CSC 440

Assignment 7: P, NP, and Approximation

Due Tuesday, April 28th by 11:59PM

You may work in small groups on this assignment, but the work you hand in must be your own. You may submit this in one of two ways:

- Write it up electronically (e.g. LaTeX) and upload a PDF to Gradescope.
- Scan your handwritten solution (you can use a scanning app for a smartphone, such as Evernote's free Scannable app) and upload a PDF to Gradescope.

Your full name: Ygory Galva Liziano

1. Coffeeshops (20 pts):

401 Cafe has decided to expand to maximize profits over all of South County, by building new 401 Cafes on many street corners. The management turns to URI's Computer Science majors for help. The students come up with a representation of the problem:

The street network is described as an arbitrary undirected graph $G = (V, E)$, where the potential restaurant sites are the vertices of the graph. Each vertex u has a nonnegative integer value p_u, which describes the potential profit of site u. Two restaurants cannot be built on adjacent vertices (to avoid self-competition). The students further come up with an exponential-time algorithm that chooses a set $U \subseteq V$, but it runs in $O(2^V|E|)$ time. The 401 Cafe ownership is dismayed, and is convinced they should have hired UConn's CS majors instead. In order to prove that nobody else could have come up with a better algorithm, you need to prove the hardness of the COFFEE problem.

Define COFFEE to be the following decision problem: given an undirected graph $G = (V, E)$, given a mapping p from vertices $u \in V$ to nonnegative integer profits $p(u)$, and given a nonnegative integer k, decide whether there is a subset $U \subseteq V$ such that no two vertices in U are neighbors in G, and such that $\sum\limits_{u \in U} p(u) \geq k$.

Prove that COFFEE is NP-hard. (Hint: Try a reduction from 3SAT.)

Also, explain why this implies that, if there is a polynomial-time algorithm to solve the original problem, i.e., to output a subset U that maximizes the total profit, then $P = NP$.

Let $\Phi = C_1 \wedge C_2 \wedge \dots C_m$ be the input formula to a 3SAT Problem, where each clause C_c has three literals choosen from $\{x_i, \bar{x}_i \mid 1 \le i \le n\}$
construct a Coffeesshops Problem (G, P, K) as follows.
The vertices V of G are $\{v_{c,j} \mid 1 \le c \le m,\ 1 \le j \le 3\}$, where $v_{c,j}$ corresponds to literal j in clause C_c. we label vertex $v_{c,j}$ with x_i or $\bar{x}_i$, whichever appears in position j of clause C_c.
The Edges E of G are of two types:

- For each clause C_c, an edge between each pair of vertices corresponding to literals in clause C_c, that is, between v_{c,j_1} and v_{c,j_2} for $j_1 \ne j_2$
- for each i, an edge between each pair of vertices for which one is labeled by x_i, and the other by $\bar{x}_i$

The threshold K is equal to m.
The function P maps all vertices to 1. The
we claim that Φ is satisfiable iff the total Profit in the Coffeshops Problem (G, P, K) can be at least K. First, suppose that Φ is satisfiable. Then there is some thuth assingment A mapping to variables to $\{true, False\}$
A must take at least one literal per clause true; for each clause, select the vertex corresponding to one such literal to be in the set U. Since there are m clauses, this yields exactly $m = K$ vertices, so the total Profit is K. Moreover, we claim that U cannot contain two neighboring vertices in G. suppose for contradiction that u, $v \in U$ and $(u, v) \in E$. Then the edge (u, v) must be of one of the two types above. But u and v cannot correspond to literals in the same clause because we selected only one vertex for each clause. And u and v cannot be labeled by x_i and $\bar{x}_i$ for the same i, because A cannot make both a variable and its negation true. Since neither possibility can hold, U cannot contain two neighboring vertices. U achieves a total Profit of K for the Coffeeshops problem (G, P, K).

Continuation of #1

Conversely, suppose that there exists $\mathcal{U} \subseteq V$, $|\mathcal{U}| \geq K = m$ containing no two neighbors in G. Since U does not contain neighbors, it cannot contain two vertices from the same clause. Therefore, we must have $|U| = m$, with exactly one vertex from each clause. Now define a truth assignment A for the variables: $A(x_i) =$ true if some vertex with label x_i is in $\mathcal{U}$, and $A(x_i) =$ false if some vertex with label $\bar{x}_i$ is in $\mathcal{U}$. For other variables the truth value can be

Additional Space as needed

arbitrary. Since $\mathcal{U}$ does not contain neighbors, $\mathcal{U}$ cannot contain two vertices with contradictory labels, so Assignment A is well-defined. A satisfies all clauses by making one literal correspondin to a vertex in $\mathcal{U}$ true in each clause. Therefore, A satisfies Φ.

For the last question, suppose that there is a polynomial-time alg to solve the original problem, i.e, to output a subset $\mathcal{U}$ that maximizes the total profit. Then this alg can be easily adapted to a polynomial-time alg for coffeshops: for any $\langle G, P, K \rangle$, simply run the assummed alg and obtain an optima subset $\mathcal{U}$. Then output true if $K \leq |\mathcal{U}|$ and false otherwise. Since I have already shown that coffeshops is NP-hard, this implies that $P = NP$

2. Ghostbusters and Ghosts (20 pts):

A group of n Ghostbusters is battling n ghosts. Each Ghostbuster carries a proton pack, which shoots a stream at a ghost, eradicating it. A stream goes in a straight line and terminates when it hits the ghost. The Ghostbusters decide upon the following strategy:

They will pair off with the ghosts, forming n Ghostbuster-ghost pairs, and then simultaneously each Ghostbuster will shoot a stream at her chosen ghost. As we all know, it is *very dangerous* to let streams cross, and so the Ghostbusters must choose pairings for which no streams will cross.

Assume that the position of each Ghostbuster and each ghost is a fixed point in the plane and that no three positions are collinear. This problem has two parts (each 10 pts):

(A) Argue that there exists a line passing through one Ghostbuster and one ghost such that the number of Ghostbusters on one side of the line equals the number of ghosts on the same side. Describe how to find such a line in $O(n \log n)$ time.

- Assume that bottom, Left-most Point is the Ghost buster
- sort the remaining points from this point which I found above
- Then, visit the sorted points, keeping the difference between number of visited Ghostbusters and Ghosts.
- I have to follow this process until the difference is -1 and then connect that point to botton, left-most point.

for the sorting, it will take $O(n \log n)$ time

(B) Give an $O(n^2 \log n)$-time algorithm to pair Ghostbusters with ghosts in such a way that no streams cross.

The algorith that I developed in Part (a) will result in one pair of Ghostbuster and Ghost.

- On each side of the line, number of Ghostbusters and Ghosts are the same
- So, use the algorithm recursively on each side of the line to find the pairs.
- Consider worst case that is after an iteration we found that one side of the line doesn't have any Ghostbuster or Ghost.

- So, I need $n/2$ total iterations to find Pairings

This algorithm will result in $O(n^2 \log n)$ time

3. TRIPLE-SAT (20 pts):

Let TRIPLE-SAT denote the following decision problem: given a Boolean formula ψ, decide whether ψ has at least three distinct satisfying assignments. Prove that TRIPLE-SAT is NP-complete using a reduction.

To show that TRIPLE-SAT is in NP, for any input formula Φ, we would only need to guess three distinct assignments and verify that they satisfy Φ.

To show that TRIPLE-SAT is NP-hard, reduce SAT to it. Let Φ denote the input Boolean formula to a SAT problem, and suppose that the set of variables in Φ are $X = \{x_1, \dots, x_n\}$

Construct a Triple-SAT problem with a boolean formula Φ' $X' = \{x_1, \dots x_n, y, z\}$

$\Phi' = \Phi$

now we claim Φ is satisfiable iff Φ' has at least 3 satisfying assignments. If Φ is satisfiable. then we can augment any particular by adding any of the 4 possible pairs of values for $\{y, z\}$ to give at least four satisfying assignments overall. On the other hand, if Φ is not satisfiable, then neither is Φ'.

number of colors = number of clusters

Turn Graph coloring into clusters

4. Clustering (20 pts):

Consider the following approach to grouping n distinct objects $p_1 \cdots p_n$ on which a distance function $d(p_i, p_j)$ is defined such that $d(p_i, p_i) = 0$, $d(p_i, p_j) > 0$ if $i \neq j$ (that is, p_i and p_j are distinct objects), and $d(p_i, p_j) = d(p_j, p_i)$ (distances are symmetric.) The clustering approach is as follows: divide the n points into k clusters such that we *minimize* the maximum distance between any two points in the same cluster. That is, we seek low-diameter clusters. Suppose we want to know whether, for a particular set of n objects, and a bound B, the objects can be partitioned into k sets such that no two points in the same set are further than B apart. Prove that this decision problem is NP-complete.

It has been proven that reduction for graph coloring is NP-hard, this is a fact. so the idea is to prove that this decision problem is NP-complete. You can do so by creating a colored graph with three different colors. The vertices that are commected by a single edge cannot be in the same cluster, meaning it can't can't share the same color. Below is a colored graph that shows this decision problem is NP-complete ~~by~~ proved by a colored graph. # of colors is the same as the ~~#~~ of clusters.

(turn graph coloring into clusters)

K colors gets you K clusters

5. Knapsack approximations (20 pts):

In the Knapsack problem, we are given a set $A = a_1, ...a_n$ of items, where each a_i has a specified positive integer size s_i and a specified positive integer value v_i. We are also given a positive integer knapsack capacity B. Assume that $s_i \leq B$ for every item i. The problem is to find a subset of A whose total size is at most B and for which the total value is maximized. In this problem, we will consider approximation algorithms to solve the Knapsack problem. Notation: For any subset S of A, we write s_S for the total of all the sizes in S and v_S for the total of all the values in S. Let *Opt* denote an optimal solution to the problem. This problem has two parts (each 10 pts).

A: Consider the following greedy algorithm Alg_1 to solve the Knapsack problem:

Order all the items a_i in non-increasing order of their *density*, which is the ratio of value to size, $\frac{v_i}{s_i}$. Make a single pass through the list, from highest to lowest density. For each item encountered, if it still fits, include it, otherwise exclude it. **Prove** that algorithm Alg_1 does not guarantee any constant approximation ratio. That is, for any positive integer k, there is an input to the algorithm for which the total value of the set of items returned by the algorithm is at most $\frac{v_{Opt}}{k}$.

Imagine you have two-item list as input a_1, a_2, where $v_1 = 2$, $s_1 = 1$, $v_2 = 2K$, and $s_2 = 2K$, with knapsack capacity $B = 2K$. I know that $\frac{v_1}{s_1} = 2$ and $\frac{v_2}{s_2} = 1$ based on the info from above. Alg_1 considers the items in order a_1, a_2 and includes a_1 and excludes a_2. The total value achieved is only 2, but the largest achievable total value is $2K$.

B: Consider the following algorithm Alg_2:

If the total size of all the items is $\leq B$, then include all the items. If not, then order all the items in non-increasing order of their densities. Without loss of generality, assume that this ordering is the same as the ordering of the item indices. Find the smallest index i in the ordered list such that the total size of the first i items exceeds B. In other words, $\sum_{j=1}^{i} s_j > B$, but $\sum_{j=1}^{i-1} s_j \leq B$. If $v_i > \sum_{j=1}^{i-1} v_j$, then return a_i. Otherwise, return $a_1, ..., a_i - 1$.

Prove that alg_2 always yields a 2-approximation to the optimal solution.

This can be solved via a greedy algorithm. optimal solution for fractional knapsack takes first $i-1$ items and takes some fraction α of item i.

$$v_{f\,opt} = \sum_{j=1}^{i-1} v_j + \alpha_j$$

$v_{opt} \leq v_{f\,opt}$ holds true because every knapsack prob is a valid fractional knapsack.

$$\max\left(\sum_{j=1}^{i-1} v_j, v_i\right) \geq \max\left(\sum_{j=1}^{i-1} v_j, \alpha v_i\right) \geq \frac{v_{f\,opt}}{2},$$

one of the two terms has to be larger than half of $v_{f\,opt}$

$$v_{opt} \leq v_{f\,opt} \leq 2\max\left(\sum_{j=1}^{i-1} v_j, v_i\right),$$

This shows that Alg_2 is a 2-approximation algorithm.