Algorithm Design
Second Homework

Students: Simone Petruzzi 1811872 , Domenico Tersigni 1817502

December 24 2022

1 Exercise 1

Before seeing the approximation for the vertex cover problem, let's recall it:
We have an undirected graph, having a set V of vertices and a set E of edges. The aim is to pick a subset S $\subseteq$ V of the vertices that includes at least one endpoint (vertex) of every edge, and we want the the cardinality $|S|$ to be the small as possible.

In order to show that any greedy algorithm gives a 2-approximation to the best vertex cover we have to remember the degree sum formula, it states that given a graph $G = (V, E)$:

$$\sum_{v \in V} deg(v) = 2\,|E| \tag{1}$$

By definition of vertex cover we need to cover all edges, and for each edge we pay a price at least equal to 1, this comes for the fact that in order to cover and edge we have to put one of its endpoints into the vertex cover set and it has a degree (so a weight) at least equal to 1. The best vertex cover has a price of at least $1 \cdot |E|$. The worst greedy algorithm we can build is an algorithm that cover all vertices but also in this case due to (1) we have that the price is $2\,|E|$. Then better greedy algorithms obviously select a smaller set of vertices with respect to the worst one, therefore the cost of these other greedy algorithms is for sure less or equal than $2\,|E|$.
Therefore we can conclude that any greedy algorithm will give a 2-approximation to the best vertex cover.

2 Exercise 2

A randomized strategy that guarantees that we select the card with the maximal value, among N arbitrary cards, with a probability of at least $\frac{1}{4}$ is the following:

- **BiggestSoFar** is the highest number seen so far.
- We turn over half the cards without stopping and then in the second half we stop when we see the first **BiggestSoFar**, i.e, a number bigger then the biggest number in the first half.

This strategy has probability of at least $\frac{1}{4}$ of success because it succedes whenever the largest number is in the second half, and the second largest number is in the first half. We define two events:

- **A** = The largest number is in the second half
- **B** = The second largest number is in the first half

Our strategy succedes if the event $A \cap B$ happens.

$$P(A \cap B) = P(A) \cdot P(B|A). \tag{2}$$

$$P(A) = \frac{N}{2}\frac{1}{N} = \frac{1}{2} \tag{3}$$

$$P(B|A) = \frac{N}{2}\frac{1}{N-1} > \frac{1}{2} \tag{4}$$

Due to (3) and (4): $P(A \cap B) > 1/4$.

We can prove the correctness of this strategy by claiming that the probabillity that the card with the maximal value is in the second half is equal to $\frac{1}{2}$. If we choose in the second half a number that is bigger then the biggest number of the first half, this value is greater or equal than $\frac{N}{2}$ values and for this reason it has a probability of at least $\frac{1}{2}$ to being the maximal value. Thus, our strategy has a probability of at least $\frac{1}{4}$ that we select the card with the maximal value.

3 Exercise 3

Given a graph $G = (V, E)$ and vertex weight w_i for each vertex $v_i \in V$, we can formulate the vertex cover problem as IP formulation by using variable x_i for each vertex taking vlaues 0,1 or 2 depending on how many times they are included in the set. We set 2 as upper bound since is useless adding a vertex more than two times, this provides no improvments since is simply adding cost to our final weight. The IP formulation is the following:

$$\begin{aligned} minimize \quad & \sum_{i=1}^{|V|} x_i w_i \\ \text{s.t.} \quad & 2x_i + x_j \geq 2 \quad \forall\, e = (x_i, x_j) \ \in \ E \\ & x_i \ \in \{0, 1, 2\} \quad x_i \ \in E \\ & x_j \ \in \{0, 1, 2\} \quad x_j \ \in E \end{aligned} \tag{5}$$

The $2x_i + x_j \geq 2$ constraint let us satisfy all cases:
The constraint is satisfied when either x_j is equal to 2 so that it is included at least 2 times in the multi-set S or when x_i is included at lest one time in S.
Since we know that IP is NP hard we can try relaxing the previous formulation obtaining a LP solution so that we can resolve it in polynomial time. We can do this just by removing the integrality constraints, so that variables can take values in $\mathbb{R}$, the formulation is the following one:

$$\begin{aligned} minimize \quad & \sum_{i=1}^{|V|} x_i w_i \\ \text{s.t.} \quad & 2x_i + x_j \geq 2 \qquad \forall e = (x_i, x_j) \ \in \ E \\ & 0 \leq x_i \leq 2 \qquad x_i \ \in \ E \\ & 0 \leq x_j \leq 2 \qquad x_j \ \in \ E \end{aligned} \tag{6}$$

Let OPT(LP*) be the optimal solution of LP problem and OPT(IP*) be the optimal solution to the IP problem. We know for sure that OPT(LP*) $\leq$ OPT(IP*) since it is a relaxation. After we solved the LP problem and got the optimal solution OPT(LP*), we can round every x_i^* of the LP solution to an integral x_i solution as follows:

$$\begin{cases} x_i = 0 \quad if \quad x_i^* < \frac{1}{2} \\ x_i = 1 \quad if \quad \frac{1}{2} \leq x_i^* \leq 1 \\ x_i = 2 \quad if \quad x_i^* > 1 \end{cases} \tag{7}$$

Now we want to prove that in the worst case we obtain a 2-approximation. This is given by the fact that in the worst case we round $x_i^* = \frac{1}{2}$ to 1 that is exactly the double of the weight, so $c(x_i) \leq 2c(x_i^*)$:

$$c(x_i) = \sum_{i=1}^{|V|} x_i w_i = \sum_{i \in |V| : x_i^* \geq \frac{1}{2}} w_i \quad \leq \sum_{i=1}^{|V|} 2x_i^* w_i = 2c(x_i^*) \tag{8}$$

So this rounding gets a 2-approximation.

4 Exercise 4

In a congestion game, strategies and payoffs are defined in terms of a finite set **R** of resources and in terms of congestion functions $c_r : R \times \{1, .., n\} \to \mathbb{R}$. Every player has to select one among different subset of resources: $\alpha_i \subseteq 2^R$ for every $\{1, .., n\}$. We define $\#(r, \alpha)$ the number of players using resource r in the strategy profile α. The cost experienced by player i is defined as: $cost_i(\alpha) := \sum_{r \in \alpha_i} c_r(\#(r, \alpha))$. Thus each player chooses a subset of the resources and the cost he pays is the total congestion of the resources he's choosing. The congestion of a resource only depends on the number of players choosing that resource.

Social Cost **(SC)** by definition is:

$$SC(s) := \sum_{i=1}^{n} cost_i(\alpha) = \sum_{i=1}^{n} \sum_{r \in \alpha_i} c_r(\#(r, \alpha)) = \sum_{r \in R} \sum_{i=1}^{\#(r,\alpha)} c_r(\#(r, \alpha)) \tag{9}$$

.

Now we point out that the Rosenthal's potential function $\Phi(\alpha)$ is different from SC(s), infact:

$$\Phi(\alpha) = \sum_{r \in R} \sum_{i=1}^{\#(r,\alpha)} c_r(i)) \leq \sum_{r \in R} \sum_{i=1}^{\#(r,\alpha)} c_r(\#(r, \alpha)) = SC(s) \tag{10}$$

Due to the fact that c_r is non decreasing, infact if c_r was decreasing then $\Phi(\alpha) \geq SC(s)$.

Now we know that no resource is ever used by more than λ players. This means that $\#(r, \alpha)) \leq \lambda$.

Then we can rewrite:

$$SC(\alpha) = \sum_{r \in R} \#(r, \alpha) \cdot c_r(\#(r, \alpha)) \leq \sum_{r \in R} \lambda \cdot c_r(\#(r, \alpha)) \leq \lambda \cdot \sum_{r \in R} \sum_{i=1}^{\#(r,\alpha)} c_r(i)) = \lambda \cdot \Phi(\alpha) \tag{11}$$

Thus we have:

$$SC(s) \leq \lambda \cdot \Phi(\alpha) \tag{12}$$

And by multiplying both members of (10) by λ we obtain:

$$\lambda \cdot \Phi(\alpha) \leq \lambda \cdot SC(s) \tag{13}$$

Now we have two possibilities, if the state with the optimal social cost s_{OPT} is a NE state, by definition we obtain PoS = 1. In this case is trivially satisfied the condition that $PoS \leq \lambda$ because λ is at least 1, due to the fact that *no resources is ever used by more than 0 players* has no sense. In general the social optimum state is not necessarily a NE state. If s_{OPT} is not a NE state, it means that in s_{OPT} there is at least one player i who can reduce his own cost by changing his strategy from α_i to $\alpha_i^{'}$ obtaining $cost_i^{'} < cost_i$. Due to this fact and by the definition of potential function we know that when a single player i changes his strategy, the change in potential function equals to the change in the cost to i: $cost_i^{'} - cost_i = \Phi(\alpha') - \Phi(\alpha)$ and then we obtain

$$\Phi(\alpha') = \Phi(\alpha) + cost_i^{'} - cost_i \tag{14}$$

$cost_i^{'} - cost_i$ is a negative quantity and it means that $\Phi(\alpha')$ is decreasing. In contrast the change in the SC equals the total change in cost to all players. Hence the SC isn't often a valid potential function.

Starting from an s_{OPT} which is not a NE state, if there is a player (or more than one) that can improve its strategy we allow that. In this way the Rosenthal's potential function always decrease. $\Phi(\alpha)$ has a minimum, because a congestion game always has at least one PNE. Finding a PNE in a congestion game is equivalent to finding a local minimum of the potential function. Any minimum of $\Phi(\alpha)$ is a PNE. Since there are a finite number of configurations and each c_r is monotone there exists an equilibrium. When we reach a minimum it is a NE state, and we call it $\hat{s}$ and the corresponding strategy profile $\hat{\alpha}$.

For what we have said before $\Phi(\hat{\alpha}) \leq \Phi(\alpha_{s_{OPT}})$ and from (12) and (13) we deduct: $SC(\hat{s}) \leq \lambda \cdot \Phi(\hat{\alpha}) \leq \lambda \cdot \Phi(\alpha_{s_{OPT}}) \leq \lambda \cdot SC(s_{OPT})$

From that we obtain:

$$\frac{SC(\hat{s})}{SC(s_{OPT})} \leq \lambda \tag{15}$$

But we have no guarantee about the fact that $\hat{s}$ is the NE state with the minimum social cost, but the NE state with the minimum social cost has a social cost less or equal than $\hat{s}$, and for this fact the relation (15) holds even with the NE state with the minimum social cost and we have terminated our proof.

5 Exercise 5

Before discussing the approximation of the proposed algorithm, let's prove that it's outputting an independent set. The given greedy algorithm takes the first vertex from a uniform vertex random permutation of a given undirected graph G=(V,E) and adds it to S. Then iteratively the algorithm scans all the vertices of the uniform random permutation and checks if for all the vertices already added to S, there is some edge that connects the current vertex to vertices already in S, if there aren't, it adds the current vertex to S. Since with this procedure the algorithm does not place any neighbor of any vertex that is placed in S, it outputs for sure an independent set. Let's now discuss how far it goes from the optimal solution: we want to achieve at least $\frac{1}{d}$ of the optimal solution where d is the maximum degree of a vertex of V.
Let T be the set containing nodes discarded by the algorithm (these are the neighbors of nodes added in S). We can give an upper bound for these nodes saying, for each vertex $t \in T$ we make an association to a vertex $s \in S$. A vertex in S can be associated at most d times since it has at most d neighbors. Hence we have that $|T| \leq d|S|$.
Now let's say OPT is a maximal independent set for the graph G. We can associate each vertex $o \in OPT \backslash S$ to a vertex $s \in S \backslash OPT$ which is a neighbor of o and for the same reasoning done above the number of nodes associated to a node $s \in S \backslash OPT$ is at most d, thus (remembering that given two sets A and B: $|A \backslash B| = |A| - |B|$):

$$|OPT \backslash S| \leq d|S \backslash OPT| \implies |S| \geq \frac{|OPT|}{d} \tag{16}$$