15-251 : Great Theoretical Ideas In Computer Science

Fall 2013

Assignment 8: Bad Movies

Due: Friday, Nov. 1, 2013 11:59 PM

Name: _____

Andrew ID: _____

Question:	1	2	3	4	5	Total
Points:	30	25	25	20	10	110
Score:						

1. Choosing a Movie We All Like

(15) (a) It's 251 movie night! The course staff and some of their friends have decided to get together in Rashid Auditorium and watch a bad movie. Unfortunately, they didn't decide in advance which movie they would watch, which resulted in lots of arguing once everyone arrived. Eventually, they decided to use a voting system that they called CleverVoting: each person would rank all k movies in order of his/her own preference, and each movie would get α_1 points for each first-place vote, α_2 points for each second-place vote, and so on, such that $\alpha_1 > \alpha_2 > ... > \alpha_k$ (all fixed nonnegative real numbers). Note that Borda count is a special case of CleverVoting with $\alpha_1 = k-1, \alpha_2 = k-2, \ldots, \alpha_k = 0$, and Plurality is a special case with $\alpha_1 = 1$ and $\alpha_j = 0$ for $2 \leq j \leq k$.

But then they kept arguing about what the αs should be! As it turned out, though, it didn't matter. Someone proved that the voting system wouldn't be Condorcet consistent, no matter what α values are chosen.

Give a single preference profile that would show Condorcet inconsistency for CleverVoting for any combination of α values. Prove your answer.

Solution:

(15) (b) Eventually, the course staff and their friends threw out the idea of using CleverVoting. Since all of their movie choices were awful, they decided that the best way to choose the movie would be to have everyone rank their preferences among all movies and then choose the movie with the fewest last-place votes. Additionally, each voter has a fixed weight $\omega \in \mathbb{Z}^+$ (a positive integer). The vote of a person with weight ω counts as ω identical votes in a normal system.

k of the course staff's friends only want to watch Sharknado, and they don't care how badly they have to lie to get it. They've managed to determine the preferences of the rest of the movie-watchers and everyone's weights, and have started a GroupMe to coordinate manipulating the vote. The voter weights are fixed and cannot be manipulated.

The LEAST-BAD-MOVIE-MANIPULATION (LBMM) problem asks: "Can these k movie-watchers coordinate to cast their votes such that the voting system uniquely chooses Sharknado as the winner, given that they know everyone else's preferences and everyone's weights?" Prove that LBMM is NP-Hard. [Hint: Show a reduction from the PARTITION problem from Recitation 8.]

Solution:

2. Birdemic: Shock and Terror

One of the problems faced by inhabitants of the *Birdemic: Shock and Terror* universe is the problem of cleaning up after the exploding birds hit the ground. Unfortunately, due to lots of death thanks to those acid-spitting, exploding birds, only two people are able and willing to clean up after the explosions.

Assume that all the exploding birds fall along a straight line. Our two intrepid clean-up workers begin located at positions $w_1 = w_2 = 0$ on this line, and whenever the i^{th} bird explodes (at a real-numbered point x_i on the line), at least one of them must move to x_i to clean up after the explosion. The clean-up workers are trying to minimize the amount of work they have to do in total, where work is measured by the total distance that the two workers have to move.

(10) (a) The clean-up workers initially use a greedy algorithm: whichever of them is the closest to the i^{th} explosion moves to x_i to clean it up. They break ties arbitrarily. Show that the worst case competitive ratio of this algorithm is not bounded from above by any constant.

Solution:

.

Consider a new algorithm, called "CrazyAlgo": Let z_1 and z_2 represent the positions of the two clean-up workers in this algorithm. Here, if $x_i \notin [z_1, z_2]$, the closer one moves to the explosion, but if $x_i \in [z_1, z_2]$, let $d = \min\{|z_1 - x_i|, |z_2 - x_i|\}$ (the minimum distance from the explosion to one of the clean-up workers), and then both workers move a distance of d in the direction of the explosion. Isn't the algorithm really crazy, doing all the unnecessary movements?

You will show that CrazyAlgo is a 2-approximation of the optimal algorithm.

(10) (b) Let OPT be the optimal *offline* algorithm assuming that the workers know in advance where all of the explosions will occur. Let a_1 and a_2 denote the positions of the clean-up workers under OPT at any point of time. Initially, we have $a_1 = w_1 = a_2 = w_2 = 0$. Define $\Phi = 2 \cdot |z_1 - a_1| + 2 \cdot |z_2 - a_2| + |z_1 - z_2|$. Thus, initially, $\Phi = 0$.

Now, on each explosion, we first let OPT move the clean-up workers to change a_1 and a_2, and analyze the change in Φ. Then, we let CrazyAlgo make its move by changing z_1 and z_2 , and analyze the change in Φ. The change, denoted $\Delta\Phi$, is defined as the value of Φ after the move minus the value of Φ before the move.

Show that when OPT makes a move with cost k, $\Delta\Phi \leq 2 \cdot k$. Show that when CrazyAlgo makes a move with cost k, $\Delta\Phi \leq -k$.

Solution:

(5) (c) Use this to complete the argument that CrazyAlgo is a 2-approximation of OPT.

Solution:

3. Troll 2

Consider a village with food caches at n locations around town. Unfortunately, those food caches have been contaminated with goblin potion made from the Stonehenge Magic Stone. If anyone eats the food from those caches, they will be turned into a plant/human hybrid and eaten by vegetarian goblins! Knowing the locations of all the caches and how long it will take you to get from one cache to any other cache (note that this may not conform to Euclidean distances), you must figure out the shortest way to visit all the caches (so you can destroy the food) exactly once (so you don't waste time) and come back to where you started. Sadly, this problem is NP-hard, so you're going to approximate it. Assume that you can go from any cache to any other cache in finite time.

(10) (a) You construct the following algorithm: begin at an arbitrary cache. Travel to the unvisited cache that you can get to in the least time. Repeat until all caches have been destroyed. Return to the first cache.

Prove that this heuristic is not a c-approximation for any constant c.

Solution:

(15) (b) Show that if $f(n)$ is any function that can be computed in time polynomial in n, then the cache-destruction problem cannot be approximated within a factor of $f(n)$ unless P=NP.

(Hint: You've seen in class that this problem is NP-complete. Stare at that reduction really hard. Try to extend that reduction; the *gap* between *the good solutions* and *the bad solutions* should be large, so that even approximation algorithms for our problem should solve the original NP-complete problem that we reduced to our problem!)

Solution:

4. The Room

Place yourself in the shoes of the famous Tommy Wiseau. Imagine that you want to make a movie. Unfortunately, a tornado full of sharks has torn through your film room, and you're close to a deadline. You had already put together n equal-length clips into a single film, but now film clips are strewn everywhere on the floor. More specifically: you had m bundles of clips, each containing a single copy of each clip. Now, though, some of the clips are missing from some of the bundles. You can analyze which clips each bundle has left, but have time to grab k of the bundles to try to reassemble as much of the full movie as possible.

The k-BIGGEST-MOVIE problem consists of taking k bundles where $k \leq m$, and using all the clips in the chosen bundles to try to reconstruct as much of the original movie as possible. That is, select k bundles such that as many different clips are contained in at least one bundle as possible.

(20) (a) Give a natural greedy approximation for k-BIGGEST-MOVIE, and show that its approximation factor is at least $1 - 1/e$. [Hint: Use the Pigeonhole principle, and that $(1 - 1/k)^k \leq 1/e$ for all $k \in \mathbb{N}$.]

Solution:

5. Bonus: Baby Geniuses 2

(10) (a) Need a polynomial time 3-SAT solver? Just ask baby geniuses! Sadly, babies speak babytalk, which is completely unintelligible to anyone anywhere for some reason, so you can only interpret what they say to form a weaker method.

Here, you're concerned with a slightly restricted version of 3-SAT, where each variable appears at most once in each clause. It is known that even this restricted version is NP-complete! Construct an algorithm for this version of 3-SAT which, on every input instance, returns an assignment that satisfies at least $\frac{7}{8}$ fraction of the maximum number of clauses of that instance that could be satisfied by any assignment, and whose expected running time is polynomial of the input size on every input.

Hint: First construct an algorithm that runs in polynomial time, but returns an assignment that only satisfies the required number of clauses in expectation*. Then, modify the algorithm so that the assignment returned always satisfies the required number of clauses, but the running time is now polynomial only* in expectation*!*

Solution: