CSC2420: Algorithm Design, Analysis and Theory
Fall 2022
An introductory (i.e. foundational) level graduate course.

Allan Borodin

October 11, 2022

Week 4

I posted Calum's slides for last weeks class where he discussed online bipartite matching. Calum will be back on Wednesday, October 26 to continue this discussion and in particular to discuss online contention resolution schemes.

We ended the September 28 class with a brief introdution to the priority model for greedy and myopic algorithms. We will extend this model in various ways that continue to model myopic algorithms.

I will first repeat the priority algorithm definition and then continue from where we left off.

Today I will avoid proofs and mainly state a lot of results to gain an appreciaton of what can and can't be done by priority algorithms? I will add some references to papers in all the slides.

The fixed (order) priority algorithm template

```
$\mathcal{J}$ is the set of all possible input items
Decide on a total ordering $\pi$ 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

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 $\pi_i$ 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

Some comments on the priority model

- A special (but usual) case is that π is determined by a function $f : \mathcal{J} \to \mathbb{R}$ 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.
- In particular, we usually do not assume that the number n of input items is known in advance. If a priority (or online) algorithm needs know n is advance, then we should replace the While statement by a FOR statement to indicate that n is known.

More comments on the priority model

- Inapproximations (i.e., negative results) are made easier by assuming n is not known. However, for many problems we can establish an inpapproximation assuming n is known.
- **NOTE:** The model does not insist on the irrevocable decsions being "greedy" in whatever way we might interpret the term greedy. In some cases, the meaning of greedy is very natural.
- We may want priority and online algorithms to be given some (hopefully easily computed or known from past history) global information such as in the case of the makespan problem the minimum and maximium processing time (load) for any input item.
- We usually refer to "priority or online algorithms with advice" to be the study of the tradeoff between performance (i.e., the competitive/approxiamtion ratio) as the number of "advice bits" without knowing how the advice bits was obtained. That is, we stay within a purely information theoretic framework. It is assumed that this advice is "trusted information".

And some more comments on the priority model

- More recently, there has been substantial interest in online algorithms with ML advice. ML advice (or any advice which is not always accurate) introduces a different tradeoff, that of robustness (i.e. how much is performace hurt if the advice is inacurate) vs consistency (i.e. how much is performance improved when the advice is good).
- As far as I know, there hasn't been any work on ML advice for priority algorithms or online algorithms with random order arrivals.
- When we discuss randomized priority and online algorithms we mean that the irrevocable decision is a randomized decision and hence the cost or profit of an algorithm is a random variable. We compare the expected performance of the algorithm compared to that performance of an optimal solution which is not a random variable.
- Of course, an online algorithm in the random order model can be viewed as a randomized algorithm but we will reserve the term "randomized" in this context to mean the decision is randomized. Thet leaves us free to talk about deterministic or randomized algorithms in the random order model.

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 $\pi_i$ 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 for deterministic priority algorithms

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.

- 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).
- For the knapsack problem, no priority algorithm can achieve a constant approximation. We note that the maximum of two greedy algorithms (sort by value, sort by value/size) is a 2-approximation.
- As 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 machine model where m is the number of machines. .

The set cover problem

In the set cover problem, we are given a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ of sets with $S_i \subseteq U$ for some universe U. In the weighted set cover problem, each set S_i has a cost or weight $c(S_i)$. The objective is to find a minimum cost subcollection $\mathcal{S}'$ such that $\cup_{S \in \mathcal{S}'} = U$.

- For the set cover problem, "the natural adaptive greedy algorithm" is essentially the best priority algorithm. What is this algorithm?

The set cover problem

In the set cover problem, we are given a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ of sets with $S_i \subseteq U$ for some universe U. In the weighted set cover problem, each set S_i has a cost or weight $c(S_i)$. The objective is to find a minimum cost subcollection $\mathcal{S}'$ such that $\cup_{S \in \mathcal{S}'} = U$.

- For the set cover problem, "the natural adaptive greedy algorithm" is essentially the best priority algorithm. What is this algorithm?
- In each iteration i, the algorithm chooses a set S minimizing $\frac{c(S)}{n_i(S)}$ where $n_i(S) =$ the number of uncovered elements in S at the start of iteration i. This algorithm achieves a $\ln m$ approximation ratio where $m = \max_j S_j \leq |U|$.
- Under some reasonable comolexity hardness assumptions, this is the best approximation achieveable in polhynomial time. There is an NP-hardness result of $\Omega(\log m)$.

More on provable limitations of priority algorithms

- 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 naturally viewed an alternating sequence of actions;
  1. The adversary eliminates some possible input items
  2. 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.

And more on limitations of priority algorithms

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 we usually analyze randomized algorithms (for any type of algorithm) with respect to an oblivious adversary.

For adaptive adveraries there is an additional distinction that needs to be made. An adaptive online algorithm must also make irrevocable decisions for eacxh input item whereas an adaptive offline adversary waits until the entire sequence has been created.

Note that for adaptive adveraries the optimal performance is also a random variable annd the competitive ratio is the ratio of two expectations.

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 algorithms 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 (priority)" algorithms is the concept of online (priority) algoirthms with "advice". 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 (priority) with advice." Online with ℓ advice bits is equivalent to the max of $k = 2^\ell$ online (priority) model.

Extensions of the priority order model continued

- Closely related to the "best of k online (priority)" algorithms is the concept of online (priority) algoirthms with "advice". 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 (priority) with advice." Online with ℓ advice bits is equivalent to the max of $k = 2^\ell$ online (priority) model.

  NOTE: As previously stated, 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 online and priority algorithms, one doesn't impose any such restriction.

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 biparitie 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 biparitie 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?

End of Wednesday, October 12 class

We ended as I just intoduced the set packing problem and the s-set packing problem. I also gave a motivation for the problem, namely that it is the underlying allocation problem in combinatorial auctions.

Next class I will quickly state two natural greedy algorithms for the set packing problem and one somewhat surprising priority algorithm that will provide an improved approximation $\sqrt{m}$ ratio for set packing. A reference for the proof of the ratio can be found in the 2002 paper "Truth revelation in approximately efficient combinatorial algorithms" by Lehmann, O'Callaghan and Shoham. (See Theorem 7.2) There is a lot information about different types of CAs in this seminal paper.

Then I will conclude (for now) our discussion of greedy algorithms with the relation between optimal greedy algorithms and matroids.

I am leaving the remaining slides.

Greedy algorithms for the set packing problem

One of the new areas in theoretical computer science is algorithmic game theory and mechanism design and, in particular, auctions including what are known as *combinatorial auctions (CAs)* . The underlying combinatorial problem in CASs is the set packing problem.

The set packing problem

We are given n subsets $S_1, \ldots, S_n$ from a universe U of size m. In the weighted case, each subset S_i has a weight w_i. The goal is to choose a disjoint subcollection $\mathcal{S}$ of the subsets so as to maximize $\sum_{S_i \in \mathcal{S}} w_i$. In the s-set packing problem we have $|S_i| \leq s$ for all i.

- This is a well studied problem and by reduction from the max clique problem, there is an $m^{\frac{1}{2}-\epsilon}$ hardness of approximation assuming $NP \neq ZPP$. For s-set packing with constant $s \geq 3$, there is an $\Omega(s/\log s)$ hardness of approximation assuming $P \neq NP$.
- We will consider two "natural" greedy algorithms for the s-set packing problem and a non obvious greedy algorithm for the set packing problem. These greedy algorithms are all fixed order priority

The first natural greedy algorithm for set packing

Greedy-by-weight (*$Greedy_{wt}$*)

```
Sort the sets so that $w_1 \geq w_2 \ldots \geq w_n$.
$\mathcal{S} := \varnothing$
For $i : 1 \ldots n$
    If $S_I$ does not intersect any set in $\mathcal{S}$ then
        $\mathcal{S} := \mathcal{S} \cup S_i$.
End For
```

- In the unweighted case (i.e. $\forall i, w_i = 1$), this is an online algorithm.
- In the weighted (and hence also unweighted) case, greedy-by-weight provides an s-approximation for the s-set packing problem.
- The approximation bound can be shown by a charging argument where the weight of every set in an optimal solution is charged to the first set in the greedy solution with which it intersects.

The second natural greedy algorithm for set packing

Greedy-by-weight-per-size

```
Sort the sets so that $w_1/|S_1| \geq w_2/|S_2| \ldots \geq w_n/|S_n|$.
$\mathcal{S} := \varnothing$
For $i : 1 \ldots n$
    If $S_I$ does not intersect any set in $\mathcal{S}$ then
        $\mathcal{S} := \mathcal{S} \cup S_i$.
End For
```

- In the weighted case, greedy-by-weight provides an s-approximation for the s-set packing problem.
- For both greedy algorithms, the approximation ratio is tight; that is, there are examples where this is essentially the approximation. In particular, these algorithms only provide an m-approximation where $m = |U|$.
- We usually assume $n >> m$ and note that by just selecting the set of largest weight, we obtain an n-approximation. So the goal is to do better than $\min\{m, n\}$.

Improving the approximation for set packing

- In the unweighted case, greedy-by-weight-per-size can be restated as sorting so that $|S_1| \leq |S_2| \ldots \leq |S_n|$ and it can be shown to provide an $\sqrt{m}$-approximation for set packing.
- On the other hand, greedy-by-weight-per-size does not improve the m-approximation for weighted set packing.

Greedy-by-weight-per-squareroot-size

```
Sort the sets so that $w_1/\sqrt{|S_1|} \geq w_2/\sqrt{|S_2|} \ldots \geq w_n/\sqrt{|S_n|}$.
$\mathcal{S} := \varnothing$
For $i : 1 \ldots n$
    If $S_I$ does not intersect any set in $\mathcal{S}$ then
        $\mathcal{S} := \mathcal{S} \cup S_i$.
End For
```

Theorem: Greedy-by-weight-per-squareroot-size provides a $2\sqrt{m}$-approximation for the set packing problem. And as noted earlier, this is asymptotically the best possible approximation assuming $NP \neq ZPP$.

Another way to obtain an $O(\sqrt{m})$ approximation

There is another way to obtain the same aysmptototic improvement for the weighted set packing problem. Namely, we can use the idea of partial enumeration greedy; that is somehow combining some kind of brute force (or naive) approach with a greedy algorithm.

Partial Enumeration with Greedy-by-weight ($PGreedy_k$)

Let Max_k be the best solution possible when restricting solutions to those containing at most k sets. Let G be the solution obtained by $Greedy_{wt}$ applied to sets of cardinality at most $\sqrt{m/k}$. Set $PGreedy_k$ to be the best of Max_k and G.

- Theorem: $PGreedy_k$ achieves a $2\sqrt{m/k}$-approximation for the weighted set packing problem (on a universe of size m)
- In particular, for $k = 1$, we obtain a $2\sqrt{m}$ approximation and this can be improved by an arbitrary constant factor $\sqrt{k}$ at the cost of the brute force search for the best solution of cardinality k; that is, at the cost of say n^k.

Combinatorial auctions

For those who are interested, a combinatorial auction is one where n agents (buyers) want one of possibly many subsets of items $S_1, S_2, \ldots, S_k$. Each agengt has has a private value $w(S_i)$ for any desired set. An auctioneer wants to allocate at most on subset to each agent so as to maximize the total value of disjoint sets that are allocated. This is called the "social welfare" objective.

We assume "free disposal" (as in the display ads problem) so that $w(S) \leq w(T)$ if $S \subset T$.

If every agent was truthful about their valuations, this would be the set packing problem. But agents are self interested and may not truthfully report the values for the sets they desire if they think it will be helpful. That is, they could over state or understate their valuations.

In order to incentivize agents to be truthful, the auctioneer decides on prices for each agent.

Combinatorial auctions continued

If we could compute an optimal solution we would have a way to speficy a price for the subset to be allocated to each agent so that the agent would be truthful. But set packing is an NP hard approximation problem as we have indicated. This result (called VCG auctions) does not hold for approximation algorithms.

Perhaps the most prominent question in CAs, is how how much truthfulness can hurt the approximation ratio for a given CA if we insist upon efficient polynomial allocations or insist upon conceptually simple allocations and pricing.

Although we need to be more precise, truthful auctions using priority algorithm allocations for say an s-CA will result in asymptotically the worst possible approxiamtion (i.e., $\Omega(\min\{n.m\}$.

Max weighted independent set in a matroid.

We conclude by mentioning a classic result about greedy algorithms and matroids.

Let U be a set of elements and $\mathcal{I}$ be a collection of subsets of U. $(U, \mathcal{I})$ is a matroid if the following hold:

- (Hereditary property) If $I \in \mathcal{I}$ and $I' \subset I$, then $I' \in \mathcal{I}$.
- (Exchange property) If $I', I \in \mathcal{I}$ and $|I'| < |I|$, then $\exists u \in I \setminus I'$ such that $I' \cup \{u\} \in \mathcal{I}$.

An *hereditary set system* $(U, \mathcal{I})$ is any set system satisfying the hereditary property so that a matroid is an hereditary set system that also satisfies the exchange property.

The sets $I \in \mathcal{I}$ are referred to as the *independent sets*. We note that there are alternative equivalent definitions. In particular, an alternative to the exchange property is that every maximal independent set is called a *basis* and has the same size. This maximum size is call the *rank* of the matroid. There are alternative statement of the exchange property.

(Basis Exhange property) If $I \neq I^*$ are bases (i.e. maximimal independent sets in I), then $\forall x \in I^* \setminus I, \exists y \in I \setminus I^*$ such that $(I^* \setminus \{x\}) \cup \{y\}$ is also a

Matroids and the greedy algorithm

Any acyclic subset of edges in a graph is an independent set in a matroid. If the graph is connected then a maximal acyclic subset is a spanning tree. This is called a graphic matroid.

An independent set of vectors in a vector space is a matroid. ‘ Whitney

[1935] defibned this elegant abstraction that applies to many other systems. Generalizations to intersections of matroids and more general indepedence systems are also known.

The minimum spanning tree (MST) problem can be seen as a maximizaton problem (negate the signs of the edge weights or subtract each edge weight from the maxium edge weight).

Kruskal’s MST algorithm can be seen as a the natural greedy algorithm for computing a maximum weight independent set in a graphic matroid.

The Rado-Edmonds theorem

The proof of the optimality of Kruskal's algorithm can be carried over to show how the natural greedy algorithm obtains an optimal solution for computing an optimal independent set **in any matroid**.
What is this matural greedy algorithm for computing a max weight independent set in a matrod?

Remarkably, there is what can be seen as a converse to this fact.

Consider any hereditary set system. If the natural greedy algorithm produces an optimal solution for every linear function of elements in an independent set, then the set system is a greedy algorithm.

Does this define greedy algorithms?

The Rado-Edmonds theorem

The proof of the optimality of Kruskal's algorithm can be carried over to show how the natural greedy algorithm obtains an optimal solution for computing an optimal independent set **in any matroid**.
What is this matural greedy algorithm for computing a max weight independent set in a matrod?

Remarkably, there is what can be seen as a converse to this fact.

Consider any hereditary set system. If the natural greedy algorithm produces an optimal solution for every linear function of elements in an independent set, then the set system is a greedy algorithm.

Does this define greedy algorithms?

Is the optimality of greedy algorithms limited to independent sets in matroids?