CSC2421: Online and Other Myopic Algorithms Fall 2025

Allan Borodin

September 17, 2025

Week 3

Annoucements

- No class next week (unless you want to meet as a group and think about what might be good research projects for individual or groups of people to work on. I am pleased to say that David Zhang will present a guest lecture on work that he has been doing with Chris Karavasilis and myself on randomness extraction from a ROM source. .
- I plan to post the beginning of the first problem set soon. Note that there are exercises and open problems at the end of most chapters.

Todays agenda

- The proportional and general knapsack problem.
- Priority algorithms
- Extensions of online and priority algorithms

The knapsack problem

The {0,1} knapsack problem

- Input: Knapsack size capacity C and n items $\mathcal{I} = \{I_1, \ldots, I_n\}$ where $I_j = (v_j, s_j)$ with v_j (resp. s_j) the profit value (resp. size) of item I_j.
- Output: A feasible subset $S \subseteq \{1, \ldots, n\}$ satisfying $\sum_{j \in S} s_j \leq C$ so as to maximize $V(S) = \sum_{j \in S} v_j$.

Note: I would prefer to use approximation ratios $r \geq 1$ (so that we can talk unambiguously about upper and lower bounds on the ratio) but many people use approximation ratios $\rho \leq 1$ for maximization problems; i.e. $ALG \geq \rho OPT$. For certain topics (e.g., knapsack problems), fractional approximation and competitive ratios are more the norm.

- It is easy to see (i.e., once you have seen the nemesis sequences) that the most natural deterministic greedy methods (sort by non-increasing profit densities $\frac{v_j}{s_j}$, sort by non-increasing profits v_j, sort by non-decreasing size s_j) will not yield any constant ratio (i.e. greater than some fixed $\epsilon > 0$).
- What other orderings could you imagine?

Randomized Priority and Online Algorithms for Knapsack Problems

Negative results motivate new directions

We already have seen that the online deterministic greedy $2 - \frac{1}{m}$ lower bound for the identical machines makespan problem motivated the offline greedy LPT algorithm.

We will see that no deterministic "greedy algorithm" can obtain a constant approximation. To prove such a lower bound we need a definition for greedy algorithms. We will formalize (and generalize) greedy algorithms by what we call *priority algorithms*.

But first, let's see some positive and negative results for priority and online algorithms with regard to the general and proportional knapsack problems.

A randomized priority algorithm for the general knapsack

Let A_1 (respectively A_2) represent the algorithm when sorting by non-increasing value densities $\frac{v_j}{s_j}$ (respectively, sorting by by non-increasing values v_j). While neither A_1 nor A_2 yield constant approximations, it turns out that "doing both" is a $\frac{1}{2}$ -approximation. That is, the sum of the values provided by A_1 and A_2 is at least as good as an optimal solution. But, of course, this sum will not usually be feasible.

But the fact that the sum can only exceed an optimal solution means that at least one of these two algorithms must provide a $\frac{1}{2}$ approximation.

Hence randomly choosing one of these two algorithms is a $\frac{1}{2}$ approximation in expectation (but not with "high probability"). We call this a 1-bit barely random algorithm.

For the proportional knapsack, the algorithm A_1 and A_2 are identical and hence there is a deterministic greedy algorithm that acheieve a $\frac{1}{2}$ approximation.

Online algorithms for the proportional and general knapsack problems?

Without loss of generality we can assume that the knapsack capacity $C = 1$. Why?

It is not difficult to see that even for the proportional knapsack problem there cannot be a constant detertministic competitive ratio. (See Theorem 3.4.3. in text.)
Remember my warning about "not difficult to see". Same warnings about "similarly we have ...", etc.

Can we have a randomized online algorithm to obtain a good comnpetitive online ratio for the proportional (or general knapsack) problem?

For the proportional knapsack, it is reasonably easy to see that we can obtain a $\frac{1}{4}$ competitive barely random algorithm. Then if there is an input item with value $v_i \geq \frac{1}{2}$, we would have a half approximation. So we can use one bit of randomness to choose between accepting greedily or waiting for an item with value at least $\frac{1}{2}$.

Online algorithms for the knapsack problem continued.

Can we improve upon the $\frac{1}{4}$ competitive ratio for the proportional knapsack problem?

Online algorithms for the knapsack problem continued.

Can we improve upon the $\frac{1}{4}$ competitive ratio for the proportional knapsack problem?

YES. There is a very closely related algorithm (Called TwoBin, Agorithm 3.4.2) which is always trying to fill bin 1 and when an item can't be added to bin 1, it tries to add it to bin 2.

Can we do better by using more random bits?

Online algorithms for the knapsack problem continued.

Can we improve upon the $\frac{1}{4}$ competitive ratio for the proportional knapsack problem?

YES. There is a very closely related algorithm (Called TwoBin, Agorithm 3.4.2) which is always trying to fill bin 1 and when an item can't be added to bin 1, it tries to add it to bin 2.

Can we do better by using more random bits?

NO. Consider two possible input sequences: a single item with value ϵ or a sequence of 2 items: an ϵ item and an item of value 1. See fact 3.4.6. Note that this is a lower bound against an oblivious adversary since the adversary knows the probability with which the algorithm will take the first ϵ valued item.

Can we obtain a constant randomized competitive ratio for the general knapsack?

Online algorithms for the knapsack problem continued.

Can we improve upon the $\frac{1}{4}$ competitive ratio for the proportional knapsack problem?

YES. There is a very closely related algorithm (Called TwoBin, Agorithm 3.4.2) which is always trying to fill bin 1 and when an item can't be added to bin 1, it tries to add it to bin 2.

Can we do better by using more random bits?

NO. Consider two possible input sequences: a single item with value ϵ or a sequence of 2 items: an ϵ item and an item of value 1. See fact 3.4.6. Note that this is a lower bound against an oblivious adversary since the adversary knows the probability with which the algorithm will take the first ϵ valued item.

Can we obtain a constant randomized competitive ratio for the general knapsack? NO See Theorem 3.10.2

Can revoking help?

Online algorithms with revoking are discussed in what is now chapter 16. (It is chapter 15 in the text draft I originally uploaded. I will upload a more recent version this week.)

It is not difficult to see that for the proportional knapsack, we can avoid ramdomization by having the ability to revoke previously accepted items. We call this the *revocable greedy algorithm*. Can you guess the algorithm?

With randomization and revoking there is a .7 competitive algorithm

Is there a deterministic (constant) competitive algorithm with revoking for the general knapsack problem?

Can revoking help?

Online algorithms with revoking are discussed in what is now chapter 16. (It is chapter 15 in the text draft I originally uploaded. I will upload a more recent version this week.)

It is not difficult to see that for the proportional knapsack, we can avoid ramdomization by having the ability to revoke previously accepted items. We call this the *revocable greedy algorithm*. Can you guess the algorithm?

With randomization and revoking there is a .7 competitive algorithm

Is there a deterministic (constant) competitive algorithm with revoking for the general knapsack problem?

NO. See Fact 16.1.13

The priority algorithm model and variants

As part of our discussion of greedy (and greedy-like) algorithms, I want to present the priority algorithm model and how it can be extended in (conceptually) simple ways to go beyond the power of the priority model.

- What is the intuitive nature of a greedy algorithm as exemplified by the algorithmns we discuss in our CSC 373 course? With the exception of Huffman coding (which we can also deal with), like online algorithms, all these algorithms consider one input item in each iteration and make an irrevocable "greedy" decision about that item..
- We are then already assuming that the class of search/optimization problems we are dealing with can be viewed as making a decision D_k about each input item I_k (e.g. on what machine to schedule job I_k in the makespan case) such that $\{(I_1, D_1), \ldots, (I_n, D_n)\}$ constitutes a feasible solution.

Priority model continued

- Note: that a problem is only fully specified when we say how input items are represented. (This is usually implicit in an online algorithm.)
- We mentioned that a "non-greedy" online algorithm for identical machine makespan can improve the competitive ratio; that is, the algorithm does not always place a job on the (or a) least loaded machine (i.e. does not make a greedy or locally optimal decision in each iteration). It isn't always obvious if or how to define a "greedy" decision but for many problems the definition of greedy can be informally phrased as "carpe diem" or "live for today" (i.e. assume the current input item could be the last item) so that the decision should be an optimal decision given the current state of the computation.

Greedy decisions and priority algorithms continued

- For example, in the knapsack problem, a greedy decision always takes an input if it fits within the knapsack constraint and in the makespan problem, a greedy decision always schedules a job on some machine so as to minimize the increase in the makespan. (This is somewhat more general than saying it must place the item on the least loaded machine.)
- In the definition of priority algorithms there is no requirement for being greedy.
- We have both fixed order priority algorithms (e.g. unweighted interval scheduling and LPT makespan) and adaptive order priority algorithms (e.g. the set cover greedy algorithm and Prim's MST algorithm).
- The key concept is to indicate how the algorithm chooses the order in which input items are considered. We cannot allow the algorithm to choose say "an optimal ordering".
- We might be tempted to say that the ordering has to be determined in polynomial time but that gets us into the "tarpit" of trying to prove what can and can't be done in (say) polynomial time.

The priority model definition

- We take an information theoretic viewpoint in defining the orderings we allow.
- Lets first consider deterministic fixed order priority algorithms. Since I am using this framework mainly to argue negative results (e.g. a priority algorithm for the given problem cannot achieve a stated approximation ratio), we will view the semantics of the model as a game between the algorithm and an adversary.
- Initially there is some (possibly infinite) set $\mathcal{J}$ of potential inputs. The algorithm chooses a total ordering π on $\mathcal{J}$. Then the adversary selects a subset $\mathcal{I} \subset \mathcal{J}$ of actual inputs so that $\mathcal{I}$ becomes the input to the priority algorithm. The input items $I_1, \ldots, I_n$ are ordered according to π.
- In iteration k for $1 \leq k \leq n$, the algorithm considers input item I_k and based on this input and all previous inputs and decisions (i.e. based on the current state of the computation) the algorithm makes an irrevocable decision D_k about this input item.

The fixed (order) priority algorithm template

$\mathcal{J}$ is the set of all possible input items
Decide on a total ordering π of $\mathcal{J}$
Let $\mathcal{I} \subset \mathcal{J}$ be the input instance
$S := \varnothing$ % S is the set of items already seen
$i := 0$ % $i = |S|$
while $\mathcal{I} \setminus S \neq \varnothing$ **do**
    $i := i + 1$
    $\mathcal{I} := \mathcal{I} \setminus S$
    $I_i := \min_\pi \{I \in \mathcal{I}\}$
    make an irrevocable decision D_i concerning I_i
    $S := S \cup \{I_i\}$
end

Figure: The template for a fixed priority algorithm

Some comments on the priority model

- A special (but usual) case is that π is determined by a function $f : \mathcal{J} \to \Re$ and and then ordering the set of actual input items by increasing (or decreasing) values $f()$. (We can break ties by say using the input identifier of the item to provide a total ordering of the input set.) N.B. We make no assumption on the complexity or even the computability of the ordering π or function f.
- NOTE: Online algorithms are fixed order priority algorithms where the ordering is given *adversarially*; that is, the items are ordered by the input identifier of the item.
- As stated we do not give the algorithm any additional information other than what it can learn as it gradually sees the input sequence.
- However, we can allow priority algorithms to be given some (hopefully easily computed) global information such as the number of input items, or say in the case of the makespan problem the minimum and/or maximium processing time (load) of any input item. (Some inapproximation results can be easily modified to allow such global information.)

The adaptive priority model template

$\mathcal{J}$ is the set of all possible input items
$\mathcal{I}$ is the input instance
$S := \varnothing$ % S is the set of items already considered
$i := 0$ % $i = |S|$
while $\mathcal{I} \setminus S \neq \varnothing$ **do**
    $i := i + 1$
    decide on a total ordering π_i of $\mathcal{J}$
    $\mathcal{I} := \mathcal{I} \setminus S$
    $I_i := \min_{\leq_{\pi_i}} \{I \in \mathcal{I}\}$
    make an irrevocable decision D_i concerning I_i
    $S := S \cup \{I_i\}$
    $\mathcal{J} := \mathcal{J} \setminus \{I : I \leq_{\pi_i} I_i\}$
    % some items cannot be in input set
end

Figure: The template for an adaptive priority algorithm

Inapproximations with respect to the priority model

Once we have a precise model, we can then argue that certain approximation bounds are not possible within this model. Such inapproximation results have been established with respect to priority algorithms for a number of problems but for some problems much better approximations can be established using extensions of the model.

1. For the weighted interval selection (a *packing problem*) with arbitrary weighted values (resp. for proportional weights $v_j = |f_j - s_j|$), no priority algorithm can achieve a constant approximation (respectively, better than a 3-approximation).
2. For the set cover problem, the natural greedy algorithm is essentially the best priority algorithm.
3. As I perhaps previously mentioned, for deterministic fixed order priority algorithms, there is an $\Omega(\log m / \log \log m)$ inapproximation bound for the makespan problem in the restricted machines model.

More on provable limitations of the priority model

The above mentioned inapproximations are with respect to deterministic priority algorithms. For an adaptive algorithm, the game between an algorithm and an adversary can conceptually be viewed as an alternating sequence of actionsr:

- The adversary eliminates some possible input items
- The algorithm makes a decision for the item with highest priority and chooses a new ordering for all possible remaining input items.

However, we note that for deterministic algorithms, since the adversary knows precisely what the algorithm will do in each iteation, it could initially set the input $\mathcal{I}$ once the algorithm is known.

More on provable limitations of the priority model

The above mentioned inapproximations are with respect to deterministic priority algorithms. For an adaptive algorithm, the game between an algorithm and an adversary can conceptually be viewed as an alternating sequence of actionsr:

- The adversary eliminates some possible input items
- The algorithm makes a decision for the item with highest priority and chooses a new ordering for all possible remaining input items.

However, we note that for deterministic algorithms, since the adversary knows precisely what the algorithm will do in each iteation, it could initially set the input $\mathcal{I}$ once the algorithm is known.

For randomized algorithms, there is a difference between an *oblivious adversary* that creates an initial subset $\mathcal{I}$ of items vs an *adaptive adversary* that is playing the game adaptively reacting to each decision by the algorithm.

Unless stated otherwise when discussing randomized algorithms we are assuming an oblivious adversary.

Extensions of the priority order model

In discussing more general online frameworks, we already implicitly suggested some extensions of the basic priority model (that is, the basic model where we have one-pass and one irrevocable decision). The following online or priority algorithm extensions can be made precise:

- Decisions can be *revocable* to some limited extent or at some cost. For example, we know that in the basic priority model we cannot achieve a constant approximation for weighted interval scheduling. However, if we are allowed to permanently discard previously accepted intervals (while always maintaining a feasible solution), then we can achieve a 4-approximation. (but provably not optimality).
- While the knapsack problem cannot be approximated to within any constant, we can achieve a 2-approximation by taking the maximum of 2 greedy algorithms. More generally we can consider some "small" number k of priority (or online) algorithms and take the best result amongst these k algorithms. The partial enumeration greedy algorithm for the makespan and knapsack problems are an example of this type of extension.

Extensions of the priority order model continued

- Closely related to the "best of k online" model is the concept of online algoitthms with "advice". (One could also study priority algorithms with advice but that has not been done to my knowledge.) There are two advice models, a model where one measures the maximum number of advice bits per input item, and a model where we are given some number ℓ of advice bits at the start of the computation. The latter model is what I will mean by "online with advice." Online with ℓ advice bits is equivalent to the max of $k = 2^\ell$ online model.

Extensions of the priority order model continued

- Closely related to the "best of k online" model is the concept of online algoitthms with "advice". (One could also study priority algorithms with advice but that has not been done to my knowledge.) There are two advice models, a model where one measures the maximum number of advice bits per input item, and a model where we are given some number ℓ of advice bits at the start of the computation. The latter model is what I will mean by "online with advice." Online with ℓ advice bits is equivalent to the max of $k = 2^\ell$ online model.
  NOTE: This model is a very permissive in that the advice bits can be a function of the entire input. Of course, in practice we want these advice bits to be "easily determined" (e.g., the number of input items, or the ratio of the largest to smallest weight/value) but in keeping with the information theoretic perspective of onine and priority algorithms, one doesn't impose any such restriction.
- There are more general parallel priority based models than "best of k" algorithms. Namely, parallel algorithms could be spawning or aborting threads (as in the pBT model to be discussed later).

Multipass algorithms

- Another model that provides improved results is to allow multiple passes (over the input items) rather than just one pass.
- This is not a well studied model but there are two relatively new noteworthy results that we will be discussing:
  1. There is deterministic 3/4 approximation for weighted Max-Sat that is achieved by two "online passes" (i.e., the input sequence is determined by an adversary) over the input sequence whereas there is evidence that no one pass deterministic online or priority algorithm can acheive this ratio.
  2. There is a $\frac{3}{5}$ approximation for bipartitie matching that is achieved by two online passes whereas no deterministic online or priority algorithm can do asymptotically better than a $\frac{1}{2}$ approximation.
- It is not clear how best to formalize these multi-pass algorithms. Why?

Multipass algorithms

- Another model that provides improved results is to allow multiple passes (over the input items) rather than just one pass.
- This is not a well studied model but there are two relatively new noteworthy results that we will be discussing:
  1. There is deterministic 3/4 approximation for weighted Max-Sat that is achieved by two "online passes" (i.e., the input sequence is determined by an adversary) over the input sequence whereas there is evidence that no one pass deterministic online or priority algorithm can acheive this ratio.
  2. There is a $\frac{3}{5}$ approximation for bipartitie matching that is achieved by two online passes whereas no deterministic online or priority algorithm can do asymptotically better than a $\frac{1}{2}$ approximation.
- It is not clear how best to formalize these multi-pass algorithms. Why? What information should we be allowed to convey between passes?