Introduction to Online Algorithms

Hsiang-Hsuan (Alison) Liu

Most of the problems and algorithms taught in Bachelor's algorithms course are *offline*. That is, an algorithm is allowed to consider the entire input to compute the best solution.

1 Online problems and online algorithms

A problem is *online* if the input instance is revealed piece by piece instead of available all at once. Each piece of instance is called as a *request* or an *event*. An *online algorithm* must make a decision upon the arrival of each request without knowledge about the future. Moreover, the decisions are *irrevocable*. That is, the decisions are permanent and cannot be changed afterwards.

Since we concern about optimization problems in most of the time, we concentrate on the value of the objective incurred by the algorithms. We call it the *cost* incurred by the algorithm, or in short, the *cost* of the algorithm.

Formal definition of online problems

An *online problem* Π consists of a set of *instances* $\mathcal{I}$, a set of feasible *solutions* $\mathcal{O}$, and a cost function $cost : \mathcal{O} \to \mathbb{R}$. Every instance $I \in \mathcal{I}$ is a sequence of *requests* $I = (x_1, x_2, \cdots, x_n)$ and every feasible solution $O \in \mathcal{O}_I$ is a sequence of *answers* $O = (y_1, y_2, \cdots, y_n)$, where $n \in \mathbb{N}^+$. Given an instance I and a corresponding feasible solution O_I, the cost associated with this solution is denoted by $cost(O_I)$. An *optimal solution* for an instance $I \in \mathcal{I}$ of Π is a solution $\mathrm{OPT}(I) \in \mathcal{O}_I$ such that for all legal inputs I,

$$cost(\mathrm{OPT}(I)) = \min_{O \in \mathcal{O}_I} cost(O).$$

If the context is clear, we abuse the notation $\mathrm{ALG}(I)$ to denote the cost of an algorithm ALG on an instance I. For example, we may replace the last equation by

$$\mathrm{OPT}(I) = \min_{O \in \mathcal{O}_\mathcal{I}} cost(O).$$

2 Competitive analysis

The decisions made by the online algorithm are only based on past events but impact the final quality of the online algorithm's overall performance. The limited knowledge about the future and irrevocability of decisions make it difficult to make the optimal decisions. However, we aim at designing an online algorithm such that its incurred cost is comparable to that of an optimal offline algorithm with full knowledge of the future (and unlimited computational power).

We measure the performance of an online algorithm by *competitive analysis*. Consider a minimization problem, an online algorithm ALG is *c-competitive* if for all (finite) input sequence I,

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

If ALG is a c-competitive algorithm, we can also say that "ALG attains a competitive ratio c". The smaller c is, the more cost-efficient ALG is compared to the OPT. Hence, the competitive ratio

of ALG establishes its performance guarantee. Also, we note that the competitive ratio is always at least 1. That is, any online algorithm can be at most as good as the optimal solution is.

Intuitively, considering every instance I of the problem and an algorithm ALG, the ratio $\frac{\text{ALG}(I)}{\text{OPT}(I)}$ is a real number. Imagine that we put all these ratios as "balls" on a real line (see Figure 1). The position of the rightmost ball on the real line is the competitive ratio of this algorithm ALG for the online problem.

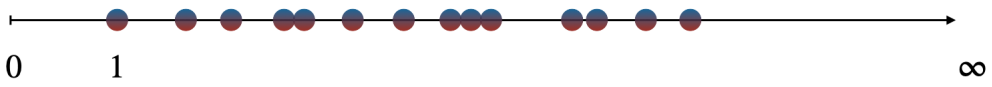

Figure 1: Illustration of competitive ratio

An algorithm is called *competitive* if it attains a constant competitive ratio. For some algorithms, the competitive ratio of an algorithm may be a function of the problem parameters (so it is not a constant), but the competitive ratio has be independent of the instance I.

Theoretically, for online algorithms, we do not care about computational efficiency. Instead, the key point is to investigate how incomplete knowledge affects the decision quality instead of the limit incurred by the computational model. However, in practice, we usually seek efficient competitive online algorithms.

Competitive ratio upper bounds. In most of the cases, an online problem can have infinite number of instances. (Even when there are finite instances, it may be still difficult to know the behavior of an optimal offline algorithm!) Therefore, it is almost impossible to enumerate all possible instances and their corresponding algorithm-optimal ratios and find the biggest one as the competitive ratio. In this case, we can only *bound* the competitive ratio of an algorithm. Intuitively, we want to find a point c on the real line in Figure 1 such that all possible balls (that is, $\frac{\text{ALG}(I)}{\text{OPT}(I)}$ for any possible instance I) are at c or at the left of the point c.

Recall that the competitive ratio is equivalently defined as $\sup_I \frac{\text{ALG}(I)}{\text{OPT}(I)}$. One way to find an upper bound of the competitive ratio is that: given any instance I, find some upper bound x of the $\text{ALG}(I)$ (that is, $\text{ALG}(I) \leq x$) and some lower bound y of the optimal solution on the same instance I (that is, $\text{OPT}(I) \geq y$. By these two bounds, we have $\frac{\text{ALG}(I)}{\text{OPT}(I)} \leq \frac{x}{y}$ for all instance I. And the ratio $\frac{x}{y}$ is a valid *upper bound* of the competitive ratio of ALG. (See the green bound in Figure 2.)

One way to upper bound the competitive ratio of algorithm ALG (Minimization problems)

If for any instance I, $\text{ALG}(I) \leq x$ and $\text{OPT}(I) \geq y$, then

$$\frac{\text{ALG}(I)}{\text{OPT}(I)} \leq \frac{x}{y} \text{ for all } I.$$

That is, the competitive ratio of ALG is at most $\frac{x}{y}$.

Different x's and y's provide different upper bounds of the competitive ratio. Our aim is to find good bounds of x's and y's by useful observations and properties. In many cases, people design online algorithms from the perspective of analysis so the algorithms inherently have nice properties and are easy to be bounded from above.

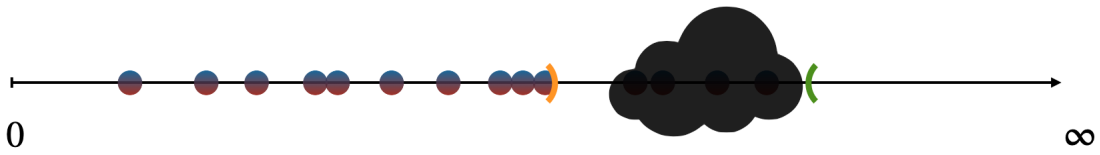

Figure 2: Upper bound (the green bound) and lower bound (the yellow bound) of the competitive ratio of an algorithm

Tight analysis and lower bounds of competitive ratio of an algorithm. Consider an online algorithm ALG, if there is an instance I such that $\frac{\text{ALG}(I)}{\text{OPT}(I)} \geq \ell$, by the definition of the competitive ratio, ALG cannot be c-competitive for any $c < \ell$. We call ℓ a *competitive ratio lower bound of the online algorithm*. Intuitively, the instance I provides a point on the real line, and thus the upper bound of the competitive ratio cannot be on the left of that point. (See the yellow bound in Figure 2.)

Recall that when we bound the competitive ratio from above, different upper bounds of $\text{ALG}(I)$ and different lower bounds of $\text{OPT}(I)$ give different upper bounds of the competitive ratio. A natural question following this is, *how do I know if my upper bound is good enough?*

We can check how good our upper bound analysis is by looking for a matched lower bound of the competitive ratio. That is, find instances I which make $\frac{\text{ALG}(I)}{\text{OPT}(I)}$ as big as possible. Once we found an instance I' such that $\frac{\text{ALG}(I)}{\text{OPT}(I)}$ equals to the upper bound c of the competitive ratio, the analysis is called *tight* and there is no room for improving the analysis. In other words, there is no way to have a smaller upper bound of the competitive ratio of the algorithm since there is a ball right at the position c on the real line.

3 Adversary game and best online algorithms

Game and adversary. The competitive analysis of an online algorithm can be seen as a game between the online algorithm and the adversary, that is, the one who designs the instance. An adversary can torture the online algorithm by revealing an unfortunate piece of input (which is, ideally, not affecting the offline optimal solution cost too much). Note that the adversary can choose the next piece of input according to the algorithm's decisions.

Recall the definition of competitive ratio. An online algorithm is c-competitive if FOR ALL instance, the cost incurred by the online algorithm is no more than c time that incurred by the optimal offline solution for the same instance. In the competitive analysis game, the online algorithm wants to make sure that its competitive ratio is small for any instance, while the adversary tries it best to make the competitive ratio high by designing at least one malicious instance.

Optimal online algorithm and lower bounds of the competitive ratio of a problem. We can even generalize the concept of the lower bound of the competitive ratio of AN algorithm to the lower bound of the competitive ratio of ANY online algorithm.

A lower bound of the competitive ratio of an online problem L is defined as follows:

$$\text{For all online algorithm ALG, there exists an instance } I_{\text{ALG}} \text{ such that } \frac{\text{ALG}(I_{\text{ALG}})}{\text{OPT}(I_{\text{ALG}})} \geq L.$$

That is, for this online problem, there is no (deterministic) c-competitive online algorithm with $c < L$. In other words, there is no (deterministic) algorithm better than L-competitive.

Intuitively, the process of proving the lower bound L of the competitive ratio of an online problem is to show that for any algorithm ALG_i, there must exists at least one instance I_i such that the ball $\frac{\text{ALG}_i(I_i)}{\text{OPT}_i(I_i)}$ is at the position L or at the right of L. Therefore, all online algorithms have competitive ratio at least L. (Note that the instance I_i may have a good ratio on another algorithm ALG_j.)

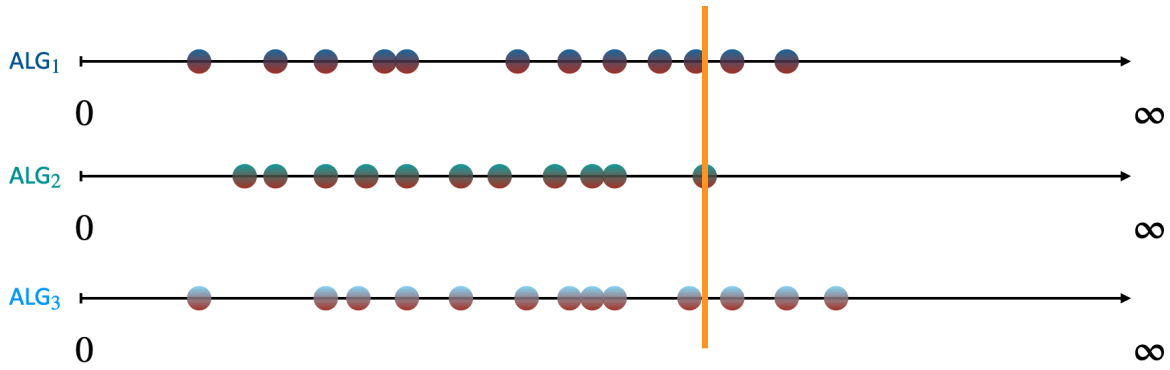

Figure 3: Illustration of one problem competitive ratio lower bound

Recall that the competitive ratio of an online algorithm is its performance guarantee. Consider the case that one have shown that the online problem has a lower bound L. For an online algorithm ALG* which is L-competitive, this algorithm ALG* is an *optimal* (deterministic) online algorithm since there is no other online algorithm outperforms it.