COMP 7720 - Online Algorithms

Advice for Giving a Talk

Shahin Kamali

Notes for Presentations - Nov. 29th, 2017
University of Manitoba
You will need presentations in your future career, inside or outside academia.

Always think of a presentation as an **opportunity to advertise** your work.

- It is a chance to make others interested in what you do.
Do not make your presentation too long.

- If you are given $x$ minutes, plan for talking $0.7x$ to $0.8x$ minutes
  - This will give chance for questions and transition
  - And also helps you not to rush or worry about time
- Remember: the best (or worst?) way to annoy your audience is to go overtime.
  - This is universal, for course projects, research projects, job talks, etc.
Style

Do not try to memorize what you are going to say.

- Memorizing adds pressure to you when talking and makes the presentation artificial.
- Think of it a presentation in a similar way as meeting with your professor; you prepare but you do not memorize.

- Instead, present the material a few times for yourself, each time possibly with different words and sentence.
  - This also helps you be more authentic.
Make a bridge

- Try to connect with your audience. Do not be afraid of them; they are not dangerous in most cases :)
  - Have eye-contact; look at their eyes.
  - If you have time, make the presentation a bit interactive

- Be authentic, i.e., be the same person that you are when you are in more comfortable situations

- At the start of a presentation, it often helps to ‘break the ice’ with talking about an interesting topic which is not necessarily related to your talk.

- Show you sense of humour; but be careful about norms and limits.
Language

- If you are not confident about your language skills:
  - First, do not panic.
  - Second, note that in most Computer Science talks, a good ratio of your audience speak English as their second or third language (this ratio is 100% in this course).
  - So, no one will judge you based on your language skills (in particular the instructor of this course).

- Try to speak slowly (something that I sometimes forget myself)
  - It is not bad to have an accent (you can be even proud of it) as long as your accent is easy to follow and understand.
  - Speaking slowly solves most accent problems.

- Once again, the language issue should not make you go to the path of memorizing
  - Try to say a sentence of your talk in a few different ways; this helps in having one come to your mind in the actual presentation.
Enthusiasm

- Show your enthusiasm for the topic you present
- **Remember: your audience are not there to see you reading the slides loudly; they can do it themselves**
- Often you say more than what is on the slides (depending on the topic, it can be an explanation, an intuition, an example, etc.).
Do not make your slides overwhelming

Include a minimum required material

- Be succinct: remove anything that can be removed

Try to arrange your slides so that they have a good transition

- It often helps that the last sentence for explaining one slide be related to the next slide.

Avoid distracting background colors, font, or any thing that does not contribute to the material your present.
Example

In the next few slides, you see an example of a 5-minute presentation

- You do not need to follow its format and flow (use it just as a sample)
- But it is important to pay attention to the tips mentioned in the previous slides
- In general, in your presentations, focus on presenting the problem, your assumptions, and initial results if you have.
Online knapsack problem

Homer Simpson & Peter Griffin
In the offline setting, knapsack asks for placing a set of items of different values into a sack of uniform capacity 1.

- Each item has a size \( \leq 1 \) and a value.
- The total size of items should be at most 1.
- The goal is to maximize the total value of items in the sack.
- It is a well-known NP-hard problem.
Online knapsack

- In the online setting, a sequence of items arrives in an online manner
  - For each item, an algorithm should decide whether to put it in sack or not
  - The decisions of an online algorithm are irrevocable
- We use competitive analysis to compare online knapsack algorithms
Model & Assumptions

When defining our problem, we note that adversary can build inputs for which the competitive ratio of any algorithm is unbounded.

- Initially, a large number of small items are presented.
- If the algorithm places a small item of size $\epsilon$ in the sack, a big item $H$ of size more than $1 - \epsilon$ is next revealed.
  - Assuming value of $H$ is unbounded, the competitive ratio becomes unbounded.
- If the algorithm never places a small item of size $\epsilon$ in the sack, adversary keeps requesting them.
  - At some point, the benefit of $OPT$ by accepting all small items results in unbounded competitive ratio.

To solve this issue, we assume the sizes are bounded by a value $d$. 
Problem Definition

- The input to the restricted knapsack problem is a sequence of $n$ items, each having a size of at most 1 and value at most $d$.
- The values of $n$ and $d$ are known to the online algorithm.
- The goal is to maximize the total value when accepting/rejecting items in an online manner.
In this project, we investigate to see whether there is a constant competitive ratio for this problem.

- The competitive ratio would be a function of $d$.
- We investigate lower bounds that might depend on $n$ (if such a lower bound exists, no algorithm is competitive).

Consider other variants of the problem:

- when an algorithm can ‘remove some of the accepted items at a penalty’

We consider the advice complexity of the problem:

- Particularly, we are interested in advice of constant size.
Current contribution

- Until now, we have achieved the following results:
  - No algorithm can be better than $d$-competitive (you might include the proof, in another slide, if there is enough time, i.e., other parts of your presentation are short)
  - To achieve an optimal solution, $O(n)$ bits of advice are sufficient and required
    - Upper bound: just encode whether an item should be accepted or not
    - Lower bound: A reduction from binary guessing (Again, you might include the idea behind those reductions if you have time, in another slide).
  - Advice of sublinear helps us achieve competitive better competitive ratios.