Assignment2

Praneeth Kacham
2015CS10600

September 2018

1 Question-1

1.1 Part-1

Let (U, W) be any partition of V with $|U| = k$. Then, assign $x_i = 1\ \forall i \in U$. Consider the edge $(i, j) \in E$. If (i, j) is in the cut, then $x_i \neq x_j$ i.e., $x_i = 1 \wedge x_j = 0$ or $x_i = 0 \wedge x_j = 1$. Then $x_i + x_j - 2x_ix_j = 1$ in both cases. If, edge (i, j) is not in cut, then $x_i = x_j = 0$ *or* 1. Then $x_i + x_j - 2x_ix_j = 0$ in both cases. Thus $I[e = (i, j)\ is\ in\ cut] = x_i + x_j - 2x_ix_j$. Hence, weight of the cut = $\sum_{(i,j)\in E} w_{ij} I[(i, j)\ in\ cut] = \sum_{(i,j)\in E} w_{ij}(x_i + x_j - 2x_ix_j)$. For every cut with smaller set having k elements, there is a feasible solution with the objective cost equal to weight of the cut. Now, let x be a feasible solution. Define $U = \{i | x_i = 1\}$. We have $|U| = k$ and weight of the cut $(U, V) = \sum_{(i,j)\in E} w_{ij}(x_i + x_j - 2x_ix_j)$. So, for every feasible solution, there exists a cut (U, V) of the weight same as the objective function. Hence, the given non-linear integer program models the maximum cut problem.

1.2 Part-2

To show that the linear program is a relaxation of the non linear integer program, we have to show that the feasible region of the non-linear program is a subset of the feasible region of the linear program. Let x be a feasible solution for the non-linear program. Now define $z_{ij} = x_i + x_j - 2x_ix_j$. We have $0 \leq x_i \leq 1$ (from feasibility of x for integer non-linear program), $0 \leq z_{ij} \leq 1$(Since, it takes only 0,1 values), $z_{ij} \leq x_i + x_j$ (Since, $x_ix_j \geq 0$) and we also have $(1 - x_i)(1 - x_j) \geq 0 \Rightarrow 1 + x_ix_j \geq x_i + x_j \Rightarrow 2 - x_i - x_j \geq x_i + x_j - x_ix_j = z_{ij}$. Hence, (x, z) is feasible for the linear program with the same objective function value as Integer non-linear program. So, the given LP is a relaxation of the integer non-linear program.

1.3 Part-3

We show that for any feasible (x, z) we have $x_i + x_j - 2x_ix_j \geq (1/2)z_{ij}$ from which the required proof follows. We have

$$\begin{aligned} 2[x_i + x_j - 2x_ix_j] &\geq 2[x_i + x_j - 2\left(\frac{x_i + x_j}{2}\right)^2] \qquad (Since,\ x_ix_j \leq (x_i + x_j)^2/4] \\ &= 2x_i + 2x_j - (x_i + x_j)^2 \\ &= (x_i + x_j)(2 - x_i - x_j) \end{aligned}$$

Case-1: $x_i + x_j \geq 1$ Then we have $(x_i + x_j)(2 - x_i - x_j) \geq (2 - x_i - x_j) \geq z_{ij}$(Feasibility).
Case-2 : $x_i + x_j \leq 1 \Rightarrow 2 - x_i - x_j \geq 1$. We have, $(x_i + x_j)(2 - x_i - x_j) \geq (x_i + x_j) \geq z_{ij}$(Feasibility).

In both the cases, we get $2[x_i + x_j - 2x_ix_j] \geq (x_i + x_j)(2 - x_i - x_j) \geq z_{ij}$. Hence, for feasible (x, z), $x_i + x_j - 2x_ix_j \geq z_{ij}/2$.

Which shows that for all feasible (x, z), $F(x) = \sum_{(i,j)\in E} w_{ij}(x_i + x_j - 2x_ix_j) \geq (1/2)\sum_{(i,j)\in E} w_{ij}z_{ij}$.

1.4 Part-4

The objective function to maximize is $F(x) = \sum_{(i,j)} w_{ij}(x_i + x_j - 2x_ix_j)$. Consider adding ϵ to x_l and subtracting ϵ from x_m. Now the objective function as a function of ϵ

$$\phi(\epsilon) = \epsilon^2(2w_{lm}) + 2\epsilon(x_l - x_m) + \epsilon \sum_{(l,j)\in E: j\neq m} w_{lj}(1 - 2x_j) - \epsilon \sum_{(m,j)\in E: j\neq l} w_{mj}(1 - 2x_j) + F(x)$$

ϕ is a quadratic in ϵ with a non-negative coefficient and hence ϕ is convex in ϵ.

Taking $\epsilon = min\{x_m, 1 - x_l\}$ takes one of the variables x_m, x_l to integer values. Taking $\epsilon = max\{-x_l, -(1 - x_m)\}$ also takes one of the variables x_l, x_m to integer values (0,1). We have $0 \in [max\{-x_l, -(1 - x_m)\}, min\{x_m, 1 - x_l\}]$ and ϕ is convex. So, $\phi(0) \leq max\{\phi(max\{-x_l, -(1 - x_m)\}), \phi(min\{x_m, 1 - x_l\})\}$. So, using one of the values $max\{-x_l, -(1 - x_m)\}, min\{x_m, 1 - x_l\}$, we can increase the objective value and as well as convert one of x_l, x_m to binary values.

1.5 Part-5

First solve the LP to obtain optimal LP solution (x^*, z^*). We have shown that $F(x^*) \geq OPT(LP)/2 \geq OPT(MAX\_CUT)/2$. If all the variables in x are $0, 1$, we are done. Otherwise due to the condition that $\sum_i x_i = k \in \mathbb{Z}$, we will have to variables say x_i, x_j such that $0 < x_i, x_j < 1$. Using the method in part-4, we can convert one of the variables into 0,1 while preserving feasibility and not decreasing the objective value. Let the x be x^1. Continue this process till we obtain an integer solution x. This process will take atmost n steps as we are decreasing no. of non-integer variable by atleat 1 in each step.

And $F(x) \geq \ldots \geq F(x^1) \geq F(x^*) \geq OPT/2$. Hence, this is a (1/2)-approximation.

2 Question-2

Let

$$X_i = \begin{cases} 1, i \in U \\ 0, otherwise \end{cases}$$

and

$$Z_{ij} = \begin{cases} 1, X_i = 1 \wedge X_j = 0 \\ 0, otherwise \end{cases}$$

X_i is defined with respect to where the randomized algorithm places i.For a solution output by the algorithm, weight of cut is given $\sum_{(i,j)\in E} w_{ij}Z_{ij}$. So, $E[wt.ofcut] = \sum_{(i,j)\in E} w_{ij}E[Z_{ij}]$. Now, $E[Z_{ij}] = P[Z_{ij} = 1] = P[X_i =$

$1 \wedge X_j = 0] = P[X_i = 1]P[X_j = 0]$ (Since, X_i, X_j are independent).

$$\begin{aligned} P[X_i = 1]P[X_j = 0] &= (1/4 + x_i/2)(1 - 1/4 - x_j/2) \\ &= (1/4 + x_i/2)(1/4 + (1 - x_j)/2) \\ &\geq (1/4 + z_{ij}/2)(1/4 + z_{ij}/2) \ (Since, (x, z) \ is \ feasible) \\ &\geq 4 * (1/4) * (z_{ij}/2) \ (Since, (a+b)^2 \geq 4ab) \\ &\geq z_{ij}/2 \end{aligned}$$

So, $E[wt.ofCut] = \sum_{(i,j)\in E} w_{ij} E[Z_{ij}] \geq \sum_{(i,j)\in E} w_{ij} z_{ij}/2 \geq OPT/2$.

Hence, the randomized algorithm is a 2-approximation.

3 Question-3

3.1 Part-1

Let x be a feasible solution for the Integer Program. Define the set S as $\{e | x_e = 1\}$. Now, weight of the sets covered by the set S = $\sum_j w_j I[S \cap S_j \neq \phi]$.

$S \cap S_j \neq \phi \iff \exists e, e \in S_j \wedge e \in S \iff \exists e \in S_j : x_e = 1 \iff \Pi_{e\in S_j}(1 - x_e) = 0 \iff 1 - \Pi_{e\in S_j}(1 - x_e) = 1$. So, S covers S_j iff $1 - \Pi_{e\in S_j}(1 - x_e) = 1$. And similarly S doesn't cover S_j iff $\forall e \in S_j : x_e = 0 \iff \Pi_{e\in S_j}(1 - x_e) = 1 \iff 1 - \Pi_{e\in S_j}(1 - x_e) = 0$. Hence, $I[ScoversS_j] = 1 - \Pi_{e\in S_j}(1 - x_e)$. Thus, for every feasible solution there is a set of size k which covers the sets with weight equal to the objective value.

Consider any set S with $|S| = k$. Define x as $x_e = 1$ if $e \in S$ and $x_e = 0$ otherwise. We have x is feasible as $\sum_e x_e = k$ and $x_e \in \{0, 1\}$. Weight of the sets covered by S is $\sum_j w_{S_j} I[S \cap S_j \neq \phi] = \sum_j w_{S_j}(1 - \Pi_{e\in S_j}(1 - x_e))$. Thus, for every subset S of E with $|S| = k$, we have a feasible solution for the integer program with objective value equal to the weight of the sets covered by S. Thus the two problems are equivalent.

3.2 Part-2

To prove that the linear program is a relaxation, we need to show that for every feasible x for IP, we have a feasible solution for (x, z) with same objective value.

Consider a feasible x for the IP. We have $x_e = 0$ *or* 1. Now define $z_j = 1 - \Pi_{e\in S_j}(1 - x_e)$. We have $z_e = 0 \iff x_e = 0 \forall e \in S_j$ and $z_e = 1 \iff \exists e \in S_j : x_e = 1$. So, $\sum_{e\in S_j} x_e \geq 1$. Hence, (x, z) is feasible and the objective function values are same for IP and LP. Hence, LP is a relaxation of IP.

3.3 Part-3

Claim-1: For any feasible (x, z), we have $(1 - \Pi_{e\in S_j}(1 - x_e)) \geq (1 - 1/e)z_e$.

Proof. We have $1 - x \leq e^{-x}$. So, $\Pi_{e\in S_j}(1 - x_e) \leq e^{-\sum_{e\in S_j} x_e}$. But $z_j \leq \sum_{e\in S_j} x_e$. So, $e^{-\sum_{e\in S_j} x_e} \leq e^{-z_j}$. Which implies $1 - \Pi_{e\in S_j}(1 - x_e) \geq 1 - e^{-z_j}$. Consider the function $f(x) = 1 - e^{-x}$. We have that $f(x)$ is concave, $f(0) = 0$ and $f(1) = 1 - 1/e$. Given that $0 \leq z_e \leq 1$ and f concave, $f(z_e) \geq (1 - z_e)f(0) + z_e f(1) = z_e(1 - 1/e)$.

So, for any feasible (x, z), we have that $(1 - \Pi_{e\in S_j}(1 - x_e) \geq (1 - 1/e)z_e$. □

It trivially follows from the above claim that, if (x^*, z^*) is the optimal solution for LP, we have $\sum_j w_j(1 - \Pi_{e\in S_j}(1 - x_e^*)) \geq (1 - 1/e)\sum_j w_j z_j^* = (1 - 1/e)OPT\_LP \geq (1 - 1/e)OPT\_IP$.

Claim-2 : Given any non-binary feasible solution, for any two non-binary variables x_e and $x_{e'}$, $\exists \epsilon$ such that $x_e + \epsilon$ and $x_{e'} - \epsilon$ also form a feasible solution and the objective value atleast the objective value with x.

Proof. Let $F(x) = \sum_j w_j(1 - \Pi_{e \in S_j}(1 - x_e))$ and $\phi(\epsilon) = F((.., x_e + \epsilon, ..., x_{e'} - \epsilon))$. Define α_j as follows:

$$\alpha_j = \Pi_{w \in S_j : w \neq e, e'}(1 - x_w) \tag{1}$$

Hence, $\phi(\epsilon) - F(x) = \sum_{j: e \in S_j \wedge e' \notin S_j} \alpha_j \epsilon + \sum_{j: e \notin S_j \wedge e' \in S_j} -w_j \alpha_j \epsilon + \sum_{j: e, e' \in S_j} w_j(-\alpha_j \epsilon(x_{e'} - x_e) + \alpha_j \epsilon^2)$. So, $\phi(\epsilon)$ is a quadratic in ϵ with coefficient of ϵ^2 being $\sum_{j: e, e' \in S_j} w_j \alpha_j$. But $w_j, \alpha_j \geq 0$. So, $\phi(\epsilon)$ is convex in ϵ. Consider the interval $[max(-x_e, x_{e'} - 1), min(1 - x_e, x_{e'})]$. We have 0 lying in the interval and convexity implies $\phi(0) = F(x) \leq max(\phi(max(-x_e, x_{e'} - 1)), \phi(min(1 - x_e, x_{e'})))$. Take the one which is max of them and adding that ϵ to x_e and subtracting ϵ from $x_{e'}$ will make one of the variables $x_e, x_{e'}$ binary. And feasibility is still maintained as the sum of variables is still k. □

Consider the following algorithm:

1. Obtain the optimal solution for LP say (x^*, z^*). $x := x^*$.
2. while(x is not binary)Pick 2 non-binary variables from x and round one of the two using the above procedure and assign the modified solution to x.

Claim-3: The algorithm terminates and gives a solution which is $(1 - 1/e)$ approximation to the optimum.

Proof. Given that $\sum_e x_e = k \in \mathbb{Z}$. So, in any non-binary feasible solution, we have atleast two literals with non-binary values. And from the above claim, we have that the objective value never decreases. So, in each iteration, we decrease the number of non-binary values by atleast 1 and the objective value doesn't decrease. Hence, the algorithm terminates in atmost $|E|$ steps.

Let the solution obtained be x. So, we have $F(x) \geq F(x^*)$. But from Claim-1, we have that $F(x^*) \geq (1 - 1/e) \sum w_e z_e^* \geq (1 - 1/e)OPT$. So, $F(x) \geq (1 - 1/e)OPT$. □

4 Question-4

First root the tree at node r. The problem can be solved in two phases. First collect all the items from s_i's and leave them at the lowest common ancestor of s_i and t_i, then return to r and then start distribution phase in which the collected items are distributed to respective t_i's. We formalize this algorithm as follows: Algorithm 1 collects

Algorithm 1: COLLECT

```
Input: A tree T, C : int
Output: Modifies Tree T with items placed at Lowest Common Ancestor
root ← T.getRoot();
subtrees ← root.getSubtrees();
for subtree in subtrees do
    if subtree.hasSourceNodes() then
        COLLECT(subtree, C);
    move items collected at subtree.getRoot() whose common ancestor is root or its ancestors to root in sizes
     of C;
```

the items from sources and then places item i at the lowest common ancestor of (s_i,t_i). The algorithm takes an edge only when there is a source node in the subtree and while transporting the items from one end of the edge to another end of the edge.

4.1 Analyzing the cost of tour taken by the Algorithm 1

Let P_i be the directed path from s_i to t_i. Define f_e as the number of paths P_i through edge e in the upward direction i.e., in the direction of root r. So, the tour takes an edge either 0 times (when there are no source nodes in the subtree attached to the edge) or $\max\{2, 2\lceil \frac{f_e}{C} \rceil\}$. 2 times when there is no path going in the direction of root r through the edge e but there are source nodes in the subtree rooted at the end of edge e which is away from root r and $2\lceil \frac{f_e}{C} \rceil$ when there is flow in the other direction.

So,

$$\text{Cost of Collect Tour} = \sum_{\substack{e=(u,v): u \text{ is parent of } v \text{ and} \\ \text{there are source nodes} \\ \text{in subtree rooted at } v}} w_e * \max\{2, 2\lceil \frac{f_e}{C} \rceil\} \tag{2}$$

Lemma 4.1. *Cost of Collect Tour $\leq$ OPT, where OPT is the optimal tour cost for the problem.*

Proof. Consider the optimal tour. Let us see how many times an edge $e = (u, v)$ is taken by the optimal tour. Without loss of generality assume that u is the parent of v in the tree rooted at r.

Case1 : No source-sink pairs in subtree rooted at v.

The optimal tour need not take the edge (u, v). If it takes the edge (u, v), it should also take the edge back (u, v) in the way back to r. Then entire cycle can be shortcut without affecting anything as there are no source-sink pairs in the subtree. So, the edge is not taken.

Case2: There are source nodes in the subtree rooted at v.

Say s_i is the source node in the subtree rooted at v. Optimal tour should definitely visit s_i and taking the edge (u, v) is the only way it can get to s_i and it should take the edge back on its way to r. Moreover the optimal tour should carry atleast f_e items along the edge (u, v) in the direction v to u across the entire tour. So, number of times the edge (u, v) is taken is $\geq 2\lceil \frac{f_e}{C} \rceil$.

Case3: There are sink nodes in the subtrees rooted at v.

Optimal tour has to take the edge ≥ 2 times to visit the sink node and go back to root r.

So,

$$OPT = \text{Optimal tour cost} \geq \sum_{e: source nodes \in subtree(e)} w_e * \max\{2, 2\lceil \frac{f_e}{C} \rceil\} = \text{Cost of Collect Tour} \tag{3}$$

□

4.2 Distribute Tour

After each of the items are placed at lowest common ancestor, distribute tour distributes them to the sinks. We define the distribute tour as follows. Let $\mathcal{I}$ be the given instance of the problem. Consider the instance $\mathcal{I}'$ of the $dial-a-ride$ problem in which we have the same metric but (t_i, s_i) pairs instead of (s_i, t_i). Now find the collect tour for this instance using the Algorithm 1. In the tour, instead of bringing the items to root, we distribute items which are at the root to respective sinks in exactly reversing the manner in which the items are brought to root in Collect tour. As can be seen easily $OPT(\mathcal{I}) = OPT(\mathcal{I}')$(Just take the tour in reverse order) and Cost of collect tour of $\mathcal{I}'$ = Cost of distribute tour of $\mathcal{I}$ which implies that

$$\begin{aligned} \text{Cost of Collect}(\mathcal{I}) + \text{Cost of Distribute}(\mathcal{I}) &= \text{Cost of Collect}(\mathcal{I}) + \text{Cost of Collect}(\mathcal{I}') \\ &\leq OPT(\mathcal{I}) + OPT(\mathcal{I}') \\ &= 2OPT(\mathcal{I}) \end{aligned}$$

Hence, the algorithm collect followed by distribute is a 2-approximation.

4.3 Part-b

We can make use of $O(\log(n))$ tree embedding to give a $O(\log(n))$ randomized algorithm. Let the optimal tour be a sequence of tuples (u, v) where the tour goes from u to v by taking the edge (u, v). Let T be a tree in the distribution of the trees. For every edge (u, v) in the sequence of optimal tour, pick the path (u, v) in T. So, the cost of this tour in T is $\sum_{(u,v)\in OPT} d_T(u, v)$. So, $OPT(T) \leq \sum_{(u,v)\in OPT} d_T(u, v)$. Let $Alg(T)$ be the tour given by the collect and distribute algorithm for tree T. So, $Alg(T) \leq 2OPT(T) \leq 2\sum_{(u,v)\in OPT} d_T(u, v)$. Now map back the solution $Alg(T)$ to the original graph by just picking the edge (u, v) in G instead of path $(u, v) \in T$. Let the cost of solution be $Alg_G(T)$. By definition of tree embedding we have

$$Alg_G(T) \leq Alg(T) \tag{4}$$

Now,

$$\begin{aligned}
E_T[Alg_G(T)] &\leq E_T[Alg(T)] \\
&\leq 2E_T[OPT(T)] \\
&\leq 2E_T[\sum_{(u,v)\in OPT} d_T(u, v)] \\
&= 2\sum_{(u,v)\in OPT} E_T[d_T(u, v)] \\
&\leq 2 * O(\log(n)) \sum_{(u,v)\in OPT} d_G(u, v) \\
&= O(\log(n))OPT
\end{aligned}$$

So, the expected cost of the tour obtained on the tree is $O(\log(n))$OPT. Hence, this is a $O(\log(n))$ approximation.