

CSC2421: Online and Other Myopic Algorithms

Fall 2026

Allan Borodin

September 16, 2026

Breaking News:

Two major open problems have been solved

- The k -server conjecture is true. See <https://arxiv.org/abs/2609.15979>
- The matroid secretary conjecture is true. See <https://arxiv.org/abs/2609.14555>

This encouraged me to use the grading scheme on the next slide.

Then we will return to our introduction to Online and Other Myopic Algorithms

Grading Scheme for the Course

The grading for this course will be based on using one or more LLMs to try to solve any of the open problems in either Volume 1 or Volume 2. More specifically, everyone will be responsible for two projects, one individual and one a collaborative 2 or 3 person project. Your two person project should be different from your individual project. Besides ChatGPT and Claude, Google has a one year free subscription for students to its LLM (Antigravity/Gemini). For each project,

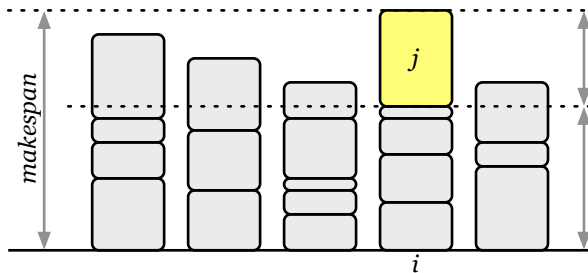
- 1 Choose 3-5, open problems and state how you will initially ask the LLM about the open problem you want to solve. Submit (probably on Markus) your choice of possible open problems and your initial queries by October 23 (before reading week).
- 2 Use the LLM(s) to decide on a specific project. Maintain a diary of the conversation you are having with the LLM.
- 3 Present whatever results have emerged from your interaction with the LLM(s) in a written report (along with the diary) and prepare a class presentation to be delivered in the last two weeks of the course. The written projects are due December 8; earlier submissions are welcome.

Greedy online algorithm for the identical machine makespan problem

- Let's start with this online greedy algorithm that dates back to Graham's 1966 and 1969 papers. Graham also considered an offline greedy algorithm called Longest Processing Time LPT in which the jobs are first sorted so $p_1 \geq p_2 \dots \geq p_n$ and then scheduled greedily.
- These are good starting points since (preceding NP-completeness) Graham conjectured that these are hard (requiring exponential time) problems to compute optimally but for which there were "good" worst case approximation ratios (although he didn't use that terminology).
- This might then be called the start of worst case approximation algorithms.
- There are some general concepts to be observed in Graham's work and (even after more than 50 years) there are still open questions concerning the initial makespan problem and the many variants of makespan problems.

The makespan problem for identical machines

- The input consists of n jobs $\mathcal{J} = J_1 \dots, J_n$ that are to be scheduled on m identical machines.
- Each job J_k is described by a **processing time** (or load) p_k .
- The goal is to minimize the latest finishing time (maximum load) over all machines.
- That is, the goal is a mapping $\sigma : \{1, \dots, n\} \rightarrow \{1, \dots, m\}$ that minimizes $\max_k \left(\sum_{\ell: \sigma(\ell)=k} p_\ell \right)$.



[picture taken from Jeff Erickson's lecture notes]

The Many Variants of Online Algorithms

As I indicated, Graham's algorithm could be viewed as the first example of what has become known as *competitive analysis* (as named in a paper by Manasse, McGeoch and Sleator) following the paper by Sleator and Tarjan which explicitly advocated for this type of analysis. Another early (pre Sleator and Tarjan) example of such analysis was Yao's analysis of online bin packing algorithms.

In competitive analysis we compare the performance of an online algorithm against that of an optimal solution. The meaning of *online algorithm* here is that input items arrive sequentially and the algorithm must make an irrevocable decision concerning each item. (For makespan, an item is a job and the decision is to choose a machine on which the item is scheduled.)

But what determines the order of input item arrivals?

The Many Variants of Online Algorithms continued

- In the “standard” meaning of online algorithms (for CS theory), we think of an adversary as creating a nemesis input set and the ordering of the input items in that set. So this is traditional worst case analysis as in approximation algorithms applied to online algorithms. If not otherwise stated, we will assume this adversarial order as the standard meaning for online algorithms. .
- We will also sometimes consider an *online stochastic model* where an adversary defines an input distribution and then input items are sequentially generated. There can be more general stochastic models (e.g., a Markov process) but the i.d. and i.i.d models are common in analysis. *Stochastic analysis* is well studied in OR.
- In the i.d. and i.i.d models, we can assume that the distributions are *known* by the algorithm or *unknown*.
- In the *random order model* (ROM) as we introduced in the secretary problem, an adversary creates a size n nemesis input set and then the items from that set are given in a uniform random order (i.e. uniform over the $n!$ permutations)

More general online frameworks

In the standard online model (and the variants we just mentioned), we are considering a one pass algorithm that makes one irrevocable decision for each input item.

There are many extensions of this one pass paradigm. For example:

- An algorithm is allowed some limited ability to revoke previous decisions.
- There may be some forms of lookahead (e.g. buffering of inputs).
- The algorithm may maintain a “small” number of solutions and then (say) take the best of the final solutions.
- The algorithm may do several passes over the input items.
- The algorithm may be given (in advance) some *advice bits* or a prediction based on the entire input.

Throughout our discussion of algorithms, we can consider deterministic or randomized algorithms. In the online models, the randomization is in terms of the decisions being made. course, the ROM model is an example of where the ordering of the inputs is randomized.)

Other measures of performance

The above variants address the issues of alternative input models, and relaxed versions of the online paradigm.

Competitive analysis is really just asymptotic approximation ratio analysis applied to online algorithms. Given the number of papers devoted to online competitive analysis, it is the standard measure of performance.

However, it has long been recognized that as a measure of performance, competitive analysis is often at odds with what seems to be observable in practice. Therefore, many alternative measures have been proposed. An overview of a more systematic study of alternative measures (as well as relaxed versions of the online paradigm) for online algorithms is provided in Kim Larsen's lecture slides that I have placed on the course web site.

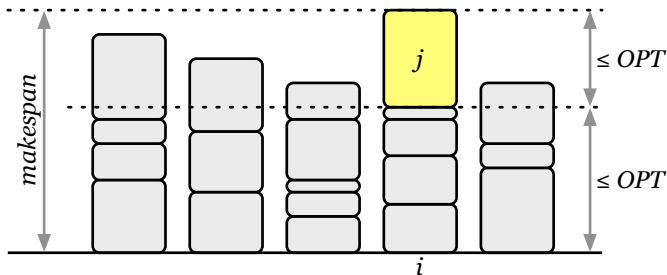
See, for example, the discussion of the *accommodating function* measure (for the dual bin packing problem) and the *relative worst order* measure for the bin packing coloring problem.

Returning to Graham's online greedy algorithm

Consider input jobs in **any order** (e.g. as they arrive in an *online* setting) and schedule each job J_j on any machine having the least load thus far.

- We will see that the **approximation ratio** for this algorithm is $2 - \frac{1}{m}$; that is, for any set of jobs $\mathcal{J}$, $C_{\text{Greedy}}(\mathcal{J}) \leq (2 - \frac{1}{m})C_{\text{OPT}}(\mathcal{J})$.
 - ▶ C_A denotes the cost (or makespan) of a schedule A .
 - ▶ OPT stands for any optimum schedule.
- **Basic proof idea:** $\text{OPT} \geq (\sum_j p_j)/m$; $\text{OPT} \geq \max_j p_j$

What is C_{Greedy} in terms of these requirements for any schedule?



[picture taken from Jeff Erickson's lecture notes]

Graham's online greedy algorithm

Consider input jobs in any order (e.g. as they arrive in an online setting) and schedule each job J_j on any machine having the least load thus far.

- In the online “competitive analysis” literature the ratio $\frac{C_A}{C_{OPT}}$ is called the **competitive ratio** and it allows for this ratio to just hold in the limit as C_{OPT} increases. This is the analogy of *asymptotic approximation ratios*.

NOTE: Often, we will not provide proofs in the lecture notes but rather will do or sketch proofs in class (or leave a proof as an exercise).

- The approximation ratio for the online greedy is “tight” in that there is a sequence of jobs forcing this ratio.
- The negative result (i.e. there is “bad” input sequence for the algorithm) is not an asymptotic result. **Is there a “meaningful” asymptotic result?**
- This bad input sequence suggests a better algorithm, namely the LPT (offline or sometimes called semi-online) greedy algorithm.

Graham's LPT algorithm

Sort the jobs so that $p_1 \geq p_2 \dots \geq p_n$ and then greedily schedule jobs on the least loaded machine.

- The (tight) approximation ratio of LPT is $(\frac{4}{3} - \frac{1}{3m})$.
- It is believed that this is the **best** “greedy” algorithm but how would one prove such a result? This of course raises the question as to **what is a greedy algorithm**.
- We will present the **priority model** for greedy (and greedy-like) algorithms. I claim that all the algorithms mentioned on the next slide can be formulated within the priority model.
- Assuming we maintain a priority queue for the least loaded machine,
 - ▶ the online greedy algorithm would have time complexity $O(n \log m)$ which is $(n \log n)$ since we can assume $n \geq m$.
 - ▶ the LPT algorithm would also have time complexity $O(n \log n)$.

Greedy algorithms in CSC373

Some of the greedy algorithms we study in different offerings of CSC 373

- The optimal algorithm for the fractional knapsack problem and the approximate algorithm for the proportional profit knapsack problem.
- The optimal unit profit interval scheduling algorithm and 3-approximation algorithm for proportional profit interval scheduling.
- The 2-approximate algorithm for the unweighted job interval scheduling problem and similar approximation for unweighted throughput maximization.
- The Kruskal and Prim optimal algorithms for minimum spanning tree.
- Huffman's algorithm for optimal prefix codes.
- Graham's online and LPT approximation algorithms for makespan minimization on identical machines.
- The 2-approximation for unweighted vertex cover via maximal matching.
- The “natural greedy” $\ln(m)$ approximation algorithm for set cover.

The fixed (order) priority algorithm template

$\mathcal{I}$ is the set of all possible input items
Decide on a total ordering π of $\mathcal{I}$
Let $\mathcal{I} \subset \mathcal{I}$ be the input instance
 $S := \emptyset$
 % S is the set of items already seen
 $i := 0$ % $i = |S|$
while $\mathcal{I} \setminus S \neq \emptyset$ **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 fixed order priority model

- A special (but usual) case is that π is determined by a function $f : \mathcal{J} \rightarrow \mathbb{R}$ 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.
- 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 maximum 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

$S := \emptyset$ % S is the set of items already considered

$i := 0$ % $i = |S|$

while $\mathcal{I} \setminus S \neq \emptyset$ **do**

$i := i + 1$

 decide on a total ordering π_i of $\mathcal{J}$

$\mathcal{I} := \mathcal{I} \setminus S$

$l_i := \min_{\leq \pi_i} \{I \in \mathcal{I}\}$

 make an irrevocable decision D_i concerning l_i

$S := S \cup \{l_i\}$

$\mathcal{J} := \mathcal{J} \setminus \{I : I \leq_{\pi_i} l_i\}$

 % some items cannot be in input set

end

Figure: The template for an adaptive priority algorithm