CS294-1 On-line Computation & Network Algorithms	**Spring 1997**

Lecture 1: January 21

Lecturer: Yair Bartal — *Scribe: Keith Vanderveen*

The course is focused on the Competitive Analysis of algorithms for On-line Problems. Traditional theoretical analysis is concerned with off-line problem where the complete input is given and we look for a time efficient algorithm. In on-line problems the input is not known in advance but instead it is revealed during the operation of the algorithm.

In this lecture we define basic concepts of on-line problems and competitive analysis and give a couple of simple examples.

1.1 Competitive Analysis

On-line algorithms are defined in the context of optimization problems, where we have a set of inputs $\mathcal{I}$, a set of outputs $\mathcal{O}$, and for a given $I \in \mathcal{I}$ we want to compute the $O \in \mathcal{O}$. Each pair (I, O) is associated a cost or a profit. Our goal is to minimize the cost or maximize the profit (depending on whether it is a minimization or a maximization problem).

In the an *on-line problem* the input is given as a sequence $\sigma^i = \sigma_1\sigma_2\sigma_3...\sigma_i$ of requests or events. An *on-line algorithm* dealing with this sequence must generate a response after each event so that the output is a sequence $a = a_1a_2a_3...a_i$. The response must depend only the input seen so far, i.e. there exist functions f_i such that: $a_i = f_i(\sigma^i)$. The **optimal off-line cost** is the cost of the best output for any given input:

$$\mathrm{Cost}_{\mathrm{OPT}} \equiv \inf_{\mathrm{B}}\{\mathrm{Cost}_{\mathrm{B}}(\sigma^i)\}$$

Traditional worst-case analysis is not useful since in many cases one can cause any algorithm to achieve the same worst-case bound (as for the Paging Problem, where an algorithm may have a page fault on every request).

Competitive Analysis introduced by Sleator and Tarjan [ST85] overcomes this difficulty by comparing the cost of an on-line algorithm to that of an optimal prescient algorithm on any sequence of requests.

The *competitive ratio* of algorithm ON is defined as follows:

$$\text{competitive ratio} = \sup_{\sigma} \frac{\mathrm{Cost}_{\mathrm{ON}}(\sigma)}{\mathrm{Cost}_{\mathrm{OPT}}(\sigma)}$$

The above definition requires that the cost of the on-line algorithm will not exceed a constant times the optimal cost for any subsequence. Usually however, when dealing with infinite sequences, initial (constant) costs are neglected and the above definition is replaced by:

$$\text{competitive ratio} = \sup_{\sigma:\text{Cost}_{\text{OPT}}(\sigma)\to\infty} \frac{\text{Cost}_{\text{ON}}(\sigma)}{\text{Cost}_{\text{OPT}}(\sigma)}$$

An alternative definition is as follows: we say an on-line algorithm ON is *c-competitive* iff there exists a constant $a \geq 0$ such that for every request sequence σ:

$$\text{Cost}_{\text{ON}}(\sigma) \leq c \times \text{Cost}_{\text{OPT}}(\sigma) + a.$$

If $a = 0$, then the on-line algorithm is *strictly c-competitive.*

1.2 Examples of On-line Problems

1.2.1 The Ski Problem

The Ski Problem [Karp92] illustrates a basic principle in competitive analysis.

A skier must decide every day she goes skiing whether to rent or buy skis, unless or until she decides to buy them. The skiier doesn't know how many days she will go on skiing before she gets tired of this hobbie. Call this number of days T. The cost to rent skis for a day is 1 unit, while the cost to buy the skis is B units.

The optimal off-line algorithm is to buy the skis the first day if $B < T$ and to rent every day otherwise.

$$\text{Cost}_{\text{OPT}} = \min B, T$$

A generic on-line algorithm is going to rent skis for i days and then buy if $T > i$.

$$\text{Cost}_{\text{ON}} = \begin{cases} T & \text{if } T \leq i \\ i + B & \text{if } T > i \end{cases}$$

Claim 1.1 *The optimal on-line algorithm is obtained by choosing $i = B - 1$ and has competitive ratio $2 - \frac{1}{B}$.*

Proof: First let us compute the competitive ratio of this algorithm.

$$\text{Cost}_{On} \leq \frac{2B-1}{B}\text{Cost}_{\text{OPT}} = (2 - \frac{1}{B})\text{Cost}_{\text{OPT}}.$$

We now show that $i = B - 1$ indeed gives the best possible ratio.

Case 1: $i \leq B - 1$
The adversary chooses $T = i + 1$. This result in the following ratio for the algorithm:

$$\frac{\text{Cost}_{\text{ON}}}{\text{Cost}_{\text{OPT}}} = \frac{i + B}{min\{i+1, B\}} \geq 2 - \frac{1}{i+1} \geq 2 - \frac{1}{B}.$$

Case 2: $i > B - 1$
Again, $T = i + 1$ gives the worst case behavior for the online algorithm:

$$\frac{\text{Cost}_{\text{ON}}}{\text{Cost}_{\text{OPT}}} = \frac{i + B}{B} \geq 2.$$

■

The Ski Principle is the following : When balancing small incremental costs against big one-time costs, you want to delay spending the big cost until you have accumulated roughly the same amount in small costs.

1.2.2 The Stanford Professor's Lost Car Problem

This problem is usually reffered to as the Lost Cow problem [BCR87].

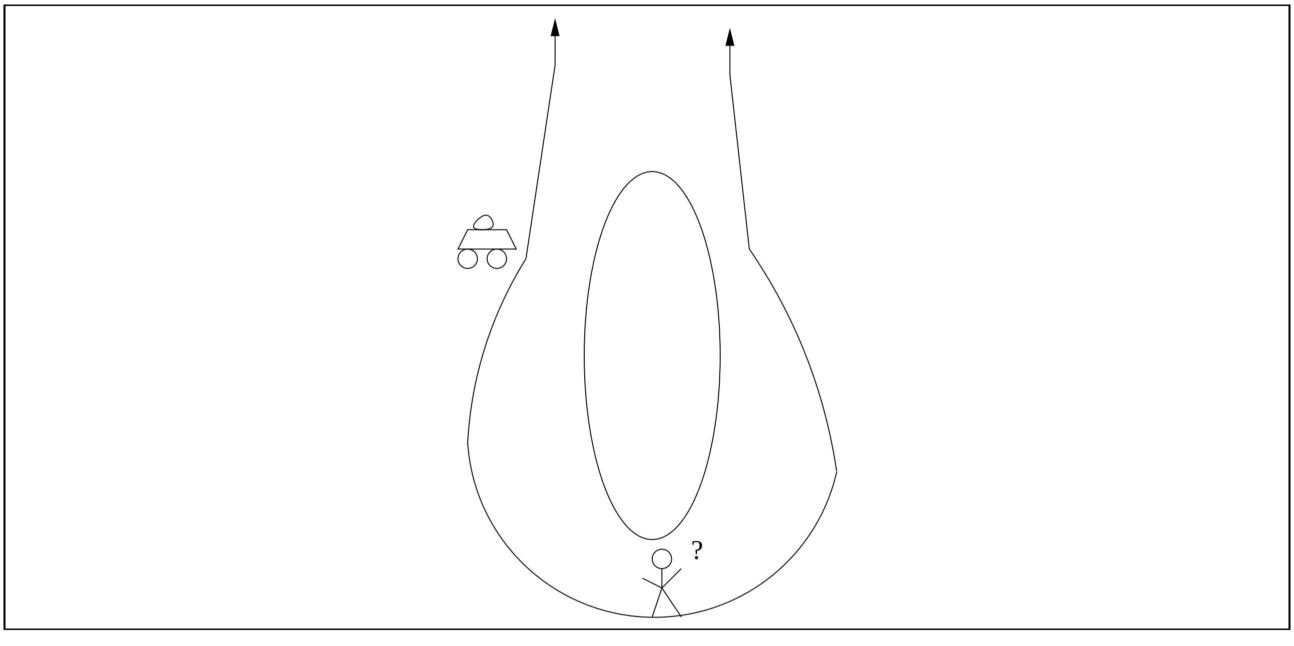

Figure 1.1: Stanford Oval (but infinite)

A Stanford Professor has parked his car on one side of the oval or another, but cannot remember which. The following is an on-line algorithm to find his car:

$d \leftarrow 1$; current_side = left
repeat
    Walk distance d on current_side
    if find car **then** exit
    else return to starting point
    $d \leftarrow 2d$
    Flip current_side

Claim 1.2 $\text{Cost}_{\text{ON}} \leq 9 \times \text{Cost}_{\text{OPT}}$

Proof: The worst case is that we find the car a little bit beyond the distance we last searched on one side, $d = 2^j + \epsilon$ where $j = \#$ of iterations and ϵ is some small distance. Then,

$$\text{Cost}_{\text{OPT}} = d = 2^j + \epsilon \approx 2^j$$

$$\begin{aligned}
\text{Cost}_{\text{ON}} &= 2(1 + 2 + 4 + \ldots + 2^{j+1}) + 2^j \\
&= 2 \cdot 2^{j+2} + 2^j \\
&= 9 \cdot 2^j \\
&= 9 \cdot \text{Cost}_{\text{OPT}}
\end{aligned}$$

∎

References

[BCR87] R.A. Baeza-Yates, J.C. Culberson, and G.J.E. Rawlins. Searching in The Plane. In *Information and Computation*, 106(2), pages 234-252, 1993.

[Karp92] R.M. Karp, On-line Algorithms Versus Off-line Algorithms: How Much is it Worth to Know the Future?. In *Proc. World Computer Congress*, 1992.

[ST85] D.D. Sleator and R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules. In *Communications of the ACM*, 28(2) pages 202-208, 1985.