CS787: Advanced Algorithms

Homework 2 Solution **Lecturer:** Shuchi Chawla

Exercise 1

Showing that Not-All-Equal-SAT is in NP is trivial. We prove that it is NP-hard by reducing SAT to it. Let ϕ be a 3-CNF formula. We are going to construct a CNF formula ϕ' such that ϕ is satisfiable if and only if ϕ' is a positive Not-All-Equal-SAT instance. We introduce a "reference variable" z. For each clause $(\ell_i \vee \ell_j \vee \ell_k)$ in ϕ (where the ℓ's are literals), add a clause $(\ell_i \vee \ell_j \vee \ell_k \vee z)$ in ϕ'. We claim that ϕ is satisfiable if and only if ϕ' is a positive Not-All-Equal-SAT instance. Suppose that ϕ is satisfiable. Then we can keep the values of the x_i's in the satisfying assignment and set z to false to satisfy ϕ' not-all-equally. Now suppose that ϕ' has a not-all-equal assignment. If z is false in this assignment, then this assignment is a satisfying assignment to ϕ. Otherwise, we can negate all the x_i's and get a satisfying assignment to ϕ. This shows that Not-All-Equal-SAT is NP-hard for 4-CNF formulas.

To show that Not-All-Equal-SAT is also NP-hard for 3-CNF formulas, we can simply break up each 4-literal clause into two 3-literal clauses, introducing a new variable as follows:

$$(x_i \vee x_j \vee x_k \vee z) \longrightarrow (x_i \vee x_j \vee w) \wedge (x_k \vee z \vee \bar{w})$$

Thus Not-All-Equal-SAT is NP-complete.

Exercise 2

Consider the complete graph K_n. The optimal solution to the LP relaxation is $x_v = 1/2$ for all $v \in V$, giving an optimal value of $n/2$. On the other hand, the minimum vertex cover has $n-1$ vertices (if there are two vertices not in the cover, then the edge between them is uncovered). The gap is $(n-1)/(n/2) = 2 - 1/n$.

Problem 1

Part a

We assume that the optimal solution contains at least 40 elements, otherwise we can find it through exhaustive search in polynomial time. The reason for this assumption is explained below.

We can solve this problem using randomized rounding in the same way that we did for set cover in class (although there is a better greedy algorithm that gives a $1 - 1/e$ approximation). We obtain a 10-approximation. The following is an LP relaxation of the max coverage problem. Here j refers to elements and i to the sets.

$$\begin{aligned}
\text{Maximize } \sum_j y_j \text{ s.t.} & \\
y_j \leq \sum_{i:S_i \ni j} x_i & \qquad \forall j \in E \\
\sum_{i=1}^{m} c_i x_i \leq B & \\
x_i, y_j \in \{0,1\} & \qquad \forall i, j
\end{aligned}$$

We can round the solution to this program as follows.

1. Solve the LP.
2. (Randomized rounding). $\forall$ i, pick S_i independently with probability $x_i/4$.
3. If the cost of the solution in step 2 is less than B and the number of elements covered is at least $1/10 \sum_j y_j$, then output the solution, else repeat step 2.

Following the analysis of set cover in class, we get that for a single run of step 2, the expected cost of the solution is $1/4 c_i x_i \leq B/4$. Therefore, applying Markov's inequality, the probability that the solution's cost exceeds B is at most 1/4.

Next, again following the analysis in class, the probability that an element j is not covered in the solution is at most

$$\prod_{i:j \in S_i} e^{-x_i/4} = e^{-1/4 \sum_{i:j\in S_i} x_i} \leq e^{y_j/4} \leq 1 - y_j(1 - e^{-1/4})$$

Here, the second to last inequality follows from noting that $\sum_{i:j\in S_i} x_i \geq y_j$, and the last follows from noting that $y_j \leq 1$ and $a^{-z} \leq 1 - z(1 - 1/a)$ for any $z \in [0,1]$ and $a > 0$.

Then, the expected number of elements covered, using the sum of expectations rule is at least $\sum_j y_j(1 - e^{-1/4}) > 40(1 - e^{-1/4}) > 8$ by our assumption at the beginning. We can now apply Chernoff bounds with $\delta = 1/2$ and get that with probability at least $1 - 1/e$, the actual number of elements covered is at least $1/2(1 - e^{-1/4}) \sum_j y_j > 1/10 \sum_j y_j$.

Therefore, with constant probability ($1/2 - 1/e > 0$), our solution from step 2 meets the criteria in step 3, and after a constant number of repetitions of step 2, we get the desired solution. Note that the factor of approximation in this algorithm can be improved to a factor arbitrarily close to $1 - 1/e$. The greedy approach gives an approximation factor of exactly $1 - 1/e$.

Part b

Suppose that maximum coverage can be approximated within a factor of $(1 - 1/e + \epsilon)$ for some constant $\epsilon > 0$. We will show how to obtain a $(1 - \epsilon) \ln n$-approximation for weighted set cover.

Let OPT be the optimal solution to some weighted set cover problem. If we set the bound of the corresponding maximum coverage problem to OPT, the optimal solution will cover all n elements, and hence by the hypothesis, our algorithm will give a solution that covers at least $n(1 - 1/e + \epsilon)$ elements. We remove all covered elements from the set and repeat the above process, until all elements have been covered. After t iterations, the number of remaining elements is at most $n(1/e - \epsilon)^t$. We want to find t such that $n(1/e - \epsilon)^t < 1$.

$$\begin{aligned} n(1/e - \epsilon)^t &< 1 \\ \ln n + t \ln(1/e - \epsilon) &< 0 \\ t \ln(1/e - \epsilon) &< -\ln n \\ t &> -\frac{\ln n}{\ln(1/e - \epsilon)} \end{aligned}$$

Note that $\ln(1/e - \epsilon) < -1$. Let $\epsilon' = 1 + 1/\ln(1/e - \epsilon) > 0$. Our above analysis shows that every element is covered after $(1 - \epsilon') \ln n$ iterations. The total weight of sets used in a single iteration is bounded by OPT, hence our overall set cover has total weight at most $(1 - \epsilon') \ln n \cdot OPT$. This gives us a $(1 - \epsilon') \ln n$ approximation for set cover.

(As an aside, it is NP-hard to obtain a $(1 - \epsilon) \ln n$ approximation for set cover for any constant $\epsilon > 0$.)

Problem 2

Part a

We give an $O(n^2)$ time algorithm for finding such a coloring. Let G be a graph with maximum degree Δ. The algorithm runs as follows:

> Pick an uncolored vertex and assign to it an arbitrary color that does not conflict with any of its colored neighbors. Repeat until all vertices have been colored.

It is clear that we can always find a feasible color in step 1, since the vertex has at most Δ neighbors which, in the worst case, eliminate Δ colors from the choices, leaving one feasible color for the vertex.

Part b

For each connected component of the graph, pick a starting vertex s and compute the shortest path distances from s to all other vertices in the component by doing breadth first search. Vertices with an odd distance are colored white, and the rest are colored black. This algorithm runs in $O(n^2)$ time.

Part c

We claim that the neighborhood of any vertex in a 3-colorable graph is 2-colorable. Let v be a vertex in a such a graph and $N(v)$ be the set of neighbors of v. Without loss of generality, suppose that v is colored with Color 1. Then none of the vertices in $N(v)$ is colored with Color 1, which means $N(v)$ can be colored with 2 colors. Such a 2-coloring can be found as described in part (b) by considering the subgraph induced by $N(v)$.

Making use of the above fact, we get the following algorithm:

1. Set k to 0.
2. Pick a vertex v with degree at least $\sqrt{n}$. If there is no such vertex, go to step 5.
3. Arbitrarily choose a subset S of $\sqrt{n}$ vertices adjacent to v and 2-color the subset with Color $3k+1$ and Color $3k+2$. Color v with Color $3k+3$.
4. Remove v and S (and all incident edges) from the graph. Increment k by 1. Go to step 2.
5. The remaining graph has maximum degree less than $\sqrt{n}$. We use the algorithm in part (a) to color it with $\sqrt{n}$ new colors.

Clearly, the algorithm runs in polynomial time and gives a valid coloring. The number of colors used is $3k + \sqrt{n}$, where k is the number of times step 4 gets executed. Note that at the each execution of step 4, $\sqrt{n}+1$ vertices are removed from the graph, thus step 4 can be executed at most $\sqrt{n}$ times. The number of colors used is therefore bounded by $4\sqrt{n}$.

Part d

The algorithm is pretty much the same as the one in part (c), except that in step 2 we require v to have degree at least $n^{2/3}$, and in step 3 we color a subset of $n^{2/3}$ neighbors of v (which is 3-colorable) using the algorithm in part (c). Again, it is clear that the algorithm runs in polynomial time and gives a valid coloring. The only thing we need to show is that the number of colors used is $O(n^{2/3})$. At the each execution of step 4, $n^{2/3}+1$ vertices are removed from the graph, thus step 4 can be executed at most $n^{1/3}$ times. The number of colors used in each iteration is $O(\sqrt{n^{2/3}}) = O(n^{1/3})$. At step 5, the graph has maximum degree less than $n^{2/3}$, and so can be colored using no more than $n^{2/3}$ according to part (a). Therefore, the total number of colors used in the algorithm is $O(n^{1/3} \cdot n^{1/3} + n^{2/3}) = O(n^{2/3})$.

Problem 3

Part a

Recall the LP for facility location:

$$\begin{array}{rll}
\text{minimize} & \sum_j f_j x_j + \sum_i \sum_j c_{ij} y_{ij} & \\
\text{subject to} & x_j - y_{ij} \geq 0 & \forall i,j \\
& \sum_j y_{ij} \geq 1 & \forall i \\
& y_i, x_{ij} \geq 0 & \forall i,j
\end{array}$$

By introducing a variable α_i for each customer i and β_{ij} for each customer-facility pair (i, j), we get the following dual LP.

$$\begin{array}{rll} \text{maximize} & \sum_i \alpha_i & \\ \text{subject to} & \sum_i \beta_{ij} \leq f_j & \forall j \\ & \alpha_j - \beta_{ij} \leq c_{ij} & \forall i, j \\ & \alpha_i, \beta_{ij} \geq 0 & \forall i, j \end{array}$$

We can interpret the dual LP as follows. The LP tries to recover the cost of opening facilities and routing from customers. For each customer i, α_i is the total amount paid by i. For each facility j and customer i, β_{ij} is the portion of i that pays for facility j. The constraint $\alpha_j - \beta_{ij} \leq c_{ij}$ states that the amount paid by customer i should not exceed the portion that goes to facility j plus the routing cost c_{ij} , for every facility j. The constraint $\sum_i \beta_{ij} \leq f_j$ states that no facility should overcharge its customers, i.e. the customers should not pay more than enough to open the facility. Under these constraints, we want to maximize $\sum_i \alpha_i$, the total amount collected from the customers.

Part b

The main idea is that feasible solutions to the relaxed LP can partially assign each customer to several partially-open facilities whereas the original ILP can only fully assign each customer to a single facility. If each customer has many possible facilities available, then this advantage becomes significant.

Suppose we have n customers and n facilities where $n \geq 2$. We let the cost $c(i, j)$ of routing customer i to facility j be 1 if $i \neq j$ and 3 if $i = j$ (the verification that this defines a metric is left to the reader). Each facility j will have cost $f_j = b$ for some constant b to be determined later.

First we determine the value of an optimal ILP solution. If we only open one facility j, the total cost is $b + (n - 1) + 3$, as the minimum routing cost for customer j is 3, and all other customers have routing cost 1. If we open two facilities, then each customer has routing cost 1 and the total cost is $2b + n$. Clearly, opening more than two facilities is suboptimal. The minimum of both of these lower bounds is maximized when $b = 2$ so we let this be the cost of opening any facility and an optimal ILP solution has cost $n + 4$.

Now we find a good feasible solution to the relaxed LP. Each facility will be a $(\frac{1}{n-1})$-fraction open so that each customer can split its requirements uniformly over $n - 1$ facilities and maintain a total routing cost of 1 per customer. This means the total value of the objective function is $bn/(n - 1) + n = 2n/(n - 1) + n$, so the integrality gap is $\frac{n+4}{2n/(n-1)+n}$ which is maximized at $n = 4$, yielding a gap of 6/5. As an aside, note there are similar costructions that achieve gaps of 5/4 and 4/3.

Problem 4

Part a

Recall that we construct a feasible solution $(\tilde{y}, \tilde{x})$ as follows.

1. For each customer i, let $S_i = \{j | c_{ij} \leq 2A_i\}$ be the set of *nearby* facilities.
2. For each i and j, if $j \notin S_i$ then set $\tilde{y}_{ij} = 0$; otherwise, set $\tilde{y}_{ij} = y^*_{ij}/\alpha_i$ where $\alpha_i = \sum_{j \in S_i} y^*_{ij}$ is the normalizing factor.
3. For each facility j, let $\tilde{x}_j = \min(2x^*_j, 1)$.

We interpret the problem of picking a subset of facilities as a set cover problem. The elements are the set of customers. For each facility j, there is a set $C_j = \{i | j \in F(i)\}$ with weight f_j. We want to cover all customers with a minimum cost collection of facilities. This is exactly the weighted set cover problem. The LP relaxation for this set cover problem is:

$$\begin{array}{rll} \text{minimize} & \sum_j f_j x_j & \\ \text{subject to} & \sum_{j \in F(i)} x_j \geq 1 & \forall i \\ & x_j \geq 0 & \forall j \end{array}$$

We now use the LP-rounding based approach to get an $O(\log n)$ approximation for set cover. Let x' be the integral solution obtained by the approximation algorithm. Observe that for every customer i, $\sum_{j \in F(i)} \tilde{x}_j \geq \sum_{j \in F(i)} \tilde{y}_{ij} = \sum_{j \in J} \tilde{y}_{ij} \geq 1$. Hence, $\tilde{x}$ is a feasible solution of the above LP and the (facility opening) cost of x' is at most $O(\log n)$ times that of $\tilde{x}$. We extend x' to a solution for the facility location problem and call it (y', x'). The routing cost of this solution is at most $\sum_i 2A_i = \sum_i \sum_j c_{ij} y^*_{ij} \leq \text{LPC}(y^*, x^*) \leq \text{LPC}(\tilde{y}, \tilde{x})$. Therefore, $\text{LPC}(y', x')$ is at most $O(\log n)$ times $\text{LPC}(\tilde{y}, \tilde{x})$. Since $\text{LPC}(\tilde{y}, \tilde{x}) \leq 2 \cdot \text{LPC}(y^*, x^*)$, $\text{LPC}(y', x')$ is at most $O(\log n)$ times $\text{LPC}(y^*, x^*)$ — we have achieved a $O(\log n)$ approximation.

Problem 5

Part a

We can find a minimum cycle cover for a complete directed weighted graph $G = (V, E)$ by reducing it to a weighted bipartite matching problem. We define the complete bipartite graph $\Gamma = (L, R, E')$ so that $L = V$, R is a "copy" of V, and every edge $(u, v) \in E'$ has weight identical to its counterpart in E, except that for all $u \in V$, (u, u) has prohibitively large weight (say, larger than all of the other edges combined). Next we note that any perfect matching in Γ that doesn't contain edges of the form (u, u) induces a cycle cover on G. In particular, each vertex in G is incident on exactly two edges in the minimum weight perfect matching, one through the right set R and one through the left set L. Moreover, the weight of the matching is exactly equal to the weight of the cycle cover. Therefore, the optimal perfect matchings for Γ correspond precisely with the optimal solutions for a cycle cover of G.

Part b

The idea is to first get a minimal cycle cover for the graph by running the algorithm above. Then for each cycle, we remove an edge from that cycle and introduce a new edge to connect it to the next cycle. Carrying out this transformation for all cycles in the obvious way converts the cycle cover into a valid TSP tour.

```
TSP-{1-2}({1,2}-graph $G = (V, E)$)
    Let $G' = (V, E') = \textsc{CycleCover}(G)$
    Let $C_0, \ldots, C_{k-1}$ be the disjoint cycles of $G'$
    **foreach** $i \in \{0, \ldots, k-1\}$
        Let $(x_i, y_i)$ be an edge in $C_i$
    Let $E'' = \{\cup_{i=0}^{k-1}(x_i, y_{(i+1) \bmod k})\} \cup E' \setminus \{\cup_{i=0}^{k-1}(x_i, y_i)\}$
    **return** $(V, E'')$
```

Let the number of cycles in the cycle cover be k. If we use c to denote the cost of the cycle cover, then since edge weights are in $\{1, 2\}$, each cycle has at least two vertices, and $c \leq OPT$, we get

$$\begin{aligned} ALG &= c - \sum_{i=0}^{k-1} w((x_i, y_i)) + \sum_{i=0}^{k-1} w((x_i, y_{(i+1) \bmod m}) \\ &\leq c - k + 2k \\ &\leq c + c/2 \\ &\leq (3/2)OPT. \end{aligned}$$

For the last inequality we used the fact that $c \geq 2k$, because each cycle contains at least two edges of cost 1 each.