15-451/651 Algorithm Design & Analysis
Fall 2022, Recitation #10

Objectives

- Understand commonly applied techniques for constructing and analyzing approximation algorithms.
- Understand the objective of an online algorithm, how to create one, and how to bound its competitive ratio.

Recitation Problems

1. **(Scheduling Jobs but with 1 Machine)** Given a single machine, we are to process n jobs each with associated release times r_i and process times p_i. The ith job cannot be started before it's release time r_i, and it takes p_i to finish after it's been released. Let T_i denote the finish time of job i, so that $T_i \geq r_i + p_i$. Our objective is to minimize $\sum_i T_i$.

   Consider the following relaxation of this problem: a preemptive schedule is a schedule that allows us to stop a job in the middle of processing and resume the job from where we left off at some later time. We can compute the optimal preemptive schedule in polynomial time[1]. Let the finish times of job i in the optimal preemptive schedule be T'_i.

   (a) Prove that scheduling with preemptive schedules is indeed a relaxation of scheduling with non-preemptive schedules. In other words, show that any valid non-preemptive schedule is also a valid preemptive schedule. What is the relationship between $\sum_i T'_i$ and $OPT = \sum_i T_i$?
   (Hint: For the second part, your answer should be one of $=, \leq, \geq$.)

   (b) Let job j be the j^{th} job that the preemptive schedule finishes. Prove the following:

   $$T'_j \geq \max_{k=1}^{j} r_k \text{ and } T'_j \geq \sum_{k=1}^{j} p_k$$

[1] In fact, you can do it greedily. At any time, process the job with shortest remaining processing time. This is called the SRPT rule. Try to show why this is optimal.

(c) Suppose that our non-preemptive schedule finishes jobs in the exact same order as our preemptive schedule. Show that $T_j \leq 2T'_j$. Conclude that this non-preemptive schedule is a 2-approx of OPT.

2. **(SONET Ring Loading)** The following problem is a classical problem in telecommunications networks. We have a cycle with n vertices, numbered 0 through $n-1$ clockwise around the cycle. We are also given a set of requests. Each request is a pair (i, j) where i is the source vertex and j the target vertex. The call can be routed either clockwise or counterclockwise through the cycle. The objective is to route the calls so as to minimize the load (total number of uses) of the most loaded edge of the cycle.

   Write an linear program relaxation for the problem, and use it to give a 2-approximation algorithm using a rounding argument. Remember that a linear program relaxation is an LP such that if you could force some variables to be integers, you would solve the problem exactly.

3. **(Online Algorithms: Splay-Coin)** 15-451 is creating a new cryptocurrency: the splay-coin. To garner interest in the coin, we are allowing each coin to be exchanged for 1 bonus point to your final grade.

   Naturally, you would like to acquire 100 splay-coins by the last day of the n day semester, since you did not turn in any homework. On day i you check the price p_i of splay-coins, and decide whether to purchase. Unfortunately, due to a bug in the system, you can only purchase once, and want to minimize the cost when you do so.

   Since cryptocurrency values are often highly volatile, 15-451 has also implemented some price controls on our coin, including a lower bound L and upper bound U on the price p_i at any day i.

   (a) Being the clever 15-451 student that you are, you'd like to use your algorithms knowledge to help you minimize the price you pay. Come up with a $\sqrt{U/L}$-competitive algorithm for this problem.

   (b) Now just to make sure that none of your classmates can best you, prove that no deterministic algorithm can achieve a competitive ratio better than $\sqrt{U/L}$.

Further Review

1. **(Short answer / multiple choice)**
   (a) Consider the job scheduling problem with the following jobs to be scheduled on 3 machines: $[4, 10, 3, 6, 9, 10, 20]$.
      i. What is the makespan returned by the greedy algorithm?
      ii. What is the makespan returned by the sorted greedy algorithm?
      iii. What is the optimal makespan for these 3 machines?

   (b) Consider the list update problem with the following sequence of Access(x) operations on an array of length 4 (1-indexed): $[2, 1, 4, 3]$
      i. What is the cost of these accesses using the move-to-front algorithm?
      ii. What is the optimal cost of these accesses?

2. **(Weighted Scheduling Jobs on 1 Machine)** Now consider the following weighted variant of the first problem: each job has an associated weight w_i and our objective is to minimize $\sum_i w_i T_i$.

   Unfortunately, it is now NP-Hard to compute the optimal weighted preemptive schedule. Instead we'll tackle this problem via the following LP (where N is the set of jobs):

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{n} w_i T_i \\ \text{s.t. } T_i & \geq r_i + p_i \qquad \forall j \in N \\ \sum_{i \in S} p_i T_i & \geq \frac{1}{2}(\sum_{i \in S} p_i)^2 \qquad \forall S \subseteq N \end{aligned}$$

   (Hint: The variables here are T_i's and everything else is a constant.)

   (a) Prove that this is a relaxation.

   (b) Given the optimal solution T^*, reorder the jobs s.t. $T_1^* \leq T_2^* \leq \dots$. Show that if we set our schedule to finish jobs in this order, we have a 3-approx of OPT.

3. **(Approximating Vertex Cover via the Dual)** In class, we have seen an 2-approximation of Vertex Cover by rounding the LP relaxation of Vertex Cover. Now, we'll solve the same problem but instead, we'll take the dual of the LP relaxation and extract a vertex cover from the solution of the dual. Recall the LP for Vertex Cover:

$$\begin{aligned}\text{minimize} \quad & \sum_{v \in V} x_v \\ \text{s.t.} \quad & x_u + x_v \geq 1 \qquad \forall \{u,v\} \in E \\ & x_v \geq 0 \qquad \forall v \in V\end{aligned}$$

And its dual, the LP for maximum matching from Recitation 8:

$$\begin{aligned}\text{maximize} \quad & \sum_{e \in E} y_e \\ \text{s.t.} \quad & \sum_{e : v \in e} y_e \leq 1 \qquad \forall v \in V \\ & y_v \geq 0 \qquad \forall e \in E\end{aligned}$$

(a) Now, given the optimal solution y^*, let our vertex cover S be the set of vertices in which the constraints on those edges are tight (i.e. we achieve equality on those constraints). Prove that S is a vertex cover.

(b) Show that $|S| \leq 2 \cdot OPT$

4. **(Planar Vertex Cover (Hard/optional))** In class we saw a 2-approximation for the vertex cover problem by rounding its LP relaxation. We first solved the following LP with variables x_v for each $v \in V$, with the idea that, in an integer solution, $x_v = 1$ if v is in the vertex cover, and $x_v = 0$ otherwise.

$$\begin{aligned}\text{minimize} \quad & \sum_{v \in V} w_v \\ \text{s.t.} \quad & w_u + w_v \geq 1 \qquad \forall \{u,v\} \in E \\ & w_v \geq 0 \qquad \forall v \in V\end{aligned}$$

When we solve the LP, we can get fractional values of x_v, so we *round* them by taking any vertices v for which $x_v \geq 1/2$. In this problem, we want to show that we can do better on planar graphs. First, we will need an interesting lemma about the solutions to this LP relaxation:

(a) **(Optional – feel free to skip and just do the next part)** Prove that for any vertex solution to the LP relaxation of vertex cover above, for any vertex v, the value of x_v is always in $\{0, \frac{1}{2}, 1\}$. This property is often called "half-integrality", because the vertex solutions to the LP are always either integral, or have variables equal to $\frac{1}{2}$, but nothing else.

Hints: Take a look at the optional material in the lecture notes for linear programming, where a very similar proof is given for the bipartite matching problem, which shows that its vertex solutions are always integral.

Now that we have this useful lemma, we are going to use another even more powerful result, arguably the most famous theorem in all of graph theory, the *four color theorem*, which says that *all planar graphs are four colorable.* Remember that a proper coloring of a graph is an assignment of a color to each vertex such that adjacent vertices are never the same color. A four coloring means that at most four different colors are needed. You may also the fact that such a coloring can be found in polynomial time.

(b) Combine this fact with the lemma from (a), and the fact that our favourite LP solving algorithms return a vertex solution, to produce a 1.5-approximation algorithm for vertex cover on planar graphs.

 Hints: The standard rounding algorithm rounds up *every* vertex where $x_v = 1/2$, but this can be a little redundant if two adjacent vertices both have $x_v = 1/2$. Find a way to skip rounding some of the vertices, such that you are still guaranteed to have a valid vertex cover.