CSC2421: Online and other myopic algorithms
Spring 2021

Allan Borodin

February 11, 2021

Week 5: agenda

Our goal today is to finish the tour of the text chapters and to try to converge as much as possible on initial projects.

Next week is reading week (perfect for a reading course). So I am hoping we can have some presentations starting Thursday, February 25.

Repeating the recurring theme

A recurring theme is bridge the gap between theory and practice. There are different ways this theme is encounterd:

- Assume some partial information is available allowing for more optimistic results. This partial information can be assumed definite knowledge, knowledge that is not fully trusted, or stochastic assumptions.
- The online framework can be relaxed. For example the assumption that decisions are irreversible can be relaxed or one can argue for less powerful adversaries (e.g., the random order model).
- Alternative measures other than the competitive ratio.
- New applications are defined to better model real world applications. One prominent example are models relating to online advertising and other examples of online matching.

We can keep these themes in mind as we walk through the remainiing chapters. If anything seems like a potential topic then please stop me and we can elaborate. I will elaborate on some things close to my current research interests.

Chapter 7: Recent progress

This chapter contains extensions to some of the classic problems, some new applications, and as well as (hopefully) a proof of the latest randomized k server results.

We present some extensions of the ski rental and bin packing

Related to the k-server is the k-taxi problem and the uber problem. We also consider some work related to page migration and and other applications related to distributed computation.

The recent progress for the k-server results is based on convex optimization as developed in online learning. This is a very substantial development.

Chapter 8: The primal dual framework for online algorithms

Primal dual analysis is a major component in algorithm design, algorithm analysis, and approximation.

Young [1991] considered the application of primal dual analysis for file caching.

A major development in the study of online algorithms was introduced by Buchbinder and Naor in 2005. I am posting (on the web page) their 2007 monograph on this general approach to the design of online algorithms.

A number of online algorithms have been developed using this framework.

In particular, randomized algorithms (some of which can be de-randomized) are developed for a number of covering and packing problems, the onliine set cover problem, auctions, and the weighted paging problem. The $O(\log k)$ weighted paging problem gave further evidence in support of the randomized k-server conjecture.

Chapter 9: game theory approaches to online algorithms

This is an area that I think should see much more prominence. Namely, can we develop competitive algorithm when online decisions are being made by self-interested agents.

We discuss the Cohen et al result about a simplified *parking problem* which is modeled as min cost matching on the line. A self interested agent arrives looking for a parking spot nearest to a specific store where the parking spots and the stores are point on the line.

Obviously if there are no costs involved, a driver will take the spot nearest to the store. But this can be shown to have a terrible competitive ratio, namely 2^{2^n} where n is the number of online drivers. (There is an actual story about the "cost of free parking".)

There is an online algorithm, with ratio $O(\log n)$ which is achieved by dynamically charging for dfferent spots. The total cost to the driver is a combination of the cost of the spot plus the distance to walk to the store.

Online algorithms with advice

Chapter 10 begins the Part II of the text where we consider alternative models from the online worst case adverarial model we have been comsidering in Part I of the text.
As we said, one of the ways, we try to improve upon pessistic worst case ratios and to get a better appreciation of "reality", it is reasonable to assume some limited knowledge about any given input instance.
There are two general types of advice.

- Trusted advice. Here the typical goal is to determine the number of advice bits to achieve optimality or a good competitive ratio. The advice bits can be given all at once or with each input arrival.

  There is a direct relation between randomized algorithms using b random bits and algorithms using b bits of advice. Similarly there is a direct relation with algorithms that partition then inputs into some finite number of classes. I am posting a 2016 survey by Boyer et al

Untrusted advice

- Untrusted advice.
  This is sometime promoted as online algorithms with ML advice.

  The goal here is typically to design an algorithm that will do considerably better than known algorithms or any algorithm without advice **IF** the advice turns out to be correct and will not do much worse than existing algorithms if the advice is not correct.

  One can have a probability for the advice being correct or some probabilistic assumptions on how close the advice is to being correct.

  This is a relatively new area of research. I am posting a 2020 survey by Mitzenmacher and Vassilvitskii.

Chapters 12 and 13: Streaming and dynamic algorithms

These two chapters present results in what would usually be considered different research areas. Each is a well studied research area by iteself. But clearly there is a definite online aspect to each of these areas.

Chaper 12 discusses streaming algorithms. Here the most typical (and also quite practical) application is when very large amounts of information are streaming in but there is not enough memory to store all this information. Instead one is usually only trying to appoximate various statistics about the data.

The objective is to use very limnited space (e.g., $O(\log n)$ space) where n is the length of the input stream while maintaing approximate solutions. .

For graph problems, one often is interested in providing a solution which willl require at least $\Omega(n)$ space. The goal is to use only $O(n)$ or $O(n \log^k n)$ space rather than $O(m)$ space where $n = |V|$ and $m = |E|$. This is called semi-streaming as introduced by Feigenbaum et al in 2005.

Chapters 11 and 12 continued

There is no direct way to relate online algorithm and say semi-streaming algorithms but often there are comparable results in both models.

Dynamic algorithms and in particular dynamic graph algorithms refer to algorithms that have to be able to promptly answer queries (e.g., what is the size of a maximum matching) about the current input which is being updated.

If the only allowable updates are insertions (e.g., in a graph a new edge or a new node and its adjacencies) then this is similar to but different from our usual online model.

The more interesting case is the *fully dynamic* setting which allows both insertions and deletions as well as queries.

The goal here is typically to study the tradeoffs between update time and query time and often one settles for amortized results. Note that dynamic algortihms do not have to commit to anything while updating.

Chapters 13: real time scheduling

We already studied scheduling problems in the online setting (e.g. makespan and bin packing are scheduling problems)

Scheduling is a classic topic. As studied in say operations research, online algorithms might better be called *real time* algorithms.

In real time algorithms there is a clock and input events (e.g., jobs arriving) happen at specific times. In real time scheduling we are not typically forced to make a decision when an input arrives but the objective function depends on say the time when jobs complete.

Scheduling problems often allow *preemption* where a job curreently executing on a machine is cancelled or is preempted to be completed at a later time. Preemption can come with a cost, jobs may have to be started from the beginning, or jobs can just resume.

Scheduling problems can be clairvoyant where the duration of a job is known upon arrival or the job or not clairvoyant where the job completion is an online event not known upon arrival.

Chapter 14: Some alternative online models

We return to our more standard meaning of online algorithms where events happen at discrete steps and there is no clock.
We consider variants of the standard model:

- Online algorithms when there is some ability to revoke previous decisions. We can view *display ads with free disposal* as such an algorithm as discussed in Chapter 20. There are also online algorithms with buffers where a small number of inputs can be buffered before being acted upon.
- For some problems we can show how to convert a randomized online algorithm into a deterministic two (or multi-pass) "online algorithm" where in each pass the same input sequence is given.
- For some problems we can run a small number of parallel streams, each stream acting on the same input sequence. And then we can take the best solution amongst the parallel streams.

Chapter 15: Alternative measures of performance

There have been many different measures of performance to hopefully better reflect real world performance or to distinguish between algorithms.

One well studied alternative to competitive analysis is to compare an online algorithm with some augmentation of its resources as compared with an optimal algorithm operating without such an augmentation. For example the (h, k) paging problem.

The chapter considers a number of other performance measures.

Chapter 16: stochastic inputs

As I said earlier, in many applications one has good statistical information about the input items. I am currently working on some research with Calum MacRury in this regard as well as with Chris Karavasilis on some current and past work.

One of the most popular stochastic approaches is to assume that each input item is being generated independently and identically from some known or unknown distribution. This is the i.i.d input model.

This can be further generalized to allow each input item to be generated independently from its own distribution. This is the i.d. input model.

One can also assume that while an adversary determines the set of input items, they arrive in a uniformly random order. This is called the random input model (ROM).

We can use either an adversarial or random order arrival proces for worst case adversarial, i.i.d. and i.d. inputs.

The random order model dates back to (at least) the secretary problem,

The KVV Ranking algorithm renewed intertest in this model as that randomized algorithm for adverasrial order can be seen to be equiavlent to a deterministic algorithm in the random order model.

The goal in the secretary problem is to choose a winner from a sequence of candidates. Each online candidate comes with value and the "employer" has to either choose this candidate (earning that value) passes on this candidate in which case the candidate is no longer available.

In this problem we assume that the number n of candidates is known in advance. This is then an example of a *stopping rule* problem.

The classical secretary problem is to determine the probability that an algorithm determines the best candidate. The optimum probability turns out to be $\frac{1}{e}$ as achieved by an elegant simple algorithm. Namely, we record the best value (say v_{best}) amongst the first $\lfloor n/e \rfloor$ candidates and then accept the first candidate whose value exceeds v_{best} (or accept the last candidate if there is no such candidate exceeding v_{best}).

Extensions and variations of the secretary problem

When the online input items are being drawn i.d. from known distributions and given in an adversarial order and the goal is to choose one "winner", this is called the prophet inequalities problem.

When the online input items are being drawn i.d. from known distributions and given in random order and the goal is to choose one "winner", this is called the prophet secretaries problem.

The secretary, prophet inequalities, and prophet secretaries problems, have all been generalzed to multi item selection (e.g., subject to a matroid constraint, a knapsack constraint, and a matching constraint.)

In the single item setting, $\frac{1}{2}$ is an optimal competitive ratio for the prophet inequalities problem. For the single item prophet secretaries problem, it has recently been shown that competitive ratio .669 can be achieved (surpassing the $1 - 1/e$ "barrier") and can be at most .732.

For the special case of selecting one item in the i.i.d. model, the optimal competitive ratio is .745, another recent result.

Stochastic matching with probing

We also consider online bipartite matching in a probing model. Namely, each edge (u, v) come with a probability that the edge exists. Each online node u has some specified constraint on the set of allowable probes to edges adjacent to u (e.g, each u can be probed at most ℓ times).

In these mathing with probing problems,the standard assumption is probing with *commitment*; that is, if an edge is found to exist (upon probing), then that edge must be incuded in the matching (if possible).

The problem is studied for for worst case inputs for both unknown and known stochastic graphs. Even in the known stochasic graph setting, edges must be probed to see if they exist.

The priority model for greedy and myopic algorithms

Until now we have been considering online algorithms, meaning that the sequence of inputs is not under the control of the algorithm.

Going beyond online algorithms, we introduce the *priority model* which models greedy and myopic algorithms.

Now we allow the algorithm some limited ability to order the inputs, using either

- a fixed order where the ordering of items is set initally before seeing any of the input items or
- an adaptive order, where the choice of the next input item is a function of all the previous items and decisions.

Fixed order priority algorithms have been referred to as *semi-online* algorithm although that term is used in other contexts.

As in online algorithms, upon receiving an input item, a priority algorithm must make an irrevocable decision for that item (unless we allow some sort of revoking as we suggested for online algorithms).

Priority algorithms continued

Priority algorithms model much of what we call greedy algorithms although I like to reserve the meaning of *greedy* to imply some sort of *live for today* requiremnt on the decisions being made. That is, when it is clear from the meaning, each decision is made to be the most optimum decision at this time.

Recall, that the online makespan algorithms that beat $2 - 1/m$ are non-greedy in that they do not necessarily place an item so as to minimize the current makespan. Instead they sometimes make decisions to guard against the future.

One extension to the online and priority models is to place decisions on a stack and then (for example) to pop the stack to insure any required packing constraints. In some cases this framework models *primal dual with reverse delete* algorithms.

Start of Part III, Applications

Chapters 18-23 are more geared to appllications.

Chapter 18 concerns online learning. This topic is very similar to and related to our standard meaning of online algorithms.

One basic problem is making online decisions a_t at each time step t and receiving a cost or reward $c(a_t, b_t)$ where b_t is some action taken *by the environment*. A classical example, is predicting the weather.

The order here is different than say in request answer games since the online action a_t is first taken and then we learn $c(a_t, b_t)$.

Instead of the competitive ratio, the standard measure of performance is called *regret*. For a sequence of T time steps, the (amortized daily) regret (for a cost problem) of an algorithm $\mathcal{A}$ is defined as:

$$R_{\mathcal{A}}(T) = \max_a \frac{1}{T}\left(\sum_{t=1}^{T} c(a, b_t) - \sum_{t=1}^{T} c(a_t, b_t)\right)$$

That is, the regret is with respect to some optimum fixed strategy a.

Chapter 20: Online advertsiing

We previously mentioned that bipartite matching is the underlying problem for a number of online problems that model online advertising.

The two main problems are *adwords* and *display ads (with free disposal)*..

They can be studied for worst case or stochastic inputs in an adversarial or random order. (They could also be studied with respect to an algorithm that can determine the ordering as could be done if the advertsising platform can batch some requests.)

Researchers working in search engine companies are often the people doing the theoretical work here indicating that this area has immendiate application (i.e., the revenue gained by the search engine company). Note that these companies have vast information on search requests and advertisers.

Chapter 21: Applications in Finance

Perhaps the most studied online problem theoretically and in the "real word" is the portfolio selection problem. This problem is studied using both the regret meaure and the competitive ratio.

Seminal work by Cover considerd the regret of a *constantly rebalancing* online algorithm with respect to the best fixed portfolio in hindsight.

Theoretically (and for some historical data) these rebalncing al;gorithm appear to be very financially rewarding. In practice ....

Chapter 21: Applications in Finance

Perhaps the most studied online problem theoretically and in the "real word" is the portfolio selection problem. This problem is studied using both the regret meaure and the competitive ratio.

Seminal work by Cover considerd the regret of a *constantly rebalancing* online algorithm with respect to the best fixed portfolio in hindsight.

Theoretically (and for some historical data) these rebalncing al;gorithm appear to be very financially rewarding. In practice ....

I was once told that "the day you understand the key to the stock market, someone changes the key".

these algorithms and it is one of my most referenced paper. With Ran El Yaniv I had a paper regarding these algorithms and it is one of my most referenced papers and I still get requests. But I am sticking to my "day job".

Chapter 22: Mechanism design

We are probably going to move the material from Chapter 9 to this chapter.

In this area, arguably the most studied topic is online auctions.

One partcularly appealing type of auction is a posted price auction. In posted price auctions, buyers (with valuations for an itema are drawn from some distribution) arrive in sequence and the seller offers a price. If the buyers draw a valuation that is at least the price, the buyer takes the iteam anmd othewise refuses the item.

The sequence of buyers can be determined adversarially, randomly, or by the algorithm. This might sound familiar. When say buyers arrive online (adversarially), this seems very similar to the prophet inequalities problem. And indeed, online posted price auctions (called order oblivious) are equivalent to the prophet inequalities problem.

Chapter 23:Online navigation

This is our final topic and one that is cenbtral to AI search. It doesn't quite fit the request answer famework in a natural way but still is very much online in nature.

While most work in online search is in AI, there is some ork in TCS.