Homework 2

Algorithm Design
Università di Roma "La Sapienza"

Francesco Bovi 1803762
George Ciprian Telinoiu 1796513

13 gennaio 2021

1 Exercise 1

1.1 1.a

Given m as the number of elves and n the number of houses we can construct the problem as ILP:

$$min(Z),$$

$\sum_i^m X_{ij} = 1 \ \forall j = 1, .., n, \ \sum_j^n X_{ij} \geq 1 \ \forall i = 1, .., m, \ \sum_j^n X_{ij} \leq Z \ \forall i = 1, .., m, \ X_{ij} \in \{0, 1\} \ \forall i, j.$
We should consider that for every elf it is not guaranteed that he can do every house, but he surely is able to do at least 1. To relax the problem to LP just consider as non-integer our variable $X_{ij} \in [0, 1] \ \forall i, j.$

1.2 1.b

Given in input an optimal solution to our LP formulation, we now need to adjust it to being a concrete solution, considering that the OPT of LP is not acceptable as it is. What we want to do, specifically, is to transform each variable X_{ij} into an integer value. The way that we decided to proceed, considering that we still need to have a feasible solution to the original problem; so we should always keep in mind the same restrictions, is the following: Consider one elf, iterate over all the houses that he he might be able to do and select the highest X_{ij} value that he has. If there are is no single max value, select the highest subset that he has, and from that choose one randomly. This is possible since the cost for each house is exactly the same, that being 1 hour, so either variable will do. After selecting a certain house for that elf, set to 0 all other values for the remaining elves for that specific house, since for sure we need to have exactly one gift for each house. Proceed to do the same for the next elf, considering the same approach, until all variables are integer.
This solution is for sure as good as the optimum that we had since the price payed by each selected elf is the same, so we are sure that even if we had, for a certain house, an elf that had a higher value, the max price for that house would still have been achieved, and at the end we would still minimize the time until the last elf is finished

2 Exercise 2

The approach that we chose in order to prove that Santa problem is hard was trough a polynomial reduction from a known NP-Complete problem, specifically SAT. The reduction that we came up is by restriction, that being to show that Santa problem is the same as SAT but with an additional restriction. A known example of this kind of reduction is the one of reducing the Hamilton Cycle problem to TSP, which bases itself on the fact that TSP is exactly HC but with edges of weight one and the restriction that we want to perform the cycle with a value set to "n" which is the number of vertices.
Therefore we started from considering a general input of a SAT problem in CNF form, where we have k clauses, each with a variable number of literals from 1 to n. This formulation is exactly equivalent to saying that we want to satisfy the k constraints that Santa has for us, which will lead to the construction of a decent sleigh. Considering that we theoretically have an infinite number of parts in the set P, if we were able to build a decent sleigh, for sure we will be able to build 187 of them, since if there exists one truth assignment that satisfies all k constraints, for sure we can replicate it 187 times since it would use the same truth assignment. We now need to find a way to also include the restriction of having 187 different sleighs, while still having that a solution for SAT is also a solution for Santa. At this point we decided to include an intermediary step in order to simplify our reduction and the way we explain it. It has been proven that SAT reduces to the k-Clique problem, and that a solution for SAT is also a solution for the k-Clique problem, so what we said before still holds even if we decide to represent our problem with this new setting. If we now consider that each k- Clique is 1 decent sleigh, that being again a sleigh that satisfies at the same time all k constraint, and that we can still replicate it 187, under the assumption that again we have some spare parts from our set P, if we were to add to each k-Clique two nodes, one with a variable and the other with its negation, we would still maintain the same k-Clique, therefore we would still have a decent sleigh, but now we made that sleigh a bit more special. This can be done 187 times with 187 different variables, and considering that the nodes that we have added cancel each other out in our initial truth assignment (similar to what we do in SAT to 3-SAT reduction) we have found a way to transform our original SAT input into Santa's input, which is exactly what a polynomial reduction requires and we know for sure that it is polynomial since SAT to k-Clique is polynomial and adding 2 nodes to each k-Clique is linear in time.

3 Exercise 3

3.1

In order to design a cut game in which the **Price of Anarchy** is 2 we used a graph composed by 4 vertices, each vertex is named v_i (where i represents the player that controls the vertex) and the edges have unit weights and they make up the following graph: [1,2],[1,3],[2,4] and [3,4]. We know that each player must choose a set between {LEFT,RIGHT} and that the CUT(alpha) is the set of the edges that have one endpoint in each set. Let's now define a variable S_v as the sum for each vertex of all the edges' weight of that vertex that are in the CUT (so is $\sum_i u_i(alpha)$), so in this case the PoA is the maximum of S_v in every possible configuration divided by the minimum of S_v in all the possible **Pure Nash Equilibrium** configurations. Given the graph explained previously we know that the maximum of S_v is obtained when RIGHT=1,4 and LEFT=2,3 (or vice-versa), in fact in this case each vertex has both the edges in the CUT (that would be composed by all the edges) and so the the total weight of each vertex is 2 and so the max_{S_v} is 8. For the minimum we must only consider the PNE configurations, that is for this graph when we have when 2 adjacent vertices in the same set. Therefore the total weight of each vertex is 1, so the min_{S_v} is 4. Being the PoA max_{S_v}/min_{S_v} by using this configuration we get exactly 2.

3.2

We know that in a PNE no player would benefit by deviating his decision, so considering the case in which all the edges have the same weight in a PNE the total S_{vi} for a vertex v_i must be at least half of the total possible weight because if it were lower it would mean that the majority of the vertices adjacent to the v_i are in the same set of it and so the player i would prefer to change sets to get a greater weight. Starting from this consideration we have that max_{S_v} is for each vertex ideally the sum of the weights of all the edges (it cannot be always possible) and min_{S_v} is the same vertex at least the half the sum of the weights of all the edges (because if it were less that player could change his decision and therefore would not be a PNE) and so is at least the half of max_{S_v}, so the PoA cannot be more than 2. The fact that we are assuming all the edges of the same weight is it is a restrictive choice because in the case with different edges in max_{S_v} will be able to choose the bows with greater weight but even in min_{S_v} in order to be a PNE so the value of PoA will be more similar to 1 than the case of all the same weights.

4 Exercise 4

4.2

The idea is to apply the center selection algorithm, that we have seen on the slides, first to S and then for each city in S to choose its nearest Antenna.Let OPT be the optimal value. For any s_i ∈ S let a_i ∈ A be its nearest antenna. (If there are more nearest antennas then select any of these as s_i .) Clearly, we must have dist(s_i , a_i) < OPT for any i. Apply the center selection algorithm to the set S of cities. This gives a set S' = $s_1, s_2, ..., s_k$ S of k cities.
Take ALG_SOL = a_i | c_i ∈ C' as solution. We prove that this gives 3-approximation.Consider an arbitrary city v ∈ S. If there is a city s_i ∈ S' at distance at most 2*OPT from v, then by the triangle inequality dist(v, a_i) ≤ dist(v, c_i) + dist(c_i , a_i) ≤ 2*OPT + OPT = 3*OPT.Now assume there is no city c_i ∈ S' at distance at most 2*OPT from v. Then, by construction of the algorithm, for any two cities in c_i , c_j ∈ S' we have dist(c_i , c_j) > 2*OPT.But then the set C' ∪ v consists of k+1 cities such that any pair is at distance strictly more then 2*OPT.This is impossible since in that case at least k + 1 antennas are needed for a solution with value OPT.
4.3 & 4.1
In order to prove hardness we decided to proceed with a reduction from Dominating Set, and show that only a 3-approximation is feasible . Given an instance G = (V, E) of the dominating set problem we define distances as follows: dist(a_v, a_w) = dist(c_v, c_w) = 2 for each pair v, w ∈ V, then dist(c_v, a_w) = 1 if v = w or if (v, w) ∈ E and finally 3 otherwise. This satisfies the triangle inequality. It's important to notice that any solution has either value 1 or 3. Assume that there is a dominating set S of size k. Consider as a solution for the problem instance the antennas that correspond with the vertices in S. Then, the value of this solution is 1 as we wanted. For the other direction, assume that there is a solution for the k antennas instance of value 1. Then, take as dominating set S the vertices that correspond with the k antennas. Also this is a dominating set. Therefore we have a dominating set of size k if and only if the optimal value of the corresponding k antennas instance is 1. Now, since the optimal value is either 1 or 3 this also gives us the lower bound of 3 on the approximation ratio. Assume we have an α-approximation algorithm Alg for the k-antennas problem with $\alpha < 3$ that we can use. Assume we are given an instance G, k of the dominating set problem. That means, we want to compute if there exists a dominating set with at most k vertices. Now we reduce it to the instance of the problem as defined above and we apply the α-approximation algorithm.If there is a dominating set of value at most k then, as argued above, the optimal value of the k-antennas instance is 1 (OPT_k=1) and the answer given by the algorithm will be at most α * 1 < 3. If, on the other hand, there is no dominating set of value at most k then the optimal value of the problem instance is 3 and the answer given by the algorithm will be (at least) 3. Therefore, by looking at the output of the algorithm we know if there is a dominating set of size at most k. However, we know that there is no polynomial time algorithm that solves the dominating set problem, assuming that P ≠ NP. We conclude that no α-approximation algorithm with $\alpha < 3$ exists, assuming that P ≠ NP