

Homework 2

Algorithm Design 2018-19 - Sapienza

Luigi Russo 1699981

January 9, 2019

Contents

1	Michele's birthday	3
2	Valerio and Set Cover	4
3	The "k min-cut" problem	6
4	Cristina and DNA	7
5	Comet and Dasher	8
6	Drunk Giorgio	9

1 Michele's birthday

We can represent each friend as a node in an undirected graph G : the total number of nodes is n . We also add an edge between two nodes x and y iff $w(x,y)$ is equal to 1. In this case the function we want to maximize is exactly the density [1] of a subgraph of G . The following algorithm achieves a 2-approximation:

```

1  $S_n \leftarrow G$ 
2  $i \leftarrow \frac{n}{2}$ 
3 while  $i > 0$  do
4     //minimum degree nodes
5      $m \leftarrow x : \deg(x) \leq y, \forall x, y \in S_i \cap M$ 
6      $f \leftarrow x : \deg(x) \leq y, \forall x, y \in S_i \cap F$ 
7
8     //new candidate solution
9      $S_{i-1} \leftarrow S_i - \{m, f\}$ 
10 return  $\max S_i, \forall i \in \{1, 2, \dots, \frac{n}{2}\}$ 

```

Let OPT be the optimal solution and its density $d_{OPT} = \frac{\sum_{e \in OPT} w(e)}{|OPT|} = \frac{W}{N}$. $\forall m, f : m \in M \cap OPT, f \in F \cap OPT$, we have that the sum of their degrees is at least twice as the density of OPT : in fact, $\frac{W}{N} \geq \frac{W - \deg(m) - \deg(f)}{N-2} \Rightarrow \deg(m) + \deg(f) \geq 2 \frac{W}{N}$. Lets now consider the i -th iteration of the algorithm in which the first $m \in OPT$ or $f \in OPT$ is removed. In S_i we have that for every possible pair (m, f) the sum of their degrees is at least $2 \cdot d_{OPT}$. We have that $W_{S_{i-1}}$ (i.e. the weight of the graph S_{i-1}) is at least equal to $\frac{2 \cdot d_{OPT} \cdot \frac{S_{i-1}}{2}}{2}$ (it could be the case that some edge is considered twice, for this we divide by 2). The density of S_{i-1} is then at least equal to $\frac{W_{S_{i-1}}}{|S_{i-1}|} = \frac{d_{OPT}}{2}$. The algorithm selects the graph with maximum density among the $\frac{n}{2}$ built, so the density of the solution is at least equal to the one of S_{i-1} and this proves the 2-approximation factor.

Running time

As for the running time, the algorithm makes $O(|V|)$ iterations, and searches for two nodes with minimum degree at each iteration; then it builds a new graph starting from the previous and removing two nodes and "their" edges. This can be achieved in $O(|V| \cdot (|V| + |E|))$ using lists of adjacency. To compute the density of a graph we need $O(|V| + |E|)$ if we want to count the number of edges and nodes in it. The total cost is $O(|V| \cdot (|V| + |E|))$

2 Valerio and Set Cover

Given a set A of required skills, a set S of all the available people, where each person is represented as a set of skills $S_j \subseteq A$, we can formulate the Set Cover with Redundancies problem using the following ILP:

$$\begin{aligned} \min \quad & \sum_{S_j \in S} c(S_j) \cdot x_j \\ \sum_{S_j | A_i \in S_j} x_j & \geq 3, & \forall A_i \in A \\ x_j & \in \{0, 1\}, & \forall S_j \in S \end{aligned}$$

I am going to show a variant of the randomized rounding applied to set cover, starting from [2]. In order to build a randomized approximation consider the associated LP problem where $x_j^* \in [0, 1]$. The LP solution is a vector x^* of real values. For each set $S_j \in S$, pick S_j with probability x_j^{*1} , the entry corresponding to S_j in x^* . Let C be the collection of sets picked. The expected cost of C is

$$E[c(C)] = \sum_{S_j \in S} Pr[S_j \text{ is picked}] \cdot c(S_j) = \sum_{S_j \in S} x_j^* \cdot c(S_j) = OPT_f.$$

Next, let us compute the probability that a skill $a \in U$ is covered at least 3 times by C . Suppose that a occurs in $k \geq 3$ (otherwise the problem has no solution) sets of S . Let the probabilities associated with these sets be p_1, \dots, p_k . Since a is fractionally covered in the optimal solution, $\sum_{i=1}^k p_i \geq 3$. The probability that a is covered by C is minimized when each of the p_i is equal to $\frac{3}{k}$. Thus,

$$Pr[a \text{ is covered}] \geq 1 - \sum_{i=0}^2 \binom{k}{i} \left(\frac{3}{k}\right)^i \left(1 - \frac{3}{k}\right)^{k-i} = 1 - \left(1 - \frac{3}{k}\right)^k - 3 \cdot \left(1 - \frac{3}{k}\right)^{k-1} \cdot \frac{9}{2} \cdot \left(1 - \frac{3}{k}\right)^{k-2}$$

and we can bound this:

$$Pr[a \text{ is covered}] \geq 1 - e^{-3} - 3e^{-3} - \frac{9}{2}e^{-3} \geq 1 - e^{-\frac{5}{6}}$$

To get a complete set cover with the redundancies, independently pick $\frac{6}{5}d \log n$ such subcollections, and compute their union, say C' , where d is a constant such that: $(e^{-\frac{5}{6}})^{\frac{6}{5}d \log n} \leq \frac{1}{4n}$. Clearly we have that:

$$Pr[a \text{ is not covered}] \leq \frac{1}{4n}$$

¹to be more precise $\min(x_j^*, 1)$

Summing up all skills a :

$$Pr[C' \text{ is not a valid solution}] \leq n \cdot \frac{1}{4n} = \frac{1}{4}$$

Clearly we have that:

$$E[c(C')] \leq \frac{6}{5} \cdot OPT_f \cdot d \log n$$

For Markov we can write:

$$Pr[c(C') \geq OPT_f \cdot 4 \cdot \frac{6}{5} \log n] \leq \frac{1}{4}$$

and implies that

$$Pr[C' \text{ is valid and has cost } \leq OPT_f \cdot 4 \cdot \frac{6}{5}] \geq \frac{1}{2}$$

If the above procedure fails to find a *good*² solution, we can repeat the entire process one more time. The expected number of repetitions is at most 2.

²where good means that it is a valid cover with redundancies and its cost is bounded by the expression defined above

3 The "k min-cut" problem

Let F^* be an optimal solution for the problem and let F_i^* be the isolating cut in the optimal solution for s_i . Since F_i is a minimum cut for s_i ,

$$\sum_{e \in F_i} c_e \leq \sum_{e \in F_i^*} c_e$$

The cost of our solution is at most

$$\sum_{i=1}^k \sum_{e \in F_i} c_e \leq \sum_{i=1}^k \sum_{e \in F_i^*} c_e$$

Since each edge in an optimal solution F^* can be present in at most 2 different F_i^* , we have that our solution is bounded by:

$$\sum_{i=1}^k \sum_{e \in F_i} c_e \leq \sum_{i=1}^k \sum_{e \in F_i^*} c_e \leq 2 \cdot \sum_{e \in F^*} c_e \leq 2 \cdot OPT$$

and this shows the 2-approximation.

4 Cristina and DNA

Let define a factorization f of the string D as an ordered multiset $\{g_1, g_2, \dots, g_p\}$ with $g_i \in G$, such that the concatenation of g_1, \dots, g_p produces the string D . Let F be the set of all possible factorizations of D and $F_g := \{f \in F : g \in f_g\}$ be the set of all factorizations that contain the gene g . We need $|G|$ boolean variables x_g , set to 1 if gene $g \in G$ is used to produce D and $|F|$ boolean variables y_f , indicating whether the factorization $f \in F$ is used or not. The ILP formulation (see [3]) and its dual are:

$$\begin{array}{ll}
 \min & \sum_{g \in G} w_g \cdot x_g \\
 \text{s. t.} & \sum_{f \in F_g} y_f \leq x_g, \quad \forall g \in G \\
 & \sum_{f \in F} y_f \geq 1 \\
 & x_g, y_f \in \{0, 1\}, \quad \forall g \in G, f \in F
 \end{array}
 \quad (*) \quad
 \begin{array}{ll}
 \max & b \\
 \text{s. t.} & \sum_{g \in f} a_g \geq b, \quad \forall f \in F \\
 & a_g \leq w_g \quad \forall g \in G \\
 & a_g, b \geq 0, \quad \forall g \in G
 \end{array}$$

where in the primal problem the first $|G|$ inequalities guarantee that a gene g is bought iff a factorization with it is picked and the second inequality states that at least one factorization has to be chosen. The dual (*) can be obtained with Lagrangian, after relaxing the original problem to LP: first of all the variables x and y are no longer boolean, but have to be ≥ 0 . Then we compute

$$L(x, y, a, b, c, d) = \sum_{g \in G} x_g(w_g - a_g - c_g) - \sum_{f \in F} y_f(b + d_f) + \sum_{f \in F_g} y_f \sum_{g \in G} a_g + b$$

and after creating the dual (d_1), and then removing both d_f (d_2) and c_g variables, we obtain the formulation (*) presented above.

$$\begin{array}{ll}
 (d_1) \quad \max & b \\
 \text{s. t.} & w_g - a_g - c_g = 0, \quad \forall g \in G \\
 & b + d_f = 0, \quad \forall f \in F \\
 & \sum_{g \in G} a_g = 0, \quad \forall g \in G \\
 & b, a_g, c_g, d_f \geq 0, \quad \forall g \in G, f \in F
 \end{array}
 \quad (d_2) \quad
 \begin{array}{ll}
 \max & b \\
 \text{s. t.} & w_g - a_g - c_g = 0, \quad \forall g \in G \\
 & b \leq \sum_{g \in f} a_g, \quad \forall f \in F \\
 & b, a_g, c_g \geq 0, \quad \forall g \in G, f \in F
 \end{array}$$

5 Comet and Dasher

The problem can be formalized with the following payouts matrix:

	T_C, T_D	T_C, H_D	H_C, T_D	H_C, H_D
Comet	2	-2	-1	4
Dasher	-2	2	1	-4

Let's now define:

- $h_X = Pr[\text{Head}]$ for player X
- $t_X = Pr[\text{Tail}]$ for player X

We can easily find that:

- $Pr[T_C, T_D] = t_C \cdot t_D$
- $Pr[T_C, H_D] = t_C \cdot h_D$
- $Pr[H_C, T_D] = h_C \cdot t_D$
- $Pr[H_C, H_D] = h_C \cdot h_D$

To guarantee that the game is fair, the expected value of Comet must be equal to the one of Dasher:

$$-2t_C t_D - t_C h_D + 2h_C t_D + 4h_C h_D = 2t_C t_D t_C h_D + -2h_C t_D + -4h_C h_D$$

with $t_C + t_D = 1$ and $h_C + h_D = 1$ since they are probability functions. Resolving the system we obtain

$$9h_C h_D - 3h_C - 4h_D + 2 = 0$$

There are infinite solutions: simple solutions are

- $t_C = 1, h_C = 0, t_D = h_D = 0.5$
- $t_D = 1, h_D = 0, h_C = \frac{2}{3}, t_C = \frac{1}{3}$

6 Drunk Giorgio

Let $P_n = Pr(Home|start = n)$ be the probability Giorgio goes back to home starting from position n and let $q = 1 - p$ the probability to make a step towards home. Let N be the distance from home (Giorgio starts at 0).

$$P_n = \begin{cases} 0, & \text{if } n = -1 \\ p \cdot P_{n-1} + q \cdot P_{n+1}, & \text{if } 0 \leq n < N \\ 1, & \text{if } n = N \end{cases}$$

We can rewrite P_n in this way: $P_n = p \cdot P_{n-1} + q \cdot P_{n+1} \Rightarrow P_{n+1} - P_n = \frac{p}{q} \cdot (P_n - P_{n-1})$.

In particular $P_1 - P_0 = \frac{p}{q} \cdot P_0$; moreover $P_2 - P_1 = (\frac{p}{q})^2 \cdot P_0$. In general we have: $P_{n+1} - P_0 = \sum_{k=0}^n (P_{k+1} - P_k) = \sum_{k=0}^n ((\frac{p}{q})^{k+1} \cdot P_0) = \sum_{k=1}^{n+1} ((\frac{p}{q})^k \cdot P_0) \Rightarrow P_{n+1} = P_0 + \sum_{k=1}^{n+1} ((\frac{p}{q})^k \cdot P_0) = P_0 \sum_{k=0}^{n+1} (\frac{p}{q})^k$

$$P_{n+1} = \begin{cases} P_0(n+2), & \text{if } p = q = 0.5 \\ P_0(\frac{1-(\frac{p}{q})^{n+2}}{1-\frac{p}{q}}), & \text{if } p \neq q \end{cases}$$

For $n = N - 1$:

$$1 = P_N = \begin{cases} P_0(N+1), & \text{if } p = q = 0.5 \\ P_0(\frac{1-(\frac{p}{q})^{N+1}}{1-\frac{p}{q}}), & \text{if } p \neq q \end{cases}$$

$$P_0 = \begin{cases} \frac{1}{N+1}, & \text{if } p = q = 0.5 \\ \frac{1-\frac{p}{q}}{1-(\frac{p}{q})^{N+1}}, & \text{if } p \neq q \end{cases}$$

The probability to go to hospital starting from 0 is:

$$Pr(Hospital|start = 0) = 1 - \lim_{N \rightarrow +\infty} P_0 = \begin{cases} 1, & \text{if } p \geq q \\ \frac{p}{q}, & \text{if } p < q \end{cases}$$

For $p \geq q$, Giorgio always goes (probability = 1) to hospital, instead for $0 \leq p < \frac{1}{3}$, Giorgio goes to hospital with probability less than 0.5 (easily obtained by solving the inequality above!).

References

- [1] *Wikipedia: Dense Subgraph*
https://en.wikipedia.org/wiki/Dense_subgraph
- [2] V. Vazirani. *Approximations algorithms*
- [3] D. Hermelin, D. Rawitz, R. Rizzi, S. Vialette. *The minimum substring cover problem*
- [4] *Wikipedia: Gambler's ruin*
https://en.wikipedia.org/wiki/Gambler%27s_ruin