Algorithm Design - Homework 2

Francesco Carpineti 1683418 - Federico Gioia 1702089 - Giuliano Abruzzo 1712313

January 15, 2019

Contents

1 Exercise 1

We have to see the problem as a graph, where each node represents a person, labeled with M or F. The friendship function $w(x, y)$ is represented with an edge if and only if the corresponding value is 1, otherwise it will be 0 (no edge). We want to use this algorithm to calculate the densest subgraph:

Algorithm 1 Densest-Subgraph($G = (V, E)$)

$n \leftarrow |V|$, $R_n \leftarrow G$
for $i = \frac{n}{2}$ to 2 **do**
  $m \leftarrow$ M node in R_i of min degree
  $f \leftarrow$ F node in R_i of min degree
  $R_{i-1} \leftarrow R_i - \{m, f\}$
return (R_j, which has the maximum density among all R_i, $i = 1, 2, \ldots, n$)

Since we have a number of iterations that is at most $\frac{n}{2}$, and in each iteration we compute the nodes m and f with lower degree, computing the degree has a cost of $\mathcal{O}(|V| + |E|)$, we have to do this for all the vertices in the graph so it becomes $\mathcal{O}(|V|^2 + |V||E|)$, multiplied for the number of iterations it comes out with a cost of $\mathcal{O}(|V|^3 + |V|^2|E|)$.

Lemma 1 *In an optimal solution* $S^* = (E^*, V^*)$, $\forall m \in M^*$, $f \in F^*$ $deg(m) + deg(f) \geq 2\lambda = 2\frac{|E^*|}{|V^*|}$

Proof: let $|E^*| = x$, $|V^*| = y$, $deg(m) + deg(f) = z$. Due to the optimality of solution, if we remove any couple of nodes m and f we have that

$$\frac{x}{y} \geq \frac{x - z}{y - 2} \tag{1}$$

and from it we find:

$$z \geq 2\frac{x}{y} \tag{2}$$

Theorem 2 *The greedy algorithm* ***Densest-Subgraph*** *achieves a 2-approximation for the densest subgraph problem in undirected networks.*

From the previous lemma, if x is the number of (m, f) couples in optimal solution S^*, then

$$\sum_{\forall (m^*, f^*) \in V^*} deg(m^*) + deg(f^*) \geq 2x\lambda^* = |V^*|\lambda^* \tag{3}$$

Consider the situation in which, in our algorithm's execution, we are going to remove, **for the first time**, a *m'* node and a *f'* node which are also in the optimal solution S* . Then, since *m'* and *f'* are the nodes with the less grade in our graph at that moment (they are about to be removed) and both of them $\in S^*$:

$$\forall (m', f') \in V', deg(m') + deg(f') \geq deg(m^*) + deg(f^*)$$

considering that *V'* is the set of vertices of our graph in that specific iteration. Then, for the Lemma:

$$deg(m^*) + deg(f^*) \geq 2\lambda^*$$

So, for the transitive property, and summing all the pairs of nodes:

$$\sum_{\forall v \in V'} deg(v) \geq 2x'\lambda^*$$

where x' is the number of pairs of our solution.

Knowing that

$$\sum_{\forall v \in V'} deg(v) = 2|E'| \tag{4}$$

we know that 2|E'| $\geq$ |V'|λ^*, i.e. $\lambda \geq \frac{1}{2}\ \lambda^*$. For any iteration of our algorithm, we will have that our density $\lambda \leq \lambda^*$, since we never reach the optimal solution (we are using an approximation algorithm). We can write, at the end, that:

$$\frac{1}{2}\lambda^* \leq \lambda \leq \lambda^*$$

expression which prove that our algorithm is a 2-approximation algorithm for this problem. Lemma 1 and Theorem 2 have been taken from densest subgraph problem applied to graphs with generic nodes[1] and modified for our purposes.

[1] `https://people.cs.umass.edu/~barna/paper/icalp-final.pdf`

2 Exercise 2

This is a modified version of set cover problem. We are given:

- U: universe of n elements e_1,...,e_n, skills of the project;
- Z = S_1,...,S_m, collection of m subsets of U, which represent persons containing some skills;
- $c(S_i)$, cost function for S_i.

The problem goal is to find a set of min-cost subsets such that their union is U and in which every skill is present at least three times. So, we have the following Primal:

$$
\begin{aligned}
& min \sum_{S \in Z} c(S)x(S) \\
& s.t. \sum_{S:e \in S} x(S) \geq 3, \forall e \in U \\
& 0 \leq x(S) \leq 1, S \in Z
\end{aligned}
$$

And then, the Dual:

$$
\begin{aligned}
& max \sum_{e \in U} 3y(e) \\
& s.t. \sum_{e \in S} y(e) \leq c(S), \forall e \in U \\
& y(e) \geq 0, \forall e \in U
\end{aligned}
$$

Now, we discuss the algorithm for this modified version of Set Cover:

Algorithm 2 Modified Set-Cover Algorithm

$x_S = 0$ for all S $\in$ Z
$y_e = 0$ for all e $\in$ U
while $\exists$ e uncovered in U **do**
    repeat y_e++ until $\sum_{e \in S} 3y_e = c(S)$ for some S $\in$ Z
    for all such sets S **do**
        $x_S = 1$
        for e' in S **do**
            set it as covered
return x_S, $\forall$ S $\in$ Z

We have to apply the Randomized Rounding to this modified version of the algorithm. These are the probabilities we need:

- Pr[e is not covered] = $\prod_{i=1}^{k}$ (1-p(S_i)) $\leq$ (1 - $\frac{1}{k}$)^k $\leq$ $\frac{1}{e}$
- Pr[e is once covered] $\leq$ $\frac{k(1-\frac{1}{k})^{k-1}}{k}$ $\leq$ $\frac{1}{e(1-\frac{1}{k})}$

- Pr[e is twice covered] $\leq \frac{\frac{k(k-1)}{2}(1-\frac{1}{k})^{\text{k-2}}}{k^2} \leq \frac{1}{2e(1-\frac{1}{k})}$

- Pr[e is at least three times covered] $\geq$ 1- $\frac{5}{2e}$

The last probability is obtained subtracting all the first three ones from 1. expected cost of randomized rounding is good but we haven't guaranteed that all elements are covered, in fact there is a good chance that all the elements aren't covered. Anyway, any element is covered with a probability of at least non-zero. So, an improvement to the randomized rounding is to repeat it for t times, until we reach a good value. This is done as follows[2]:
X = $\bigcup_{i=1}^{t}$ $\mathrm{X_i}$,
where $\mathrm{X_i}$ is the result of the i-th randomized rounding. The improved randomized rounding has:

- E[C(X)] $\leq$ t*OPT;

- Pr[an e is covered by at most two sets] $\leq \frac{5}{2e}^{\text{t}}$;

- Pr[some e is covered by at most two sets] $\leq n\frac{5}{2e}^{\text{t}}$

where n is the number of elements in U.
Now, given a treshold ϵ, we have to compute how many rounds of randomized rounding are needed in order to assure that the probability that some element is not covered is lower than ϵ:
$n\frac{5}{2e}^{\text{t}} \leq \epsilon \rightarrow t \geq \frac{\ln\frac{n}{\epsilon}}{\ln\frac{2}{5}+1}$
So, a solution that returns X = $\bigcup_{i=1}^{t}$ $\mathrm{X_i}$, satisfies E[C(X)] $\leq$ ($\frac{\ln\frac{n}{\epsilon}}{\ln\frac{2}{5}+1}$)*OPT and Pr[X isn't a valid solution] $\leq \epsilon$. From Markov's inequality: Pr[Y $\geq \alpha$] $\leq \frac{E[Y]}{\alpha}$, we obtain Pr[C(X) $\geq$ 4OPT*($\frac{\ln\frac{n}{\epsilon}}{\ln\frac{2}{5}+1}$)] $\leq$ ($\frac{E[C(X)]}{4*OPT*\frac{\ln n/\epsilon}{\ln\frac{2}{5}+1}}$) $\leq \frac{1}{4}$.
If we set $\epsilon = 1/4$, we obtain:

$$Pr\left[Xvalid - and - C(X) \leq 4 * OPT * \frac{\ln 4n}{\ln 2/5 + 1}\right] \geq \frac{1}{2}$$

and the corresponding algorithm is:

1. Solve LP;
2. Repeat randomized rounding for $\frac{\ln 4n}{\ln 2/5+1}$ times and get a solution X = $\bigcup_{i=1}^{t}\mathrm{X_i}$
3. if C(X) $\geq$ 4*OPT*$\frac{\ln 4n}{\ln 2/5+1}$ or X is not valid, go back to step 2, else return X as final solution.

[2] `http://www.ccs.neu.edu/home/rraj/Courses/7880/F09/Lectures/LP.pdf`

3 Exercise 3

This is an example of well known multiway-cut problem[3]. When we have a number of vertices k≥3, we need to run k times the algorithm for min-cut problem. Assuming as source a vertex s_i, with i=1,...,k, and the other nodes of our k-sized set collapsed to a single node t_i (sink), we will generate k min-cuts as output:

$$C = \bigcup_{i=1}^{k} C_i$$

Now we need to prove that this kind of solution is a 2-approximated solution of the optimal one, i.e. |C| ≤ 2*OPT. Let's suppose that F^* is the optimal solution cut and A_i is a cut which takes as source s_i mentioned above. Since each edge in F^* is adjacent to 2 vertices, each edge must be into two different cuts A_i. Hence:

$$\sum_{i=1}^{k} |A_i| = 2|F^*|$$

where the module represents how many edges are in that cut. Considering the min-cuts mentioned at the beginning, we can say that $|C_i| \leq |A_i|$, and hence their sums too, we can say that:

$$\sum_{i=1}^{k} |C_i| \leq \sum_{i=1}^{k} |A_i| = 2|F^*|$$

So, $|C| \leq 2^*|F^*| \rightarrow$ |C| ≤ 2*OPT. and this proves that our solution is a 2-approximated one w.r.t. the optimal one.

[3] `https://cs.stackexchange.com/questions/7205/2opt-approximation-algorithm-for-multiway-cut-problem`

4 Exercise 5

As we can see, the game is pretty unfair w.r.t. Dasher:

$\boldsymbol{u_1,u_2}$	**Head**	**Tail**
Head	+4,-4	-1,+1
Tail	-2,+2	+2,-2

where rows are referred to Comet and columns to Dasher. So, we have to bias the coins in order to make the game fair for both the reindeers. This is a *zero-sum* and *finite* game, so we can use the procedure seen in classes for this particular class of problems. We write this set of equations:

$$\begin{aligned} 4x_1y_1 - x_1y_2 - 2x_2y_1 + 2x_2y_2 &= -4y_1x_1 + 2y_1x_2 + y_2x_1 - 2y_2x_2 \\ x_1 + x_2 &= 1 \\ y_1 + y_2 &= 1 \end{aligned}$$

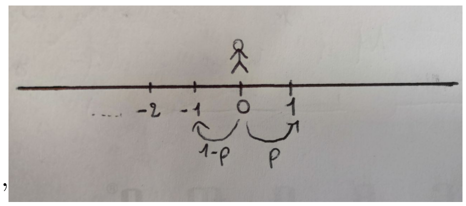
where x_1 and x_2 are Head and Tail for Comet's coin and y_1 and y_2 are Head and Tail for Dasher's coin. In the first equation we say that both the expected values of the two reindeers in the game are equal. In fact, this condition is essential for making the game fair for both the players. The expressions written above are parametric ones, because there are infinite solutions. So we have to assign some values to x_1 and x_2 and then we calculate y_1 and y_2 or viceversa. If we substitute $y_1 = 0$ and $y_2 = 1$, we obtain $x_1 = 2/3$ and $x_2 = 1/3$ and these are the values with whom we have to bias the two coins.

5 Exercise 6 (bonus)

This problem is an example of random walk[4] on the integer line. Giorgio starts from 0 and does a step towards the highway with probability p; obviously, the probability that Giorgio does a step away from the highway will be *1-p*. We know that the highway is on position -1. We have to find values of p in a way such that:

- Giorgio reaches the highway with probability at most 1/2;
- Giorgio reaches the highway for sure (with probability of 1).

In order to use a formula that we will explain later, we will use this equivalent problem:

”

So, the highway will be on position 1 and Giorgio will do a step towards it with probability p. Position 1 is called *absorbing barrier* of the problem. Let's define the following variable: T_r = first passing time on position r. Then, the probability of visiting r, sooner or later, is defined using a function *f(P)* as follows:

$$f(P) = P(Tr < \infty) = \sum_{n=0}^{\infty} P(T_{\mathrm{r}} = n)$$

Now we have to use a generating function, the formula we cited above:

$$G(s) = \sum_{n=0}^{\infty} s^n P(T_{\mathrm{r}} = n)$$

For a theorem[5], we know that for r=1 (so, our study case, position 1) the generating function expression is this:

$$G(s) = \left[\frac{1 - \sqrt{1 - 4p(1-p)s^2}}{2(1-p)s} \right]^r$$

we set s=1, so:

$$G(1) = f(P) = \frac{1 - \sqrt{1 - 4p(1-p)}}{2(1-p)}$$

Now, obtained the function *f(P)* we wanted to find, we have to find p for both the two cases mentioned at the beginning of the paragraph, i.e. *f(P)* $\leq \frac{1}{2}$ and *f(P)* $= 1$. Solving the two

[4] `https://en.wikipedia.org/wiki/Random_walk`

[5] `http://home.ustc.edu.cn/~zt001062/PTmaterials/Grimmett&Stirzaker--Probability%20and%20Random%20Processes%20%20Third%20Ed(2001).pdf`, page 164

expressions, we will find, for the first case, p$\leq \frac{1}{3}$ and ,for the second one, p$\geq \frac{1}{2}$. This last result is important because in a balanced random walk, the probability of reaching position r is 1, and if we augment the p value for the second case, we will increase the drift to the right for Giorgio (so, a confirmation of our result).