Algorithm Design Exercise Sheet 2

Course of Engineering in Computer Science, Academic Year 2020/2021

Riccardo Salvalaggio 1750157 & Lorenzo Scarlino 1761016

January 10, 2021

1 Exercise 1.

1.1 Exercise 1a

$L_e = \sum_{h \in H_e} x_{eh}$ ∀ e∈E
L = $\max_{e \in E} L_e$
ILP Formulation:
min L
s.t.
1. $x_{eh} \in \{0,1\}$ ∀ e∈E,∀ h∈H_e ▷ x=1 if elf e visits house h, 0 otherwise
2. $\sum_h x_{eh} \leq$ L ∀ e∈E ▷ Sum of all houses visited by an elf isn't greater than makespan
3. $|\bigcup_{e \in E} \bigcup_{h \in H_e} x_{eh}| = |H|$, $x_{eh} = 1$ ▷ Cardinality of all houses visited by all elves is cardinality of H
4. $\sum_{e \in E} x_{eh} = 1$ ∀ h∈H ▷ Each house has to be visited from exactly one elf, once
Relaxed LP Formulation:
min L
s.t.
1. $x_{eh} \in [0,1]$ ∀ e∈E,∀ h∈H
2. $\sum_h x_{eh} \leq$ L ∀ e∈E
3. $|\bigcup_{e \in E} \bigcup_{h \in H_e} x_{eh}| = |H|$, $x_{eh} > 0$
4. $\sum_{e \in E} x_{eh} = 1$ ∀ h∈H

1.2 Exercise 1b

In the LP-relaxation optimal solution, all variables x_{e_h} take a fractional value between the range [0,1]: Each value can be seen as the probability to set that variable to 1, that is the probability with which elf e visits house h. Each house h has total expectation to appear in the scheduling $\mathbb{E} \sum_{e \in E} x_{eh} = \sum_{e \in E} \mathbb{E}[x_{eh}] = 1$, $\forall h \in H$. The algorithm randomly assigns each house to one of the candidate elves, taking into account its probability distribution (given by x_{eh}), and generates a feasible solution to Santa's problem. In fact, in this way all houses will be scheduled exactly once. Given OPT as the LP-optimum and |OPT| the number of houses visited by the busiest elf, the approximated solution returned by our algorithm, F, will necessarily have |F| ≤ |OPT|. Indeed, by treating variables in OPT as probabilities, what is going to happen in the integer approximated solution is that variables $x_{eh} = 0$ will have zero probability to appear in F, while the remaining ($x_{eh} > 0$) may or may not appear in F. So in the worst case, the busiest elf in F will be the same of OPT, with the same number of houses to visit, hence $|F| = |OPT|$.

1.3 Exercise 1c

As said, rounding LP obviously brings to achieve a worse result with respect to the LP optimum (Definition of relaxation), but we can prove that it stands inside a certain bound with Chernoff Bounds notion.
Chernoff bounds are based on the Markov's Inequality: $\forall a > 0$, $\Pr(X > a) \leq \frac{E(X)}{a}$.
The Theorem of the Chernoff Bounds for upper tails is the following: 'Let $X = \sum_{i=1}^{n} X_i$, where $X_i = 1$ with probability p_i and $X_i = 0$ with probability $1 - p_i$, and all X_i are independent. Let $\mu = E(X) = \sum_{i=1}^{n} p_i$. Then $\Pr(X > (1+\delta)\mu) \leq [\frac{e^\delta}{1+\delta^{1+\delta}}]^\mu$.'
In our case we assume $\mu = OPT$, and since $\mu > 0$ this probability is surely less or equal to $\frac{e^\delta}{1+\delta^{1+\delta}}$, and since it is proven to be $\leq e^{-\frac{\delta^2}{2+\delta}\mu}$; we can choose $\delta = O(\ln m)$ and so:

$$\Pr(X > (1+\delta)\mu) \leq \frac{1}{e^{\frac{\delta^2}{2+\delta}\mu}} \leq \frac{1}{e^{O(\ln m)}} = \frac{1}{m}$$

Finally, it is possible to rewrite: $$\Pr(X < (1+\delta)\mu) \geq 1 - \frac{1}{m}$$

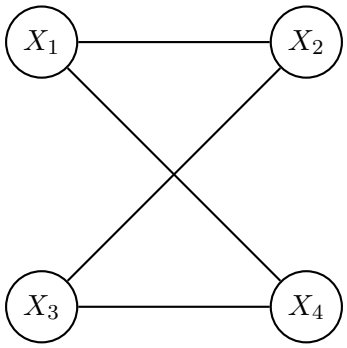
2 Exercise 2.

The problem of the construction of 187 different sleighs is a hard one, and in general it's not solvable in a day. A way to show it consists in proving that it's at least as hard as a well-known hard problem. A Boolean satisfiability formulation, or SAT, is the problem of determing if, given a CNF formula Φ, exists a satisfying assignment which returns true. SAT is NP-complete because it's both NP and NP-hard.
By reducing the SAT problem to the 187-sleighs problem, we can prove that our problem is at least as hard as SAT, hence an eventual poly-time solution for the sleighs problem would be suitable for SAT too.
Our problem takes in input n different pieces (each of them with lots of duplicates) and Santa's constraints, therefore we need to build a SAT formulation which can be transformed into 187-different-sleighs formulation in polynomial time.
For the construction of the first sleigh we can consider a SAT problem composed by the following clauses (all related by an AND operator):
1) An inclusive OR between all n pieces:$(X_1 \vee X_2 \vee .... \vee X_n)$
2,3,..,m) The use of AND,OR,NOT operators for combinations of pieces in order to describe Santa's constraints.
For the construction of the second sleigh, we can use the same formulation used in the previous case, plus we have to introduce the uniqueness constraint. Given a feasible combination for the first sleigh, to ensure that the next one is different, we have to include in the SAT formulation the clause: $\wedge\neg(X_i \wedge .... \wedge X_j)$, with $(X_i, ..., X_j)$ the pieces of the previous sleigh.
By going with this method, for every k-th sleigh, we have to formulate a SAT consisting of 1) and 2) to m) clauses about sleigh validation, plus k-1 clauses about uniqueness constraint.
In this way, the construction of the last sleigh will be represented by a SAT formulation composed by the first usual m clauses plus 186 uniqueness constraints, each of them generated by a previous SAT formulation.
On that basis we can finally prove that the 187-sleighs is a really hard problem, and if there exists a polynomial time algorithm that solves this one, that algorithm would solve the SAT problem too.

3 Exercise 3.

3.1 Exercise 3.1

A cut game Γ with four vertices abd unit weights, in which does exist a Pure Nash Equilibirum, is the following:



The set of equilibria for the minimum global payoff is given by the following CUT(α): $X_1 \leftarrow$ {LEFT}, $X_2 \leftarrow$ {LEFT},$X_3 \leftarrow$ {RIGHT}, $X_4 \leftarrow$ {RIGHT}.
Total payoff of CUT(α): U(α) $\leftarrow \sum_{i\in N}\sum_{e\in CUT(\alpha)\cap N_i} \omega_e = 4$.
The set for the maximum global payoff is given by the following CUT(α): $X_1 \leftarrow$ {LEFT}, $X_2 \leftarrow$ {RIGHT},$X_3 \leftarrow$ {LEFT}, $X_4 \leftarrow$ {RIGHT}.
Total payoff of CUT(α): U(α) $\leftarrow \sum_{i\in N}\sum_{e\in CUT(\alpha)\cap N_i} \omega_e = 8$.
The Price of Anarchy of game Γ is the quantity:

$$\text{PoA}(\Gamma) = \frac{max_{s\in S}\sum_{i\in N} u_i(s)}{min_{s\in E}\sum_{i\in N} u_i(s)} = \frac{8}{4} = 2.$$

3.2 Exercise 3.2

In a pure Nash equilibrium, each vertex v_i in the graph G has no interest in changing his status (from LEFT to RIGHT or viceversa), so the total weight of his neighbouring vertices in the cut (i.e. with different status from v_i) is at least equal to the total weight of neighbouring vertices which are not in the cut (i.e. with the same status of v_i). If this were not the case, the vertex v_i should have an advantage in changing his status, and we could not speak of PNE.
By expanding this concept to the entirety of the graph, in any cut game Γ, in a PNE, the global payoff (total weight in the cut) is at least equal to $2\cdot \frac{TotalWeigth(\Gamma)}{2}$ = TotalWeigth(Γ) (each edge appears twice in the calculation).
On the other hand, the maximum global payoff achievable in Γ is at most 2·TotalWeigth(Γ), that it means that every edge in the graph is in the cut (every vertex has only neighbours with a different status).
Thus, the Price of Anarchy of Γ is generally:

$$\text{PoA}(\Gamma) = \frac{max_{s\in S}\sum_{i\in N} u_i(s)}{min_{s\in E}\sum_{i\in N} u_i(s)} \leq \frac{2\cdot TotalWeight(\Gamma)}{TotalWeight(\Gamma)} \leq 2.$$

4 Exercise 4.

4.1 Exercise 4.1

The k-center selection, one of the most known NP-hard problems, can be polynomially reduced to our situation. In fact, given the k-center selection inputs, which consist in a set S of n cities s1,....sn, an integer k and all the possible points in the plane that contains all the cities, in polynomial time is possible to reduce them into our problem inputs: the set S and k remain the same, while the set of points in the plane is reduced to a set C of specific points in the plane (the possible antennae locations).
If there exists a polynomial time algorithm which solves our problem, it will also solve the k-center selection problem (by choosing the same antennae locations used in our problem's resolution).
Thus, our problem is at least as hard as the k-center selection (which is NP-hard), so it is possible to say that it's NP-hard.

4.2 Exercise 4.2

Since our problem is NP-hard, am approximation algorithm is the only way to obtain a feasible solution in polynomial time, even if it sacrifices the accuracy.
A deterministic implementation should check every possible City-Antenna combination in order to minimize the maximum distance from any City to the set of k Antennae. In this implementation we adopt a greedy approach by selecting for every iteration an Antenna x and all its adjacent Cities inside a range of $3 \cdot \min_{y \in Y} d(x,y)$, and assigning x to the Actual_Best_Solution set; then we iterate for every Antenna x'!= x and at each step we add x' to Actual_Best_Solution if |Actual_Best_Solution| ¡ k and if Improve(x') = maxImprove(every other antenna) (with Improve(x) number of cities that would be added selecting Antenna x).At the end of the algorithm every city is covered. Since for definition $d\{x,y\}>0$ (if x != y), $d\{x,y\}+d\{y,z\} \geq d\{x,z\} \forall$ x,y,z $\in$V and our bound choose of $3 \cdot \min\{d(x,y)\}$, every city will be inside a bound of 3 to its nearest Antenna.

4.3 Exercise 4.3