Introduction

- **Online Algorithms** are algorithms that need to make decisions without full knowledge of the input.
  - Full knowledge of the past but no or inexact knowledge of the future
  - High degree of uncertainly
- Evaluation: *competitive* to within a constant factor of the *optimum offline algorithm*
  - OPT: the algorithm that has perfect knowledge of the future

Outline

- Introduction to Online Algorithm
- Online Financial Problems
  - Ski Rental Problem
  - Bahncard Problem
  - Money Exchange
- Q&A?

Competitive Ratio

- For a given optimization problem input, \( I \), let \( OPT(I) \) be the optimum solution on input \( I \)
- For a given algorithm, \( ALG \), and input sequence, \( I \), let \( ALG(I) \) be the cost incurred be \( ALG \) on input \( I \)
- An online algorithm is said to be *c-competitive* if there is a constant \( k \) such that for all finite input sequences \( I \),
  \[
  ALG(I) \leq c \cdot OPT(I) + k
  \]

Financial problems
Ski-Rental, Bahncard, Money Exchange

Presented by Cheng Sin Ying (Vivying)
Email: cs_csyaa@stu.ust.hk
October 7, 2003
What is Ski-Rental Problem?

- A set of skis costs 50 ($x$) dollars per day to rent
- A set of skis costs 500 ($kx$) dollars to buy

Each day you decide whether to
- 1) rent skis or
- 2) buy skis

**Question:** What’s the best online algorithm for minimizing the money you spend?

Buy or Rent?

<table>
<thead>
<tr>
<th>Rent costs: $50</th>
<th>Buy costs: $500</th>
</tr>
</thead>
</table>

- 1st Request?
- 2nd Request?
- ...
- 5th Request?
- ...
- 9th Request?
- 10th Request?

Rent is Better....

Buy is Better....

Financial Problems

- Search problems
  - Search for jobs / employees
- Replacement problems
  - Equipment replacement
- Portfolio selection
  - Investment planning
- Leasing problems
  - Buy/Rent question

Checkpoint

- Introduction to Online Algorithm
- Ski-Rental Problem
  - Background and Problem Definition
  - Offline Analysis and different online algorithms
  - Performance of the optimal online algorithm
- Bahncard Problem
- Money Exchange
**Analysis on Ski-Rental Problem**

- **Case 1:** \( n \leq k \) (e.g. \( n=4 \))
  - Total rent cost: 200 \((nx)\)
  - Total buy cost: 500 \((kx)\)

- **Case 2:** \( n > k \) (e.g. \( n = 20 \))
  - Total rent cost: 1000 \((nx)\)
  - Total buy cost: 500 \((kx)\)

*Offline optimum cost is* \( \min\{nx, kx\} \)

**Optimal General Online Algorithm**

- **Rent for \((k-1)\) days and buy**
  - Case i) \( 0 \leq n \leq k-1 \)
    - OPT is \( nx \) and ALG is also \( nx \)
    - the competitive ratio in this case is 1

  - Case ii) \( n \geq k \)
    - OPT is \( kx \) and ALG is also \( (k-1)x + kx = (2k-1)x \)
    - the competitive ratio = \( (2k-1)/k = (2-1/k) \).

  *Conclusion: competitive ratio\(=\!(2-1/k)\)*

**SRP – Online Algorithm (I)**

- **Buying on the first day**
  - Case i) \( n=1 \)
    - ALG is \( kx \) and OPT is \$x.\)
    - The competitive ratio equals \( k \).

  - Case ii) \( 1 < n < k \)
    - ALG is \( kx \) and OPT is \( nx \).
    - The ratio will be gradually decrease from \( k \) to 1

  - Case iii) \( n \geq k \)
    - ALG is \( kx \) and OPT is also \$kx.
    - The ratio will be 1.

  *Conclusion: competitive ratio equals to \( k \)*
The Bahncard Problem

Generalization of Ski-Rental Problem
- Suppose the MTR company is introducing a new Bahncard card
  - $24 HK dollars
  - 50% discount for any travel
  - Valid for 1 month
- Let BP(C, β, T) denote the Bahncard problem
  - C is the Bahncard costs
  - Reduce the price p of any ticket to β p
  - The card is valid during time T
- MTR Bahncard MBP($24, 0.5, 1 month)

Why “k-1” is the best?

1. If rent for less than k-1 and buy, say k-2
   - The adversary choose n=k-1
     - OPT = $(k-1)x, ALG = $(k-2)x +kx
     - Competitive ratio = (2k-2)/(k-1) = 2

2. If rent for greater than k-1 and buy, say k
   - The adversary choose n=k+1
     - OPT = $kx, ALG = $kx + $kx
     - Competitive ratio = 2k/k = 2

Best Strategy

Checkpoint

- Introduction to Online Algorithm
- Ski Rental Problem
- Bahncard Problem
  - Background and Problem Definition
  - Offline analysis and Typical strategies
  - Lower Bound of Deterministic online algorithm
  - Better deterministic online algorithm
  - Randomized online algorithm
- Money Exchange
Problem Definition

**Critical cost**

\[ C_{\text{crit}} = \frac{C}{1 - \beta} = \frac{24}{1 - 0.5} = 48 \]

**Cheap or Expensive?**
- Depends on partial cost: \( P^l(\sigma) < C_{\text{crit}} \)
- Cheap...

**T-Cost (or T-Recent-Cost)**
- How much spent recently:
  \[ r^\sigma(t) = \sum_{i: T_i \in I} p_i \]
- Measure according to the partial cost

**Regular T-Cost**
- Only concern the cost of regular request:
  \[ r^\sigma_A(t) = \sum_{i: \sigma_i \text{ is a regular request in } I} p_i \]

Checkpoint

- Introduction to Online Algorithm
- Ski Rental Problem
- BahnCard Problem
  - Background and Problem Definition
  - Offline analysis and Typical strategies
  - Lower Bound of Deterministic online algorithm
  - Better deterministic online algorithm
  - Randomized online algorithm
- Money Exchange

Problem Definition

- **A finite sequence of travel requests** \( \sigma = \sigma_1, \sigma_2, \ldots \)
  - \( \sigma_2: \{ \text{Day 2: From Choi Hung to Diamond Hill} \} \)
  - \( \sigma_3: \{ \text{Day 25: From Choi Hung to Airport} \} \)

- **Cost** on \( \sigma_i \)
  - \( C(\sigma_i) : \text{Cost of travel (From Diamond Hill to Choi Hung)} \)
  - \( C(\sigma_i) = 4 \) (without BahnCard)
  - \( C(\sigma_i) = 2 \) (with BahnCard)

**Reduced request**
- \( \beta p_i \) called Reduced request
- \( p_i \) otherwise called Regular request

**Critical cost**

\[ C_{\text{crit}} = \frac{C}{1 - \beta} = \frac{24}{1 - 0.5} = 48 \]

**Cheap or Expensive?**
- Depends on partial cost: \( P^l(\sigma) < C_{\text{crit}} \)
- Cheap...

**T-Cost (or T-Recent-Cost)**
- How much spent recently:
  \[ r^\sigma(t) = \sum_{i=1}^{T-1} p_i \]
- Measure according to the partial cost

**Regular T-Cost**
- Only concern the cost of regular request:
  \[ r^\sigma_A(t) = \sum_{i: \sigma_i \text{ is a regular request in } I} p_i \]

**B-Schedule**
- the sequence of when to buy BahnCard (s) suggested by an algorithm \( A \)

**T-Cost of A**
- For whole travel period \( \Gamma \), total number of BahnCard:
  \[ C_A(\sigma) = |\Gamma_A(\sigma)| + \sum_{i \geq 1} C_A(\sigma_i) \]

**Partial costs** during time interval \( I \)
- E.g. \( I = \text{(day 1 to day 2)} \):
  \[ P^l(\sigma) = \sum_{i: i \in I} p_i \]
- Money spent by A on tickets during interval \( I \)
  - Excluding the BahnCard cost:
    \[ C_A^l(\sigma) = \sum_{i: i \in I} C_A(\sigma_i) \]
Given $n$ travel requests, we can compute an optimal B-schedule and its minimal cost in time $O(n)$.

Typical Online Algorithm

- **Buy-Never-Algorithm (NEVER)**
  - $1/\beta$-competitive
  - Optimal if $\beta=1$

- **Ticket-Office Algorithm (TOA)**
  - Buy Bahncard iff total cost at least $C_{\text{crit}} = \frac{C}{1-\beta}$
  - $1/\beta$-competitive
  - Can handle expensive request but not cheap request

Deterministic Lower Bound

- No deterministic online algorithm for $\text{BP}(C,\beta,T)$ can be better than $(2-\beta)$-competitive
  - Constant request of arbitrarily small constant during interval $[0,T)$
  - Let $s$ be the accumulated cost (excluding current request)
  - Adversary stops request after buying Bahncard

Checkup
No deterministic online algorithm for BP(C,β,T) can be better than $(2-\beta)$-competitive.

$C_A(s) = s + C + \beta \varepsilon$

- $C + \beta (s + \varepsilon)^{+ s}$ if $s + \varepsilon \leq C_{crit}$
- $C + \beta (s + \varepsilon)^{+ s}$ if $s + \varepsilon \geq C_{crit}$

Competitive Ratio:
- $C_{OPT}(s) = \frac{C + s + B_p}{C + \beta (s + \varepsilon)^{+ s}}$ if $s + \varepsilon \leq C_{crit}$
- $C_{OPT}(s) = \frac{C + s + B_p}{C + \beta (s + \varepsilon)^{+ s}}$ if $s + \varepsilon \geq C_{crit}$

- min value: $s = C_{crit} - \varepsilon$

Better Online Algorithm (I)

Sum-Algorithm (SUM)

- Buy Bahncard at regular request $(t, p)$ iff regular T-cost at time $t$ is at least $C_{crit}$ and SUM does not already have a Bahncard, i.e.

$$n_{SUM}^s(t) = \sum_{i: \sigma_i \text{ is a regular request in } I} p \geq C_{crit}$$

- This algorithm is $(2-\beta)$-Competitive for BP(C,β,T)

<table>
<thead>
<tr>
<th>$s$</th>
<th>$C_{OPT}^s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>$s$ = 25</td>
</tr>
<tr>
<td>35</td>
<td>$s$ = 35</td>
</tr>
<tr>
<td>40</td>
<td>$s$ = 40</td>
</tr>
<tr>
<td>60</td>
<td>$s$ = 60</td>
</tr>
</tbody>
</table>

1.5 – Competitive

Better Online Algorithm (II)

Optimistic-Sum-Algorithm (OSUM)

- Buy Bahncard at regular request $(t, p)$ iff

$$p \geq \frac{C - s(1 - \beta)}{2 - (1 - \beta)}$$

where $s$ is the regular T-cost at $t$.

$$s = \sum_{i=1}^{t-1} n_{SUM}^s(i)$$

- This algorithm is $(2-\beta)$-Competitive for BP(C,β,T)

<table>
<thead>
<tr>
<th>$s$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>$s$ = 25</td>
</tr>
<tr>
<td>35</td>
<td>$s$ = 35</td>
</tr>
<tr>
<td>40</td>
<td>$s$ = 40</td>
</tr>
<tr>
<td>60</td>
<td>$s$ = 60</td>
</tr>
</tbody>
</table>

1.5 – Competitive
### Money Exchange

- **Trading of Money**
  - 1-way trading
  - 2-way trading

- **Exchange rate sequence** $E = e_1, e_2, e_3, \ldots$
  - $e_i$ is the exchange rate at $i^{th}$ day, (i.e. $1 = ?yen$)

- **Question:** What's the best way for maximizing the exchange of money?

### Statistical Adversary

- Generate worst case input sequences that satisfy *statistical properties*
- Money-exchange Rate $[m, M]$:
- Sequence of exchange rate:
  - $n$ days
  - Optimal off-line return > known quantity $n$
  - $(n, n)$-adversary
  - $R_A(E)$ is the *return* of Algorithm $A$

### Randomized Lower Bound

- Randomizing $\Rightarrow$ improves the competitive ratio
- *No randomized online algorithm for BP(C, β, T) can be better than $e/(e-1+β)$-competitive*
- $R-SUM$ and $R-OSUM$ are $2/(1+β)$-competitive for $BP(C, β, T)$
  - Using probability $q=1/(1+β)$
- RAND and RAND2 is $e/(e-1+β)$-competitive

### Checkpoint

- Introduction to Online Algorithm
- Ski-Rental Problem
- Bahncard Problem
- Money Exchange
  - Problem Definition
  - Statistical Adversary and offline analysis
  - Money-Making algorithm
  - Threat-based strategy
### Threat-based strategy

- New modification by Eisuke Dannoura, Kouichi Sakurai (1998)

- Rule 1: at the end of the game, all remaining dollars are exchanged
- Rule 2: except at the end of the game, a purchase may be made only when the current rate is the highest yet seen
- Rule 3: Whenever the exchange rate reaches a new maximum, just enough may be converted as to ensure that a competitive ratio of \( r \) would be obtained if an adversary were to drop the exchange rate to \( m \) and keep it there for the rest of the game

\[ p_{\text{1-way}} \] - competitive for 1-way trading

\[ p_{\text{2-way}} \] - competitive for 2-way trading

- Not yet the optimal...

### Offline algorithm analysis

- \( R_{\text{OPT}}(E) \)
  - Always convert all dollars to yen at local maxima in \( E \)
  - Always convert all yen to dollars at local minima, except at the last transaction

\[
\text{M=15} \quad \text{Return} = \left(\frac{e_2}{e_5}\right) \left(\frac{e_6}{e_7}\right) \left(\frac{e_8}{e_9}\right) = \left(\frac{e_2 e_3 e_4}{e_3 e_4 e_5}\right) \left(\frac{e_6}{e_7}\right) \left(\frac{e_8}{e_9}\right)
\]

### Money-Making Algorithm

- An algorithm A is **money-making** if for any exchange rate sequence \( E \), with \( R_{\text{OPT}}(E) > 1 \), then \( R_A(E) > 1 \)
- Guarantee positive profit
- Threat-based algorithm

### Reference


3. Anna R. Karlin, Claire Kenyon, Dana Randall: "Dynamic TCP acknowledgement and other stories about e/(e-1)". STOC 2001: 502-509


~ Thank You ~

Any Questions?