of the k-server problem

The (m, n)-server Problem

- Given m mobile servers in a metric space M.

Generalization of the k-server problem

The (m, n)-server Problem

- Given m mobile servers in a metric space M.
- Serve requests online in the metric space.

Generalization of the k-server problem

The (m, n)-server Problem

- Given m mobile servers in a metric space M.
- Serve requests online in the metric space.
- Each request requires n servers to move to the request point

Generalization of the k-server problem

The (m, n)-server Problem

- Given m mobile servers in a metric space M.
- Serve requests online in the metric space.
- Each request requires n servers to move to the request point
- Goal: minimize total distance moved.

Background

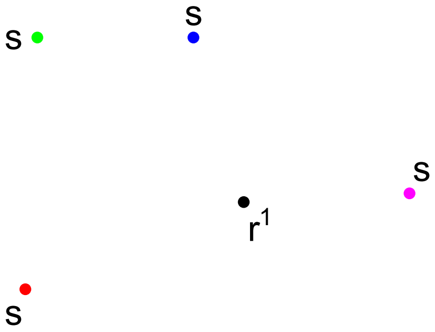
$(4, 2)$-server Problem

Background

$(4, 2)$-server Problem

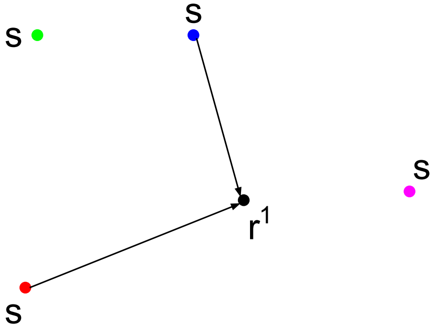
Background

$(4,2)$-server Problem



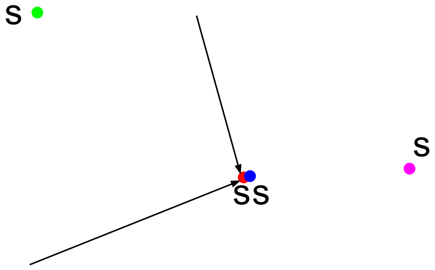
Background

(4, 2)-server Problem



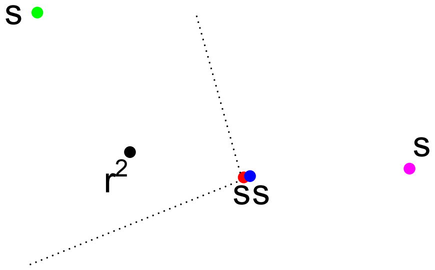
Background

(4, 2)-server Problem



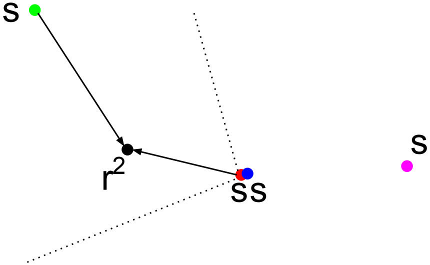
Background

(4, 2)-server Problem



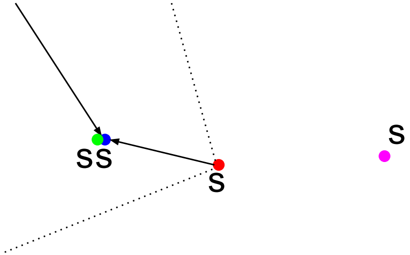
Background

$(4, 2)$-server Problem



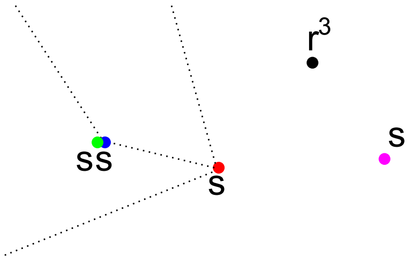
Background

$(4, 2)$-server Problem



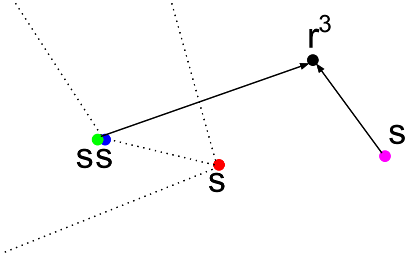
Background

$(4,2)$-server Problem



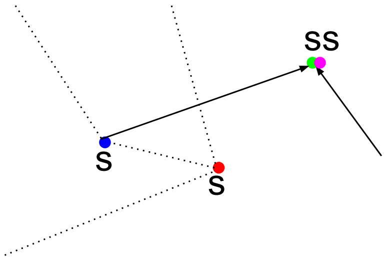
Background

$(4, 2)$-server Problem



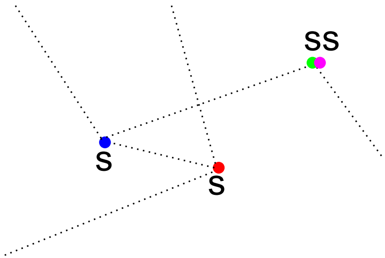
Background

$(4, 2)$-server Problem



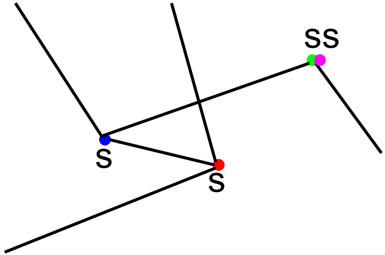
Background

$(4, 2)$-server Problem



Background

(4, 2)-server Problem



Useful Results

Useful Results

Theorem 1

$$C\text{-competitive } (2n, n)\text{-server Algorithm}$$
$$\Downarrow$$
$$C\text{-competitive 2-server Algorithm}$$

Useful Results

Theorem 1

$$\begin{gathered}C\text{-competitive } (2n, n)\text{-server Algorithm} \\ \Downarrow \\ C\text{-competitive 2-server Algorithm}\end{gathered}$$

Theorem 2

Optimal offline strategy for the $(2n, n)$ server problem keeps the servers in two blocks of size n.

R–LINE

We give an online algorithm for the 2-server problem where the metric space is the real line.

R–LINE

We give an online algorithm for the 2-server problem where the metric space is the real line.

Outline

- Define a randomized online algorithm for the $(2n, n)$-server problem.

R–LINE

We give an online algorithm for the 2-server problem where the metric space is the real line.

Outline

- Define a randomized online algorithm for the $(2n, n)$-server problem.
- Use 2-person zero-sum game theory for randomized moves.

R–LINE

We give an online algorithm for the 2-server problem where the metric space is the real line.

Outline

- Define a randomized online algorithm for the $(2n, n)$-server problem.
- Use 2-person zero-sum game theory for randomized moves.
- Prove competitiveness by solving non-linear constrained optimization problem and a suitable potential.

R–LINE

We give an online algorithm for the 2-server problem where the metric space is the real line.

Outline

- Define a randomized online algorithm for the $(2n, n)$-server problem.
- Use 2-person zero-sum game theory for randomized moves.
- Prove competitiveness by solving non-linear constrained optimization problem and a suitable potential.
- Derive randomized 2-server algorithm from $(2n, n)$-server algorithm, via Theorem 1.

R–LINE

We give an online algorithm for the 2-server problem where the metric space is the real line.

Outline

- Define a randomized online algorithm for the $(2n, n)$-server problem.
- Use 2-person zero-sum game theory for randomized moves.
- Prove competitiveness by solving non-linear constrained optimization problem and a suitable potential.
- Derive randomized 2-server algorithm from $(2n, n)$-server algorithm, via Theorem 1.
- As n grows large, competitiveness decreases.

R–LINE

We give an online algorithm for the 2-server problem where the metric space is the real line.

Outline

- Define a randomized online algorithm for the $(2n, n)$-server problem.
- Use 2-person zero-sum game theory for randomized moves.
- Prove competitiveness by solving non-linear constrained optimization problem and a suitable potential.
- Derive randomized 2-server algorithm from $(2n, n)$-server algorithm, via Theorem 1.
- As n grows large, competitiveness decreases.
- For R–LINE, $C \leq 1.901$

R–LINE Details

T-Theory on the Line

R–LINE Details

T-Theory on the Line

- For R–LINE, we have our algorithms servers, $s_1, s_2, ..., s_{2n}$, and

R–LINE Details

T-Theory on the Line

- For R–LINE, we have our algorithms servers, $s_1, s_2, ..., s_{2n}$, and
- Two adversary servers, a_1 and a_2 for a total of $2n+2$ points.

R–LINE Details

T-Theory on the Line

- For R–LINE, we have our algorithms servers, $s_1, s_2, ..., s_{2n}$, and
- Two adversary servers, a_1 and a_2 for a total of $2n + 2$ points.
- Define $\alpha_{i,j}$, the $(i,j)^{th}$ *isolation index* of a configuration, to be the length of the longest interval that has exactly i algorithm servers to the left and exactly j adversary servers to the left.

R–LINE Details

T-Theory on the Line

- For R–LINE, we have our algorithms servers, $s_1, s_2, ..., s_{2n}$, and
- Two adversary servers, a_1 and a_2 for a total of $2n + 2$ points.
- Define $\alpha_{i,j}$, the $(i,j)^{th}$ *isolation index* of a configuration, to be the length of the longest interval that has exactly i algorithm servers to the left and exactly j adversary servers to the left.
- Formally,

$$\alpha_{i,j} = \max\{0, \min\{s_{i+1}, a_{j+1}\} - \max\{s_i, a_j\}\}$$

R–LINE Details

T-Theory on the Line

- For R–LINE, we have our algorithms servers, $s_1, s_2, ..., s_{2n}$, and
- Two adversary servers, a_1 and a_2 for a total of $2n + 2$ points.
- Define $\alpha_{i,j}$, the $(i,j)^{th}$ *isolation index* of a configuration, to be the length of the longest interval that has exactly i algorithm servers to the left and exactly j adversary servers to the left.
- Formally,

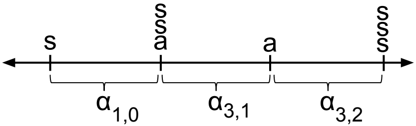
$$\alpha_{i,j} = \max\{0, \min\{s_{i+1}, a_{j+1}\} - \max\{s_i, a_j\}\}$$

- $s_0 = a_0 = -\infty$ and $s_{2n+1} = a_3 = \infty$

R–LINE Details

T-Theory on the Line

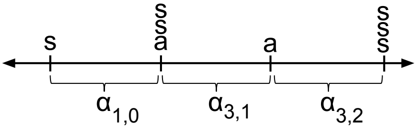
- $\alpha_{i,j}$ is the length of the longest interval that has exactly i algorithm servers to the left and exactly j adversary servers to the left.
- $\alpha_{i,j} = \max\{0, \min\{s_{i+1}, a_{j+1}\} - \max\{s_i, a_j\}\}$
- Example,



R–LINE Details

T-Theory on the Line

- $\alpha_{i,j}$ is the length of the longest interval that has exactly i algorithm servers to the left and exactly j adversary servers to the left.
- $\alpha_{i,j} = \max\{0, \min\{s_{i+1}, a_{j+1}\} - \max\{s_i, a_j\}\}$
- Example,



$$\alpha_{3,0} = 0, \alpha_{4,1} = 0, \ldots$$

R–LINE Details

Isolation Index Coefficients

R–LINE Details

Isolation Index Coefficients

- Every isolation index, α, has an associated coefficient, η.

R–LINE Details

Isolation Index Coefficients

- Every isolation index, α, has an associated coefficient, η.
- R-LINE is defined in terms of these constants, η.

R–LINE Details

Isolation Index Coefficients

- Every isolation index, α, has an associated coefficient, η.
- R-LINE is defined in terms of these constants, η.
- Define constants $\eta_{i,j}$, the $(i,j)^{th}$ *isolation index coefficient*.

R–LINE Details

Isolation Index Coefficients

- Every isolation index, α, has an associated coefficient, η.
- R-LINE is defined in terms of these constants, η.
- Define constants $\eta_{i,j}$, the $(i,j)^{th}$ *isolation index coefficient*.
- The isolation index coefficients satisfy a symmetry property,

$$\eta_{i,j} = \eta_{2n-i,2-j}$$

R–LINE Details

Isolation Index Coefficients

- Every isolation index, α, has an associated coefficient, η.
- R-LINE is defined in terms of these constants, η.
- Define constants $\eta_{i,j}$, the $(i,j)^{th}$ *isolation index coefficient*.
- The isolation index coefficients satisfy a symmetry property,

$$\eta_{i,j} = \eta_{2n-i,2-j}$$

- We also have $\eta_{0,0} = \eta_{2n,n} = 0$.

R–LINE Details

Definition of the Potential, ϕ

R–LINE Details

Definition of the Potential, ϕ

- For any configuration, the potential is defined as the sum of all isolation indices multiplied by their associated coefficients.

R–LINE Details

Definition of the Potential, ϕ

- For any configuration, the potential is defined as the sum of all isolation indices multiplied by their associated coefficients.
- Formally,

$$\phi = \sum \eta_{i,j} \cdot \alpha_{i,j}$$

R–LINE Details

Configurations

R–LINE Details

Configurations

Notation

R–LINE Details

Configurations

Notation

- We refer to s_i as the i^{th} server and also its location.

R–LINE Details

Configurations

Notation

- We refer to s_i as the i^{th} server and also its location.
- We number the servers left to right.

R–LINE Details

Configurations

Notation

- We refer to s_i as the i^{th} server and also its location.
- We number the servers left to right.
- $s_1 \leq s_2 \leq ... \leq s_{2n}$

R–LINE Details

Configurations

Notation

- We refer to s_i as the i^{th} server and also its location.
- We number the servers left to right.
- $s_1 \leq s_2 \leq \ldots \leq s_{2n}$
- Current request is r.

R–LINE Details

Configurations

Notation

- We refer to s_i as the i^{th} server and also its location.
- We number the servers left to right.
- $s_1 \leq s_2 \leq ... \leq s_{2n}$
- Current request is r.
- Previous request is r'.

R–LINE Details

Configurations

Notation

- We refer to s_i as the i^{th} server and also its location.
- We number the servers left to right.
- $s_1 \leq s_2 \leq ... \leq s_{2n}$
- Current request is r.
- Previous request is r'.
- WLOG $r' < r$.

R–LINE Details

Configurations

Notation

- We refer to s_i as the i^{th} server and also its location.
- We number the servers left to right.
- $s_1 \leq s_2 \leq ... \leq s_{2n}$
- Current request is r.
- Previous request is r'.
- WLOG $r' < r$.
- Servers do not pass each other.

R–LINE Details

S-Configuration (Satisfying)

R–LINE Details

S-Configuration (Satisfying)

R–LINE Details

S-Configuration (Satisfying)

- There are n servers at the request point.

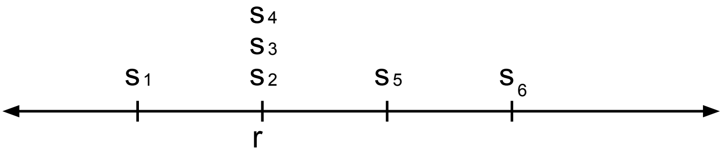
$(6,3)$ Example

R–LINE Details

S-Configuration (Satisfying)

- There are *n* servers at the request point.

$(6, 3)$ Example



R–LINE Details

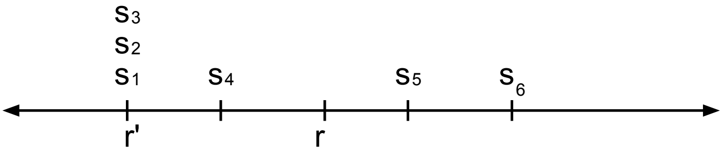
D-Configuration (Deterministic)

R–LINE Details

D-Configuration (Deterministic)

- 1. More than n algorithm servers either strictly to the left or strictly to the right of r; $r > s_{n+1}$ or $r < s_n$.
  2. If fewer than n algorithm servers at r'
     - 2.1 No algorithm server strictly between r' and r
     - 2.2 At least n algorithm servers at the points r' and r combined.

$(6, 3)$ Example

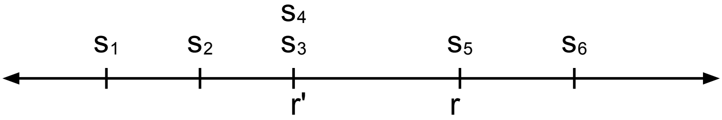


R–LINE Details

D-Configuration (Deterministic)

- 1. More than n algorithm servers either strictly to the left or strictly to the right of r; $r > s_{n+1}$ or $r < s_n$.
  2. If fewer than n algorithm servers at r'
     2.1 No algorithm server strictly between r' and r
     2.2 At least n algorithm servers at the points r' and r combined.

$(6, 3)$ Example



R–LINE Details

D-Configuration Moves

1. Must be m servers to the left of r, for some $m > n$.

R–LINE Details

D-Configuration Moves

1. Must be m servers to the left of r, for some $m > n$.
2. Move $s_{n+1} ... s_m$ to r.

R–LINE Details

D-Configuration Moves

1. Must be m servers to the left of r, for some $m > n$.
2. Move $s_{n+1}...s_m$ to r.

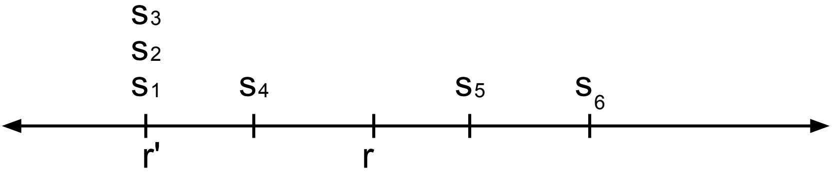
$(6, 3)$ Example

R–LINE Details

D-Configuration Moves

1. Must be m servers to the left of r, for some $m > n$.
2. Move $s_{n+1} \ldots s_m$ to r.

$(6, 3)$ Example

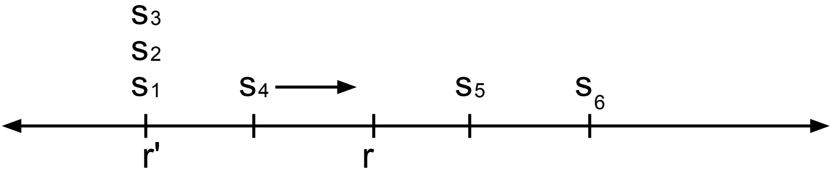


R–LINE Details

D-Configuration Moves

1. Must be m servers to the left of r, for some $m > n$.
2. Move $s_{n+1} \ldots s_m$ to r.

$(6, 3)$ Example

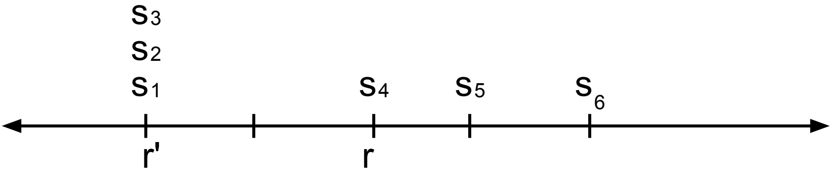


R–LINE Details

D-Configuration Moves

1. Must be m servers to the left of r, for some $m > n$.
2. Move $s_{n+1} \ldots s_m$ to r.

$(6, 3)$ Example

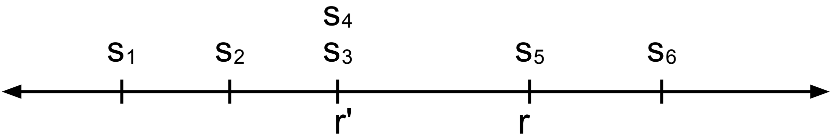


R–LINE Details

D-Configuration Moves

1. Must be m servers to the left of r, for some $m > n$.
2. Move $s_{n+1}...s_m$ to r.

$(6, 3)$ Example

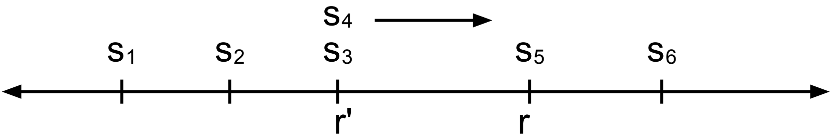


R–LINE Details

D-Configuration Moves

1. Must be m servers to the left of r, for some $m > n$.
2. Move $s_{n+1} \ldots s_m$ to r.

$(6, 3)$ Example

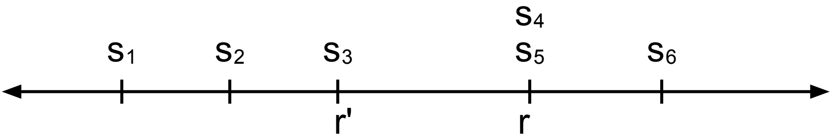


R–LINE Details

D-Configuration Moves

1. Must be m servers to the left of r, for some $m > n$.
2. Move $s_{n+1}...s_m$ to r.

$(6,3)$ Example



R–LINE Details

R-Configurations (Randomized)

R–LINE Details

R-Configurations (Randomized)

The adversary's hidden server.

R–LINE Details

R-Configurations (Randomized)

The adversary's hidden server.

- The adversary has two servers.

R–LINE Details

R-Configurations (Randomized)

The adversary's hidden server.

- The adversary has two servers.
- The current request, r.

R–LINE Details

R-Configurations (Randomized)

The adversary's hidden server.

- The adversary has two servers.
- The current request, r.
- The other server's location, a, is "hidden".

R–LINE Details

R-Configurations (Randomized)

The adversary's hidden server.

- The adversary has two servers.
- The current request, r.
- The other server's location, a, is "hidden".
- There are only two hidden server locations to consider.

$(6, 3)$ Example

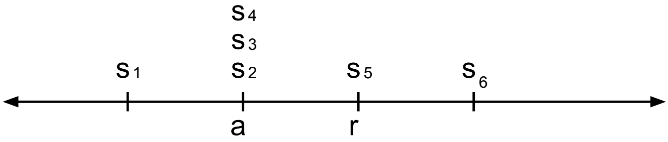
R–LINE Details

R-Configurations (Randomized)

The adversary's hidden server.

- The adversary has two servers.
- The current request, r.
- The other server's location, a, is "hidden".
- There are only two hidden server locations to consider.

$(6, 3)$ Example



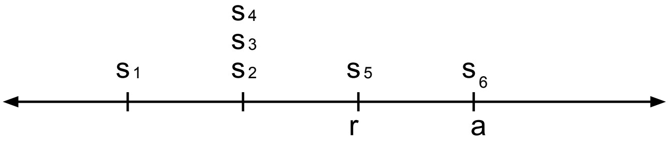
R–LINE Details

R-Configurations (Randomized)

The adversary's hidden server.

- The adversary has two servers.
- The current request, r.
- The other server's location, a, is "hidden".
- There are only two hidden server locations to consider.

$(6, 3)$ Example



R–LINE Details

R-Configuration (Randomized)

R–LINE Details

R-Configuration (Randomized)

1. Exactly n algorithm servers on the same side of r as r'. Either $r' = s_n < r$ or $r < r' = s_{n+1}$.

R–LINE Details

R-Configuration (Randomized)

1. Exactly n algorithm servers on the same side of r as r'. Either $r' = s_n < r$ or $r < r' = s_{n+1}$.
2. No algorithm server strictly between r' and r.

R–LINE Details

R-Configuration (Randomized)

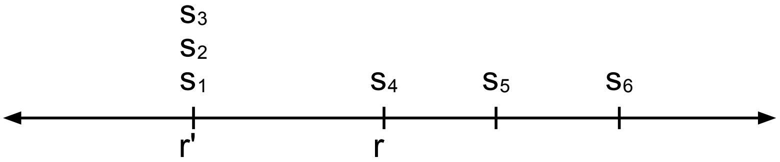
1. Exactly n algorithm servers on the same side of r as r'. Either $r' = s_n < r$ or $r < r' = s_{n+1}$.
2. No algorithm server strictly between r' and r.
3. At least n algorithm servers at the points r' and r combined.

R–LINE Details

R-Configuration (Randomized)

1. Exactly n algorithm servers on the same side of r as r'. Either $r' = s_n < r$ or $r < r' = s_{n+1}$.
2. No algorithm server strictly between r' and r.
3. At least n algorithm servers at the points r' and r combined.

$(6, 3)$ Example



R–LINE Details

R-Configuration (Randomized) Moves

R–LINE Details

R-Configuration (Randomized) Moves

- There are two possible moves.
  1. Move a single server.

R–LINE Details

R-Configuration (Randomized) Moves

- There are two possible moves.
  1. Move a single server.
  2. Complete the request using the servers from r'.

R–LINE Details

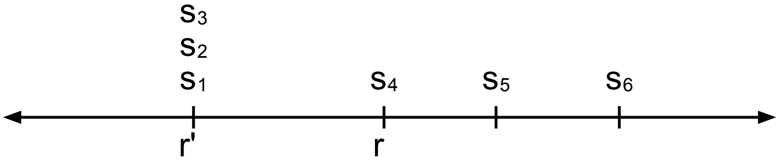
R-Configuration (Randomized) Moves

- There are two possible moves.
  1. Move a single server.
  2. Complete the request using the servers from r'.
- Choose between the two alternatives using randomization, by solving a 2-person zero-sum game.

R–LINE Details

R-Configuration (Randomized) Moves

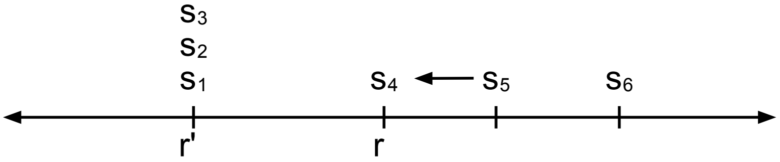
- There are two possible moves.
  1. Move a single server.
  2. Complete the request using the servers from r'.
- Choose between the two alternatives using randomization, by solving a 2-person zero-sum game.



R–LINE Details

R-Configuration (Randomized) Moves

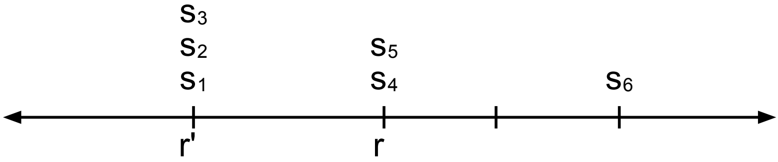
- There are two possible moves.
  1. Move a single server.
  2. Complete the request using the servers from r'.
- Choose between the two alternatives using randomization, by solving a 2-person zero-sum game.



R–LINE Details

R-Configuration (Randomized) Moves

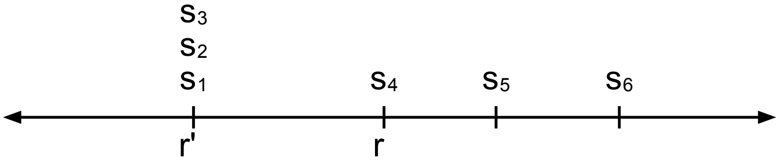
- There are two possible moves.
  1. Move a single server.
  2. Complete the request using the servers from r'.
- Choose between the two alternatives using randomization, by solving a 2-person zero-sum game.



R–LINE Details

R-Configuration (Randomized) Moves

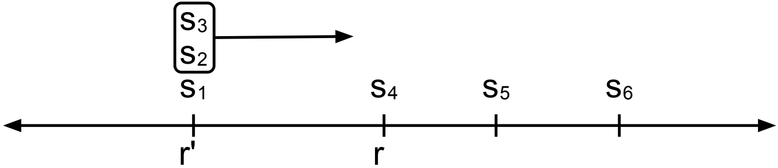
- There are two possible moves.
  1. Move a single server.
  2. Complete the request using the servers from r'.
- Choose between the two alternatives using randomization, by solving a 2-person zero-sum game.



R–LINE Details

R-Configuration (Randomized) Moves

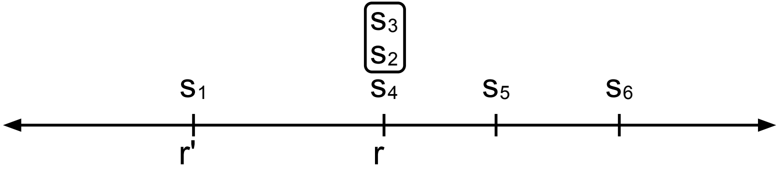
- There are two possible moves.
  1. Move a single server.
  2. Complete the request using the servers from r'.
- Choose between the two alternatives using randomization, by solving a 2-person zero-sum game.



R–LINE Details

R-Configuration (Randomized) Moves

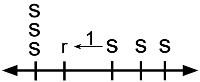
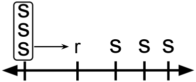
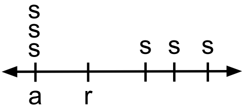
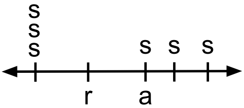
- There are two possible moves.
  1. Move a single server.
  2. Complete the request using the servers from r'.
- Choose between the two alternatives using randomization, by solving a 2-person zero-sum game.



R–LINE Details

Configurations: R-Configurations (Randomized)

Configurations: R-Configurations (Randomized)

		
	$(\Delta\phi + cost)_{11}$	$(\Delta\phi + cost)_{12}$
	$(\Delta\phi + cost)_{21}$	$(\Delta\phi + cost)_{22}$

R–LINE Details

Configurations: R-Configurations (Randomized)

R–LINE Details

Configurations: R-Configurations (Randomized)

$(\Delta\phi + cost)_{11}$	$(\Delta\phi + cost)_{12}$
$(\Delta\phi + cost)_{21}$	$(\Delta\phi + cost)_{22}$

R–LINE Details

Configurations: R-Configurations (Randomized)

$(\Delta\phi + cost)_{11}$	$(\Delta\phi + cost)_{12}$
$(\Delta\phi + cost)_{21}$	$(\Delta\phi + cost)_{22}$

- Entries game matrix computed in terms of the isolation index coefficients, η.

R–LINE Details

Configurations: R-Configurations (Randomized)

$(\Delta\phi + cost)_{11}$	$(\Delta\phi + cost)_{12}$
$(\Delta\phi + cost)_{21}$	$(\Delta\phi + cost)_{22}$

- Entries game matrix computed in terms of the isolation index coefficients, η.
- If currently p servers located at r:

R–LINE Details

Configurations: R-Configurations (Randomized)

$(\Delta\phi + cost)_{11}$	$(\Delta\phi + cost)_{12}$
$(\Delta\phi + cost)_{21}$	$(\Delta\phi + cost)_{22}$

- Entries game matrix computed in terms of the isolation index coefficients, η.
- If currently p servers located at r:

$$
\begin{aligned}
(\Delta\phi + cost)_{11} &= (\eta_{n+p+1,2} - \eta_{n+p,2} + 1) \cdot (s_{n+p+1} - r) \\
(\Delta\phi + cost)_{12} &= (\eta_{p,1} - \eta_{n,1} + n - p) \cdot (r - s_n) \\
(\Delta\phi + cost)_{21} &= (\eta_{n+p+1,1} - \eta_{n+p,1} + 1) \cdot (s_{n+p+1} - r) \\
(\Delta\phi + cost)_{22} &= (\eta_{p,0} - \eta_{n,0} + n - p) \cdot (r - s_n)
\end{aligned}
$$

R–LINE Details

The Algorithm R–LINE

R–LINE Details

The Algorithm R–LINE

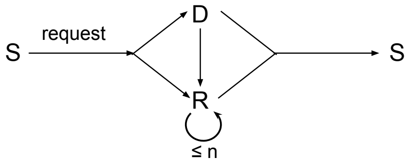
For a given round of execution:

R–LINE Details

The Algorithm R–LINE

For a given round of execution:

1. Start in S-Config. Receive a request.
2. If D–Config, make deterministic moves.
   2.1 If result is S–Config, done.
   2.2 Otherwise result is R–Config.
3. If R–Config, make randomized moves until S–Config.



Proof of Competitiveness

Overview of Proof

1. Provide a system of inequalities, $\mathbb{S}$, involving the isolation index coefficients, $\eta_{i,j}$, and the competitiveness, C.
2. Show that if there exists an assignment of values to every $\eta_{i,j}$ that satisfies $\mathbb{S}$, then R–LINE is C-competitive.
3. Use numeric methods to find a solution to $\mathbb{S}$ that minimizes C.

Proof of Competitiveness

Sufficient Inequalities, $\mathbb{S}_n$

$$\begin{aligned}
\forall\, 0 \le i \le 2n:\ |\eta_{i,1} - \eta_{i,0}| &\le n \cdot C && (1)\\
\forall\, 1 \le i \le n \text{ and } \forall\, 1 \le j \le 2:\ \eta_{i,j} + 1 &\le \eta_{i-1,j} && (2)\\
\forall\, 1 \le i \le n \text{ and } \forall\, 1 \le j \le 2:\ \eta_{i-1,j-1} &\le \eta_{i,j-1} + 1 && (3)\\
\forall\, 1 \le i \le n:\ (\eta_{i-1,1} - \eta_{i,1} + 1)(\eta_{n-i,1} - \eta_{n,1} + i) &\le (\eta_{i-1,0} - \eta_{i,0} + 1)(\eta_{n-i,0} - \eta_{n,0} + i) && (4)
\end{aligned}$$

Proof of Competitiveness

Overview of Proof Steps

Show that if the system of inequalities, $\mathbb{S}_n$, is satisfied, then the following properties hold:

Proof of Competitiveness

Overview of Proof Steps

Show that if the system of inequalities, $\mathbb{S}_n$, is satisfied, then the following properties hold:

1. For adversary moves: $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$.

Proof of Competitiveness

Overview of Proof Steps

Show that if the system of inequalities, $\mathbb{S}_n$, is satisfied, then the following properties hold:

1. For adversary moves: $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$.
2. For R–LINE deterministic moves: $\Delta\phi + cost \leq 0$.

Proof of Competitiveness

Overview of Proof Steps

Show that if the system of inequalities, $\mathbb{S}_n$, is satisfied, then the following properties hold:

1. For adversary moves: $\Delta\phi \leq C \cdot \mathit{cost}_{\mathcal{A}dv}$.
2. For R–LINE deterministic moves: $\Delta\phi + \mathit{cost} \leq 0$.
3. WLOG, adversary's hidden server is at one of at most two possible locations.

Proof of Competitiveness

Overview of Proof Steps

Show that if the system of inequalities, $\mathbb{S}_n$, is satisfied, then the following properties hold:

1. For adversary moves: $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$.
2. For R–LINE deterministic moves: $\Delta\phi + cost \leq 0$.
3. WLOG, adversary's hidden server is at one of at most two possible locations.
4. For R–LINE randomized moves: $\mathrm{E}(\Delta\phi + cost) \leq 0$.

Proof of Competitiveness

Overview of Proof Steps

Show that if the system of inequalities, $\mathbb{S}_n$, is satisfied, then the following properties hold:

1. For adversary moves: $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$.
2. For R–LINE deterministic moves: $\Delta\phi + cost \leq 0$.
3. WLOG, adversary's hidden server is at one of at most two possible locations.
4. For R–LINE randomized moves: $\mathrm{E}(\Delta\phi + cost) \leq 0$.

Adding all of the inequalities over a request round:

$$E(cost_A(\sigma)) \leq C \cdot cost_{\mathcal{ADV}}(\sigma) + K$$

Solving the Inequalities

Given the system $\mathbb{S}_n$

- Find a solution to $\mathbb{S}_n$ that minimizes C.

Solving the Inequalities

Given the system $\mathbb{S}_n$

- Find a solution to $\mathbb{S}_n$ that minimizes C.
- Transform $\mathbb{S}_n$ into a new system $\mathbb{S}'_n$ to simplify the solution.

Solving the Inequalities

Introduce variables ϵ_i and δ_i for all $0 \leq i < n$. Then $\mathbb{S}'_n$ consists of the following constraints:

Solving the Inequalities

Introduce variables ϵ_i and δ_i for all $0 \leq i < n$. Then $\mathbb{S}'_n$ consists of the following constraints:

$$
\begin{array}{rcll}
(2i + \epsilon_{n-i})(2 - \epsilon_i + \epsilon_{i-1}) & = & 4i & \forall\ 0 < i < n \\
(2n + \epsilon_0)(2 + \epsilon_{n-1}) & \geq & 4n & \\
\delta & = & -\epsilon_0/2 & \\
C & = & (2n - \delta)/n & \\
\delta_i & = & \epsilon_i + 2\delta & \forall\ 0 \leq i < n \\
\eta_{i,0} & = & 3i - \delta_i & \forall\ 0 \leq i < n \\
\eta_{i,0} & = & 2n + i - 2\delta & \forall\ n \leq i \leq 2n \\
\eta_{i,1} & = & 2n - i - \delta & \forall\ 0 \leq i < n \\
\eta_{i,1} & = & i - \delta & \forall\ n \leq i \leq 2n \\
\eta_{i,2} & = & \eta_{2n-i,0} & \forall\ 0 \leq i \leq 2n
\end{array}
$$

Solving the Inequalities

Converting to a Differential Equation

- As $n \to \infty$, $\mathbb{S}'_n \to D$
- D is given by:

$$(x + 1 + f(-x)) \cdot (1 - f'(x)) = x + 1$$

$$-1 \leq x \leq 1$$

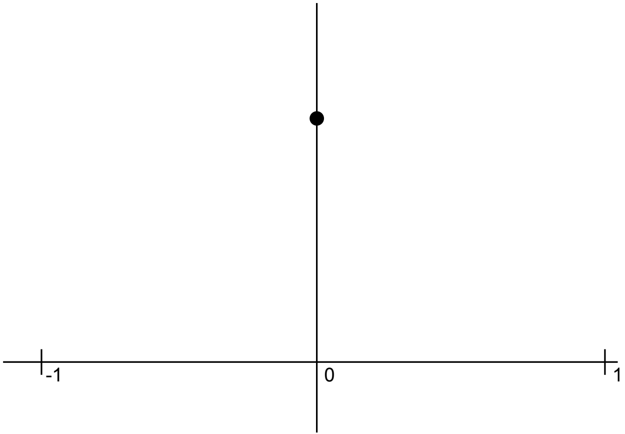
Solving the Inequalities

Euler Method for Reflective Differential Equation

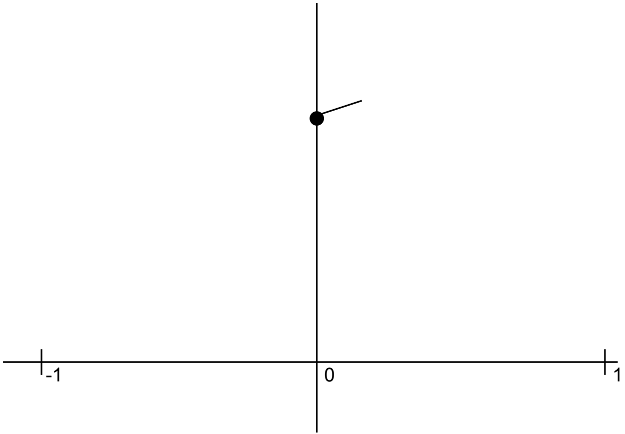
1. Choose a step size, $h = \frac{2}{n}$.
2. Choose an initial value y_0 at t_0
3. Compute updates:

$$y_{n+1} = y_n + h \cdot f(t_{-n}, y_{-n}),$$
$$y_{-n-1} = y_{-n} + h \cdot f(t_n, y_n),$$

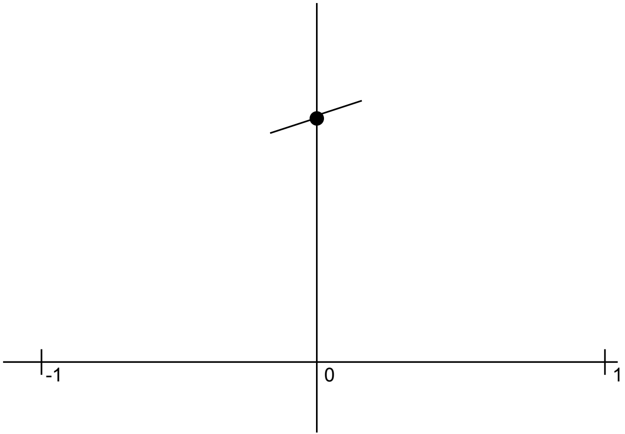
Solving the Inequalities



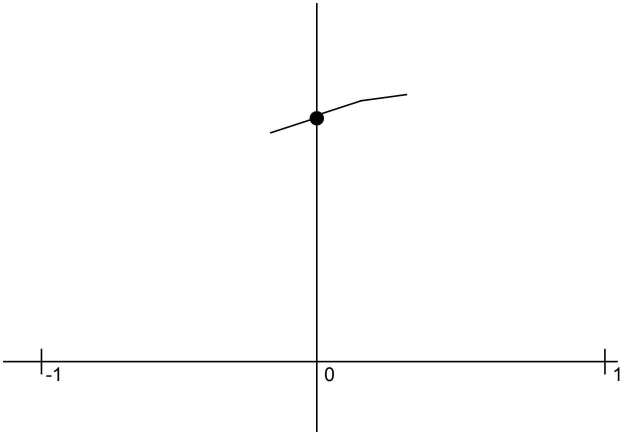
Solving the Inequalities



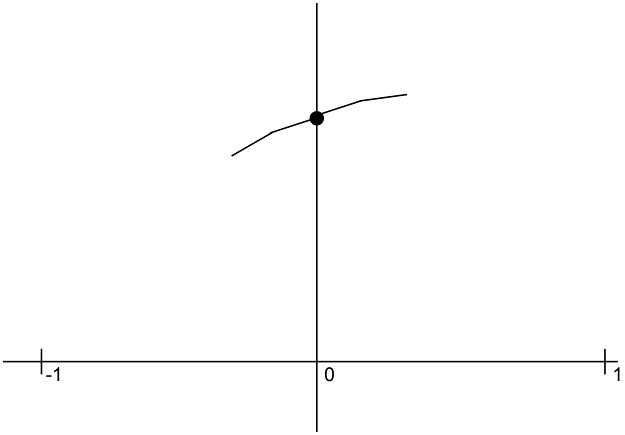
Solving the Inequalities



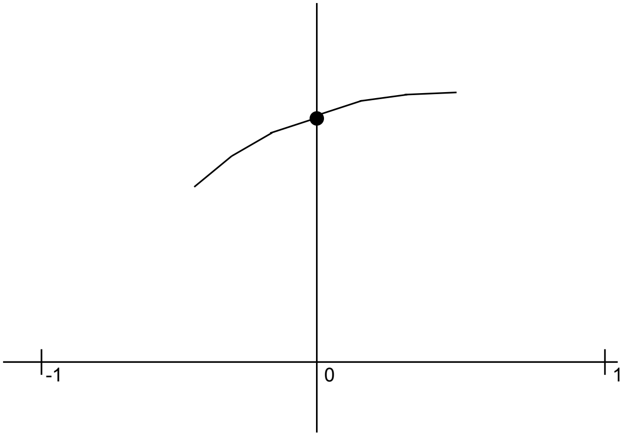
Solving the Inequalities



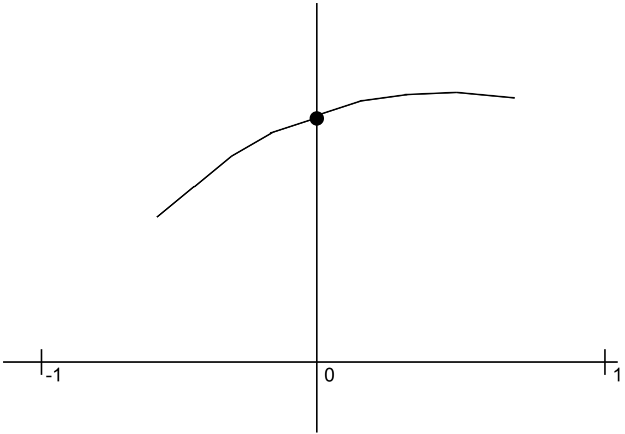
Solving the Inequalities



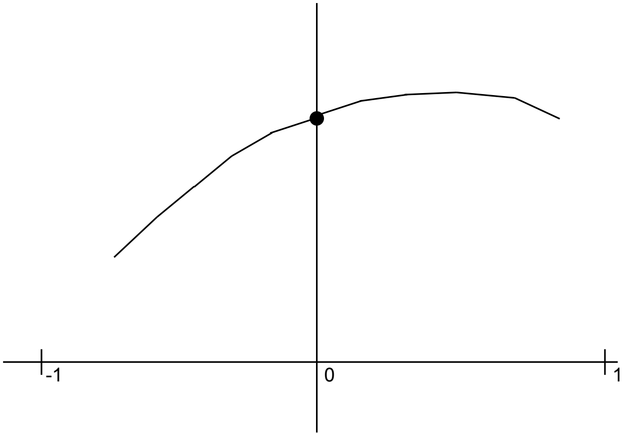
Solving the Inequalities



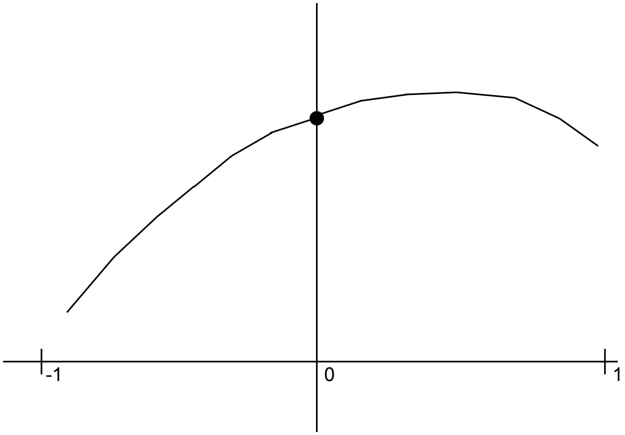
Solving the Inequalities



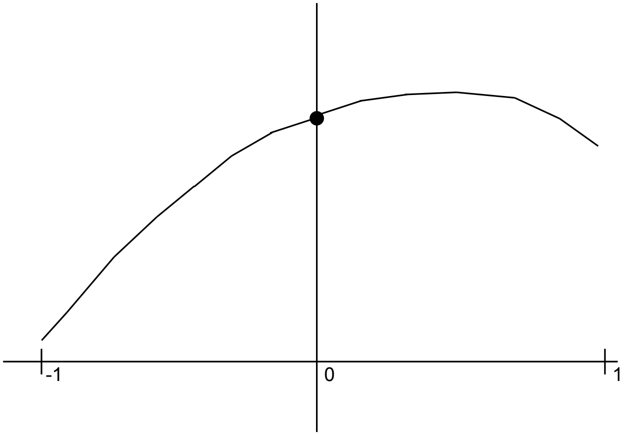
Solving the Inequalities



Solving the Inequalities



Solving the Inequalities



Solving the Inequalities

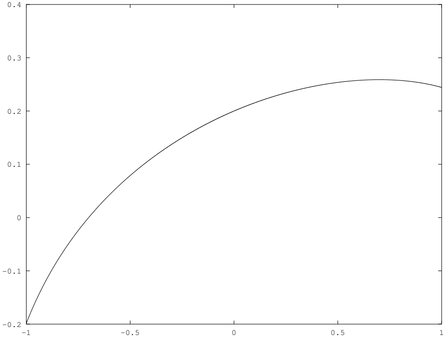
Algorithm to Find Minimum C

1. Choose initial value $f(0)$.
2. Approximate f on interval $[-1, 1]$.
3. Use substitutions to find $\eta_{i,j}$.
4. Verify that $\eta_{i,j}$ satisfy $\mathbb{S}_n$.
5. Compute corresponding value of C.
6. Binary search on $f(0)$ to find minimum C.

Solving the Inequalities

Solution for large n

- We find that $C \approx 1.9007452$ for $n = 10000$.
- The corresponding approximation of $f(x)$:



Future Work

Other Metric Spaces

Future Work

Other Metric Spaces

- Trees - Preliminary work done, strongly suggests $C \leq 1.901$.
- Manhattan Plane, Circle, Euclidean.
- General Spaces!!

Future Work

Other Metric Spaces

- Trees - Preliminary work done, strongly suggests $C \leq 1.901$.
- Manhattan Plane, Circle, Euclidean.
- General Spaces!!

$k > 2$ on the Line

- (kn, n)-server algorithm $\Rightarrow$ k-server algorithm.

Future Work

Other Metric Spaces

- Trees - Preliminary work done, strongly suggests $C \leq 1.901$.
- Manhattan Plane, Circle, Euclidean.
- General Spaces!!

$k > 2$ on the Line

- (kn, n)-server algorithm $\Rightarrow$ k-server algorithm.

Analytic Solution to the Differential Equation

- Would give exact minimum value of C for R–LINE.

Acknowledgements

Thank you!

- Committee Members: Lawrence Larmore, Wolfgang Bein, Matt Pedersen, Ebrahim Salehi.
- UNLV Graduate and Professional Student Association.
- Attendees.

R–LINE

The End.

R–LINE

The End.

R–LINE

The End.

Proof of Competitiveness

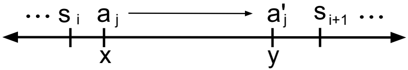
Proof of Property 1

If $\mathbb{S}$ holds, then Property 1 holds: For any move by the adversary, $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$

Proof of Competitiveness

Proof of Property 1

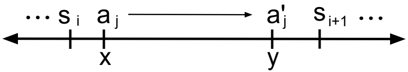
If $\mathbb{S}$ holds, then Property 1 holds: For any move by the adversary, $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$



Proof of Competitiveness

Proof of Property 1

If $\mathbb{S}$ holds, then Property 1 holds: For any move by the adversary, $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$

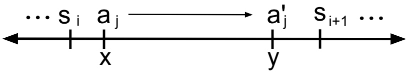


- By inequality (1), $|\eta_{i,j} - \eta_{i,j-1}| \leq n \cdot C$ for $j = 1, 2$.

Proof of Competitiveness

Proof of Property 1

If $\mathbb{S}$ holds, then Property 1 holds: For any move by the adversary, $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$

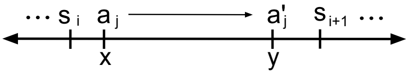


- By inequality (1), $|\eta_{i,j} - \eta_{i,j-1}| \leq n \cdot C$ for $j = 1, 2$.
- Adversary server a_j moves to the right, from x to y, where $x < y$, with $s_i \leq x$ and $y \leq s_{i+1}$.

Proof of Competitiveness

Proof of Property 1

If $\mathbb{S}$ holds, then Property 1 holds: For any move by the adversary, $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$

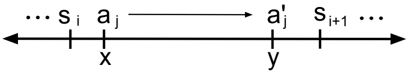


- By inequality (1), $|\eta_{i,j} - \eta_{i,j-1}| \leq n \cdot C$ for $j = 1, 2$.
- Adversary server a_j moves to the right, from x to y, where $x < y$, with $s_i \leq x$ and $y \leq s_{i+1}$.
- Thus, $\alpha_{i,j}$ decreases by $y - x$ and $\alpha_{i,j-1}$ increases by $y - x$.

Proof of Competitiveness

Proof of Property 1

If $\mathbb{S}$ holds, then Property 1 holds: For any move by the adversary, $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$

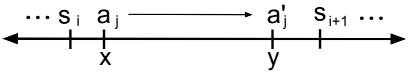


- By inequality (1), $|\eta_{i,j} - \eta_{i,j-1}| \leq n \cdot C$ for $j = 1, 2$.
- Adversary server a_j moves to the right, from x to y, where $x < y$, with $s_i \leq x$ and $y \leq s_{i+1}$.
- Thus, $\alpha_{i,j}$ decreases by $y - x$ and $\alpha_{i,j-1}$ increases by $y - x$.
- The cost to the adversary of this move is $n(y - x)$.

Proof of Competitiveness

Proof of Property 1

If $\mathbb{S}$ holds, then Property 1 holds: For any move by the adversary, $\Delta\phi \leq C \cdot cost_{\mathcal{A}dv}$



- By inequality (1), $|\eta_{i,j} - \eta_{i,j-1}| \leq n \cdot C$ for $j = 1, 2$.
- Adversary server a_j moves to the right, from x to y, where $x < y$, with $s_i \leq x$ and $y \leq s_{i+1}$.
- Thus, $\alpha_{i,j}$ decreases by $y - x$ and $\alpha_{i,j-1}$ increases by $y - x$.
- The cost to the adversary of this move is $n(y - x)$.
- By definition of the potential, $\Delta\phi = (\eta_{i,j} - \eta_{i,j-1})(y - x) \leq n \cdot C \cdot (y - x) \leq C \cdot cost_{\mathcal{A}dv}$.

Proof of Competitiveness

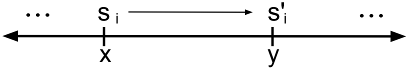
Proof of Property 2

If $\mathbb{S}$ holds, then Property 2 holds: for any deterministic move by R–LINE, $\Delta\phi + cost \leq 0$.

Proof of Competitiveness

Proof of Property 2

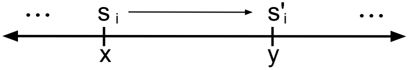
If $\mathbb{S}$ holds, then Property 2 holds: for any deterministic move by R–LINE, $\Delta\phi + cost \leq 0$.



Proof of Competitiveness

Proof of Property 2

If $\mathbb{S}$ holds, then Property 2 holds: for any deterministic move by R–LINE, $\Delta\phi + cost \leq 0$.

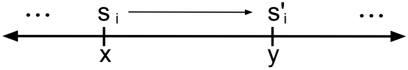


- s_i moves from x to y, where $x < y$.

Proof of Competitiveness

Proof of Property 2

If $\mathbb{S}$ holds, then Property 2 holds: for any deterministic move by R–LINE, $\Delta\phi + cost \leq 0$.

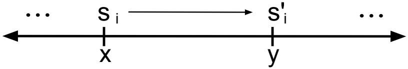


- s_i moves from x to y, where $x < y$.
- The algorithm cost of the step is $y - x$.

Proof of Competitiveness

Proof of Property 2

If $\mathbb{S}$ holds, then Property 2 holds: for any deterministic move by R–LINE, $\Delta\phi + cost \leq 0$.

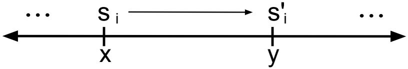


- s_i moves from x to y, where $x < y$.
- The algorithm cost of the step is $y - x$.
- The move causes $\alpha_{i,j}$ to decrease by $y - x$ and $\alpha_{i-1,j}$ to increase by the same amount.

Proof of Competitiveness

Proof of Property 2

If $\mathbb{S}$ holds, then Property 2 holds: for any deterministic move by R–LINE, $\Delta\phi + cost \leq 0$.



- s_i moves from x to y, where $x < y$.
- The algorithm cost of the step is $y - x$.
- The move causes $\alpha_{i,j}$ to decrease by $y - x$ and $\alpha_{i-1,j}$ to increase by the same amount.
- By inequality (2), $\eta_{i,j} + 1 \leq \eta_{i-1,j}$, and the definition of the potential: $\Delta\phi + cost_{R--LINE} = (y - x)(\eta_{i,j} - \eta_{i-1,j} + 1) \leq 0$.

Proof of Competitiveness

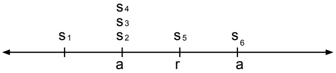
Proof of Property 3

If $\mathbb{S}$ holds, then Property (3) holds: We may assume the adversary's hidden server is at one of at most two possible locations.

Proof of Competitiveness

Proof of Property 3

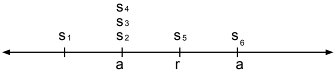
If $\mathbb{S}$ holds, then Property (3) holds: We may assume the adversary's hidden server is at one of at most two possible locations.



Proof of Competitiveness

Proof of Property 3

If $\mathbb{S}$ holds, then Property (3) holds: We may assume the adversary's hidden server is at one of at most two possible locations.

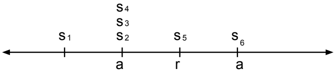


- Since a could be any point on the line, the payoff matrix of the game has infinitely many rows.

Proof of Competitiveness

Proof of Property 3

If $\mathbb{S}$ holds, then Property (3) holds: We may assume the adversary's hidden server is at one of at most two possible locations.

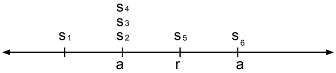


- Since a could be any point on the line, the payoff matrix of the game has infinitely many rows.
- We prove that just two of those rows, namely $a = s_n$ and $a = s_{n+p+1}$, dominate the others.

Proof of Competitiveness

Proof of Property 3

If $\mathbb{S}$ holds, then Property (3) holds: We may assume the adversary's hidden server is at one of at most two possible locations.

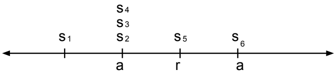


- Since a could be any point on the line, the payoff matrix of the game has infinitely many rows.
- We prove that just two of those rows, namely $a = s_n$ and $a = s_{n+p+1}$, dominate the others.
- By batching the row strategies, we illustrate the $\infty \times 2$ payoff matrix in the next slide.

Proof of Competitiveness

Proof of Property 3

If $\mathbb{S}$ holds, then Property (3) holds: We may assume the adversary's hidden server is at one of at most two possible locations.



- Since a could be any point on the line, the payoff matrix of the game has infinitely many rows.
- We prove that just two of those rows, namely $a = s_n$ and $a = s_{n+p+1}$, dominate the others.
- By batching the row strategies, we illustrate the $\infty \times 2$ payoff matrix in the next slide.

Proof of Competitiveness

Proof of Property 3

		Move s_{n+p+1}	Move $s_{p+1} \dots s_n$
I	$a \le s_n$	$\left(\eta_{n+p+1,2} - \eta_{n+p,2} + 1\right)\left(s_{n+p+1} - r\right)$	$(\eta_{p,1} - \eta_{n,1} + n - p)(r - s_n)$
II	$s_n \le a \le r$	$\left(\eta_{n+p+1,2} - \eta_{n+p,2} + 1\right)\left(s_{n+p+1} - r\right)$	$(\eta_{p,1} - \eta_{n,1} + n - p)(r - a)$ $+$ $(\eta_{p,0} - \eta_{n,0} + n - p)(a - s_n)$
III	$r \le a \le s_{n+p+1}$	$\left(\eta_{n+p+1,2} - \eta_{n+p,2} + 1\right)\left(s_{n+p+1} - a\right)$ $+$ $(\eta_{n+p+1,1} - \eta_{n+p,1} + 1)(a - r)$	$(\eta_{p,0} - \eta_{n,0} + n - p)(r - s_n)$
IV	$a \ge s_{n+p+1}$	$\left(\eta_{n+p+1,1} - \eta_{n+p,1} + 1\right)\left(s_{n+p+1} - r\right)$	$(\eta_{p,0} - \eta_{n,0} + n - p)(r - s_n)$

By inequalities (2) and (3), the rows $a = s_n$ and $a = s_{n+p+1}$ dominate all other rows.

Proof of Competitiveness

Proof of Property 3

	Move s_{n+p+1}	Move $s_{p+1} \ldots s_n$
$a = s_n$	$(\eta_{n+p+1,2} - \eta_{n+p,2} + 1)(s_{n+p+1} - r)$	$(\eta_{p,1} - \eta_{n,1} + n - p)(r - s_n)$
$a = s_{n+p+1}$	$(\eta_{n+p+1,1} - \eta_{n+p,1} + 1)(s_{n+p+1} - r)$	$(\eta_{p,0} - \eta_{n,0} + n - p)(r - s_n)$

By inequalities (2) and (3), the rows $a = s_n$ and $a = s_{n+p+1}$ dominate all other rows.

Proof of Competitiveness

Proof of Property 4

If $\mathbb{S}$ holds, then Property 4 holds: For any randomized move by R–LINE, $E(\Delta\phi + \mathit{cost}) \leq 0$.

Proof of Competitiveness

Proof of Property 4

If $\mathbb{S}$ holds, then Property 4 holds: For any randomized move by R–LINE, $E(\Delta\phi + \mathit{cost}) \leq 0$.

$(\Delta\phi + \mathit{cost})_{11}$	$(\Delta\phi + \mathit{cost})_{12}$
$(\Delta\phi + \mathit{cost})_{21}$	$(\Delta\phi + \mathit{cost})_{22}$

Proof of Competitiveness

Proof of Property 4

If $\mathbb{S}$ holds, then Property 4 holds: For any randomized move by R–LINE, $E(\Delta\phi + cost) \leq 0$.

$(\Delta\phi + cost)_{11}$	$(\Delta\phi + cost)_{12}$
$(\Delta\phi + cost)_{21}$	$(\Delta\phi + cost)_{22}$

- By $\mathbb{S}$, the upper left and lower right entries are negative.

Proof of Competitiveness

Proof of Property 4

If $\mathbb{S}$ holds, then Property 4 holds: For any randomized move by R–LINE, $E(\Delta\phi + cost) \leq 0$.

$(\Delta\phi + cost)_{11}$	$(\Delta\phi + cost)_{12}$
$(\Delta\phi + cost)_{21}$	$(\Delta\phi + cost)_{22}$

- By $\mathbb{S}$, the upper left and lower right entries are negative.
- The upper right and lower left entries are positive.

Proof of Competitiveness

Proof of Property 4

If $\mathbb{S}$ holds, then Property 4 holds: For any randomized move by R–LINE, $\mathrm{E}(\Delta\phi + cost) \le 0$.

$(\Delta\phi + cost)_{11}$	$(\Delta\phi + cost)_{12}$
$(\Delta\phi + cost)_{21}$	$(\Delta\phi + cost)_{22}$

- By $\mathbb{S}$, the upper left and lower right entries are negative.
- The upper right and lower left entries are positive.

Proof of Competitiveness

Proof of Property 4

If $\mathbb{S}$ holds, then Property 4 holds: For any randomized move by R–LINE, $E(\Delta\phi + \mathit{cost}) \leq 0$.

Proof of Competitiveness

Proof of Property 4

If $\mathbb{S}$ holds, then Property 4 holds: For any randomized move by R–LINE, $\mathrm{E}(\Delta\phi + cost) \leq 0$.

- The value of our game is

$$\frac{\det(G)}{(\eta_{n+p+1,2} + \eta_{n+p+1} - \eta_{n+p,2} - \eta_{n+p+1,1}) \cdot (s_{n+p+1} - r) + (\eta_{p,0} + \eta_{n,1} - \eta_{n,0} - \eta_{p,1}) \cdot (r - s_n)}$$

Proof of Competitiveness

Proof of Property 4

If $\mathbb{S}$ holds, then Property 4 holds: For any randomized move by R–LINE, $E(\Delta\phi + cost) \leq 0$.

- The value of our game is

$$\frac{\det(G)}{(\eta_{n+p+1,2} + \eta_{n+p+1} - \eta_{n+p,2} - \eta_{n+p+1,1}) \cdot (s_{n+p+1} - r) + (\eta_{p,0} + \eta_{n,1} - \eta_{n,0} - \eta_{p,1}) \cdot (r - s_n)}$$

- The numerator is non-negative by inequality 4. The denominator is negative, which we can prove by combining inequalities of $\mathbb{S}$ labeled (2) and (3).

Proof of Competitiveness

Proof of Property 4

If $\mathbb{S}$ holds, then Property 4 holds: For any randomized move by R–LINE, $\mathrm{E}(\Delta\phi + cost) \leq 0$.

- The value of our game is

$$\frac{\det(G)}{(\eta_{n+p+1,2} + \eta_{n+p+1} - \eta_{n+p,2} - \eta_{n+p+1,1}) \cdot (s_{n+p+1} - r) + (\eta_{p,0} + \eta_{n,1} - \eta_{n,0} - \eta_{p,1}) \cdot (r - s_n)}$$

- The numerator is non-negative by inequality 4. The denominator is negative, which we can prove by combining inequalities of $\mathbb{S}$ labeled (2) and (3).
- Thus, $E(\Delta\phi + cost_{R--LINE}) = v(G) \leq 0$.

Proof of Competitiveness

Thus, by properties (1), (2), (3), and (4), if $\mathbb{S}$ is satisfied then R–LINE is C-competitive.