Parcial AA

Pere Carrillo, Paula Grau, Marc Ordoñez, Marc Valls,
Marc Delgado (cabrons), neuwus, Jordi, Haonan i Adrià

Problema 8: Max clique problem - Marc Delgado

Enunciat

Consider the **Max Clique** problem. Given an undirected graph $G = (V, E)$ compute a set of vertices that induce a complete subgraph with maximum size.

For each $k \geq 1$, define G^k to be the undirected graph (V^k, E^k), where V^k is the set of all ordered k-tuples of vertices from V . E^k is defined so that $(v_1, \ldots, v_k)$ is adjacent to $(u_1, \ldots, u_k)$ iff, for $1 \leq i \leq k$, either $(v_i, u_i) \in E$ or $v_i = u_i$.

(a) Prove that the size of a maximum size clique in G^k is the k-th power of the corresponding size in G.

(b) Argue that if there is a constant approximation algorithm for Max Clique, then there is a polynomial time approximation schema for the problem.

https://es.overleaf.com/project/646890ae48ac072a79807e51

Solució

(a) Consider a graph $G = (V, E)$ where its Max Clique is of size 2 (an edge $e = (a, b) \in E$ connecting a pair of vertices $a, b \in V$). In G^2, all vertices consisting of combinations of a and b will be connected, since for any two vertices $x = (x_1, x_2), y = (y_1, y_2) \in G^2$, either $x_j = y_j$ or $x_j \neq y_j \implies$ one is a and the other one is b, and since $\exists (a, b) \in E$, that implies that $\exists (x, y) \in E^2$ $\forall x, y \in V^2$ such that $x = (x_1, x_2), y = (y_1, y_2)$ and $\forall j \in \{1, 2\}$ $x_j, y_j \in \{a, b\}$.

Extend this reasoning for a clique of size n in G. Suppose all vertices that form the clique in G are $C = \{a_1, a_2, \ldots, a_n\}$ and thus, $\forall i, j \in [1, n], i, j \in \mathbb{Z}, i \neq j$, $\exists (a_i, a_j) \in E$. In G^2, these n nodes will generate n^2 nodes of the form:

$$C^2 = \{a_1a_1, a_1a_2, \ldots, a_1a_n, a_2a_1, a_2a_2, \ldots, a_2a_n, \ldots, a_na_1, a_na_2, \ldots, a_na_n\}$$

Since C is a clique, $\forall x, y \in C^2, x = (x_1, x_2), y = (y_1, y_2)$, $\forall j \in \{1, 2\}$ we know that $x_j, y_j \in C$. and either $x_j = y_j$ or $x_j \neq y_j \implies \exists (x_j, y_j) \in E$, just like before. Concluding that $\forall x, y \in C^2$ $\exists (x, y) \in E^2$ and thus C^2 is a clique of size n^2.

Finally, extend this reasoning for any G^k. Same as before, consider all vertices that form the Max Clique in G to be $C = \{a_1, a_2, \ldots, a_n\}$ of size n. In G^k for any $k \in \mathbb{N}$, these n nodes will generate n^k vertices of the form:

$$C^k = \{(c_1, c_2, \ldots, c_k) \mid \forall j \in [1, k], j \in \mathbb{N}, c_j \in C\}$$

Once again, we can reason that $\forall x, y \in C^k$ such that $x = (x_1, x_2, \ldots, x_k), y = (y_1, y_2, \ldots, y_k)$ and $\forall j \in [1, k], x_j, y_j \in C$, then either $x_j = y_j$ or $x_j \neq y_j \implies \exists (x_j, y_j) \in E$, since C is a clique, concluding that C^k is a clique of size n^k.

Thus, we have concluded that if we have a Max Clique of size n in G, we shall have a Max Clique in G^k whose size will be **at least** n^k. Now we shall proceed to prove that this Max Clique cannot be bigger than n^k.

If we had a clique of size $> n^k$, that would mean that there are vertices in G^k adjacent to every node in C^k that do not belong to C^k. For simplicity, consider that there exists 1 node that fulfills these conditions and, thus, the Max Clique in G^k is of size $n^k + 1$. Call this node w. For $w = (w_1, w_2, \ldots, w_k)$ to be adjacent to all nodes in C^k it must fulfill that $\forall j \in [1, k]$, either $\exists i \in [1, n]$ such that $w_j = a_i$, which means that $w_j \in C$, or $\forall i \in [1, n]\ \exists (w_j, a_i) \in E$.
Since we have restricted $w \notin C^k$, that implies that $\exists w_j \notin C$, otherwise w would belong in C^k. This implies that there exists a node $w_j \in V, w_j \notin C$ which is adjacent to all vertices in C. However, this is a contradiction because if this node w_j existed, it would be adjacent to all n vertices of the Max Clique of G, which would mean that the set of vertices $C \cup \{w_j\}$ would be the actual Max Clique of G, of size $n+1$. Since we had fixed the Max Clique of G to be of size n, we conclude that the Max Clique of G^k cannot be of size greater than n^k and, thus, it is proved that the Max Clique of G^k is of size **exactly** n^k.

(b) If there is a constant approximation algorithm for Max Clique, that means that there is an algorithm A such that the size of the Max Clique found by A is at least the size of the actual Max Clique of G, OPT, divided by k.

$$\frac{OPT}{A(G)} < k$$

A polynomial time approximation schema is such that, for a given $\epsilon > 0$ we can compute a solution for the problem within a factor $1 + \epsilon$ of being optimal.
In this case, the existence of a constant approximation algorithm for Max Clique implies the existence of a polynomial time approximation schema for the problem, as we shall prove here.

We will begin with a simple example that will make everything much clearer. Suppose that given an instance $G = (V, E)$ of Max Clique, we construct G^2 and apply the constant approximation algorithm A. We know that its result $A(G^2)$ will be such that $OPT(G^2) < k \cdot A(G^2)$. By applying what we saw in the previous section, we know that the Max Clique of G^2 will be exactly the square of the size of the Max Clique of G. Hence, both $OPT(G^2) = OPT(G)^2$ and $A(G^2) = A(G)^2$ satisfy, which leads to $OPT(G)^2 < k \cdot A(G)^2 \implies OPT(G) < \sqrt{k} \cdot A(G)$.

By constructing G^2 we have improved our constant approximation algorithm A by a factor of $\sqrt{k}$. This can be improved as much as we want by constructing G^l for any l, which will lead to better and better approximations, resulting in $OPT(G) < \sqrt[l]{k} \cdot A(G)$. In fact,

$$\lim_{l \to \infty} \sqrt[l]{k} = 1, \forall k > 1$$

which assures that we can obtain an approximation as accurate as we wish.

Given an $\epsilon > 0$, we can choose l such that $\sqrt[l]{k} < 1 + \epsilon$ and construct G^l, run our constant approximation algorithm A on G^l, and obtain an approximation $A(G)$ within a factor $\sqrt[l]{k}$ of being optimal. The construction of G^l can be done in $O(l \cdot |V|^{2l})$, since the construction of V^l takes $O(|V|^l)$ and E^l needs to run l checks for each pair of nodes in V^l, resulting in $O(l \cdot |V|^{2l})$, which is polynomial regarding the size of the instance G at hand. We are supposing that an optimal data structure where checking the existence of an edge in E is $O(1)$, such as an adjacency matrix, is being used.

Problema 11: APEI - Haonan i Adrià

Enunciat

The association for the promotion of the European Identity is planning a workshop formed by a series of debates over different European topics. They have a participants list formed by the persons that have committed to participate in the workshop provided the association issues them a formal invitation.

The organizing committee in view of the planned topics and previous experiences has selected for each debate two disjoint lists of people: the success list and the failure list. The committee is considering those list in an optimistic perspective. The presence of at least one of the persons in the success list or the absence of at least one of the persons in the failure list of a particular debate guarantees that this debate will be successful.

The organizing committee faces the problem of selecting the subset of people to whom the association will send a formal invitation. The association wish to invite a set of people that guarantees that the number of successful debates (according to the previous rule) is maximized.

Design a 2-approximation algorithm for the problem. Justify its correctness and efficiency.

Solució

Per solucionar el problema reduirem el problema a un problema de **MAX-SAT**. La reducció és la següent:

(i) Crearem una variable per cada participant y una clàusula per cada debat que hi ha. Les variables indiquen si el participant està invitat o no al taller.

(ii) Per cada participant de un debat, afegirem la variable del participant positiva si aquest participant pertany a la llista de èxits o afegirem la variable del participant negativa si aquest participant pertany a la llista de fracasos. Com les llistes son disjuntes no hi haurà cap tautologia.

(iii) Farem la conjunció de totes les clàusules.

D'aquesta manera aconseguirem una formula F en CNF que representa el problema. Una vegada feta la reducció utilitzarem el següent algoritme randomitzat.
Sigui Y_i la variable indicadora que representa que la clàusula i està satisfeta. Asignam cada variable de la formula F a certa o falsa amb una probabilitat de $\frac{1}{2}$. La probabilitat de que Y_i no sigui 1 es de:

$$P[Y_i == 0] = (\frac{1}{2})^n$$

on n és el número de variables dins la cláusula i. Llavors, la probabilitat de que Y_i sigui certa és de:

$$P[Y_i == 1] = 1 - P[Y_i == 0] = 1 - (\frac{1}{2})^n$$

Com que, com a mínim, una persona ha de participar en cada debat, almenys hi ha una variable dins la clàusula. Això fa que la probabilitat de que Y_i sigui certa es com a mínim de $\frac{1}{2}$. Aleshores,

$$E[Y_i] = Pr[Y_i == 1] \geq \frac{1}{2}$$

El que volem veure és el nombre de clàusules satisfetes. Sigui W el nombre de clàusules satisfetes:

$$W = \sum_i Y_i$$

Aleshores el nombre esperat de clàusules satisfetes és el següent:

$$E[W] = E[\sum_i Y_i] = \sum_i E[Y_i] \geq \sum_i \frac{1}{2} \geq \frac{1}{2} \cdot M \geq \frac{1}{2} \cdot OPT$$

sent M el nombre de clàusules de la fórmula i la darrera part es certa ja que una solució óptima com a molt té el tots els debats exitosos. Aleshores, l'algoritme randomitzat és una 2-aproximació del problema.

Per demostrar la correctesa hem de veure que la reducció és correcta y l'algoritme aconsegueix l'objectiu de maximitzar el nombre de debats amb èxit. A la reducció, les variables representen la invitació del participant i al taller, es a dir, la variable x_i és certa si el participant i està invitat al taller i és falsa al contrari. Per a que un debat tingui èxit és necesari que o una persona de la llista d'èxit estigui invitada o que una persona de la llista de fracasos no estigui invitada. Al construir les clàusules de la manera explicada abans, per a que la clàusula sigui certa, llavors el debat té èxit, la clàusula necessita o que una de les variables positives sigui certa o que una de les variables negatives sigui falsa. D'aquesta manera ens asegurem que o una persona de la llista d'èxit estigui invitada o que almenys una persona de la llista de fracasos no estigui invitada. Llavors, el objectiu és conseguir el màxim nombre de aquestes clàusules satisfetes, el problema de **MAX-SAT** que amb l'algoritme s'aconsegueix una 2-aproximació.

En quant l'eficiència, el cost d'aquest algoritme serà el cost de fer la reducció més el cost de l'algoritme aleatori. Fer la reducció hem de mirar quins participants hi ha en cada debat per fer l'asignació de variables a cada clàusula. Això tindrà un cost de $O(D \cdot P)$, on D és el nombre de debats i P és el nombre de participants. Després, el cost de l'algoritme aleatori serà lineal respecte al nombre de variables ja que hem d'assignar un valor a cada variable, llavors tindrà un cost de $O(P)$ perquè hi ha una variable per cada participant. Aleshores, l'algoritme tindrà un cost de $O(D \cdot P)$.

Problema 14: Min makespan scheduling - Jordi i Neus

Enunciat

Consider the following scheduling problem: Given n jobs, m machines and, for each $1 \leq i \leq m$ and each $1 \leq j \leq n$, the amount of time t_{ij} required for the i-th machine to process the j-th job, find the schedule for all n jobs on these m machines that minimizes the makespan, i.e., the maximum processing time over all machines. You can assume that once a job starts in a machine it must run in this machine until it finishes.

A solution for the problem can be represented by 0-1 variables, x_{ij} , $1 \leq i \leq m$ and $1 \leq j \leq n$, indicating whether the i-th machine will process the j-th job.

(i) Using those variables provide an integer programming formulation for the problem and its corresponding LP relaxation.

Let us call *fractional optimal schedule* an optimal solution of the LP and opt the makespan of an optimal solution of the IP.

Observe that if $t_{ij} >$ opt, then job j will not be assigned to machine i in an optimal solution. This suggest that we can set an upper bound T and set $x_{ij} = 0$, whenever $t_{ij} > T$. Of course, we need to seek a suitable bound T so that a good assignment of tasks to machines is still possible.

(ii) Provide an LP formulation to check if, for a given bound T, a fractional optimal schedule in which $x_{ij} = 0$ when $t_{ij} > T$ exists.

(iii) Show how to find in polynomial time the value T^*, the minimum among all the integers T's for which the previous condition holds.

Assume that x^* is a fractional optimal schedule solution in which $x_{ij} = 0$ when $t_{ij} > T^*$.

Let $J = \{j \mid \exists i, \quad 0 < x^*_{ij} < 1\}$ and $M = \{1, \ldots, m\}$. Consider the bipartite graph $H = (M, J, E)$ where $E = \{(j, i) | 0 < x^*_{ij} < 1\}$; that is, there is an edge (j, i) connecting j to i if and only if the j-th job is partially assigned to the i-th machine.

Assuming that a maximum matching in H covers all the vertices in J:

(iv) Provide a rounding algorithm to obtain, from x^*, a feasible solution to Min makespan scheduling. Your rounding algorithm should provide a 2-approximate solution.

(v) Justify whether, the process to compute such a 2-approximate solution requires polynomial time or not.

Solució

i) ***Nota:*** *makespan = temps de processament màxim sobre totes LOS MAQUINAS*

`https://ac.informatik.uni-freiburg.de/lak_teaching/ws11_12/combopt/notes/makespan_scheduling.pdf`
literalment el mateix problema

Apartat i): M és el conjunt de màquines, J el conjunt de feines
xi,j són variables que indiquen si la feina j està assignada a la màquina i. L'objectiu es minimitzar el *makespan*.
El primer conjunt de restriccions assegura que cada feina estigui assignada a exactament una màquina, i el segon conjunt de restriccions assegura que totes les màquines tinguin com a molt una càrrega de feina de t.
Per fer la relaxació a LP, només cal permetre que les variables xi,j pertanyin a l'interval [0, 1].

$$
\begin{aligned}
\text{minimize} \quad & t \\
\text{subject to} \quad & \sum_{i \in M} x_{i,j} = 1 \quad j \in J, \\
& \sum_{j \in J} p_{i,j} x_{i,j} \leq t, \quad i \in M, \\
& x_{i,j} \in \{0,1\}.
\end{aligned}
$$

Apartat ii) Definim un nou conjunt $S(T) = \{(i,j) : pi, j \leq t\}$, per fer que totes les parelles feina-màquina tals que $tij > T$ tinguin xij = 0.
Llavors, una formulació d'LP podria ser:

$$
\begin{aligned}
\text{minimize} \quad & 0 \\
\text{subject to} \quad & \sum_{i:(i,j) \in S_t} x_{i,j} = 1 \quad j \in J, \\
& \sum_{j:(i,j) \in S_t} p_{i,j} x_{i,j} \leq t, \quad i \in M, \\
& x_{i,j} \geq 0 \quad (i,j) \in S_t.
\end{aligned}
$$

Apartat iii) Trobar T* amb cerca binària.
N'hi ha prou en fer una cerca binària per trobar el valor més petit de t per tal que la resolució d'aquest problema en LP sigui factible. La cerca es farà dins l'interval $[\frac{\alpha}{m}, \alpha]$.
El valor d'α no és més que una base per començar a fer la cerca. El calculem mitjançant un algorisme *greedy* bàsic, que assigna cada tasca a la màquina amb menys temps total de funcionament. α serà el *makespan* resultant d'aquest algorisme. També serà la cota superior del nostre rang de cerca, ja que amb l'algorisme *greedy* hem vist que existeix una assignació pel *makespan* α.
Per altra banda, sabem que la cota inferior és $\frac{\alpha}{m}$. Recordem que α és un *makespan*, és a dir: el temps que triga la màquina que treballa més estona. Imaginem el cas que el problema té només **una sola tasca**, que el *greedy* assignaria a una de les màquines. L'α resultant seria llavors el temps d'aquella tasca, el qual podríem millorar fraccionant la tasca entre totes les m màquines, obtenint així un *makespan* $= \frac{\alpha}{m}$. Conseqüentment, també haurem trobat la cota inferior del rang ja que el temps de la tasca està repartit equitativament i qualsevol altra assignació resultaria en un *makespan* més gran.

iv) Tenim x*
Input. J, M , $t_{i,j}$ per tot $i \in M$ and $j \in J$ *Output.* $x_{i,j} \in \{0,1\}$ per a tot $i \in M$ and $j \in J$

1. Fem la cerca binària de l'apartat iii) per obtenir el valor x^*.
2. Construir el graf bipartit H i trobar-ne el seu *perfect matching* P.
3. Arrodonir en x totes les tasques que estiguin assignades fraccionàriament.

v) (L'algorisme és una 2-aproximació pq ho diu la font d'on ho hem trobat, de totes maneres sembla que l'enunciat NO demani demostrar que és una 2-aproximació, sinó que demostris que la 2-aproximació es pot fer en temps polinòmic. De totes maneres, la demostració es troba al Teorema 7.11 del document `https://ac.informatik.uni-freiburg.de/lak_teaching/ws11_12/combopt/notes/makespan_scheduling.pdf`)
És veu clarament que la 2-aproximació tarda temps polinòmic perquè:

1. L'algorisme *greedy* previ a la cerca binària es pot fer en temps polinòmic: Inicialment tenim totes les màquines buides, i anirem assignant tasques a la primera màquina de la llista. Després de cada assignació ordenarem la llista de manera creixent segons els temps assignats a cada màquina o, en altres paraules, com que només haurem canviat el temps de la primera màquina de la llista, només haurem reordenar l'element en qüestió. Podem fer servir doncs un *insertion sort* que, en aquest cas, és $O(n)$

2. És ben sabut que la cerca binària es pot fer perfectament en temps logarítmic.

3. El graf bipartit es pot construir facilment en temps polinòmic: vertexs = maquines v tasques, arestes = assignacions. Trobar un matching perfecte té cost polinòmic en la mida del problema ($\mathcal{O}(\sqrt{|V|}E)$ usant l'algorisme de Hopcroft-Karp).

4. L'arrodoniment és fàcil de fer en temps polinòmic.

Problema 23: Strong influential group - Pere

Enunciat

The students of the MEI PTDMA course have to design an algorithm to provide a list of contacts pointing to a strong influential group in a social network. To pass the course it is enough to design an App that identifies a smallest *node hub* in the social network. A *node hub* is a subset of participants whose removal leaves a *fully connected* vertex subset. For this exercise you can assume that the social network is an undirected connected graph and that a vertex subset X is *fully connected* if and only if, for every pair of distinct vertices in X, either they are connected by an edge or they are both connected by an edge to a third vertex in X.

(a) Design a constant factor approximation algorithm for the problem. Justify its correctness and efficiency.

(b) Prove that the decision version of the problem when parameterized by the size of the node hub belongs to FPT.

Solució

Primer de tot, cal veure que un graf està *fully connected* si i només si no té cap camí mínim de mida 3 i és connex. La definició de *fully connected* és que per cada parell de vèrtexs, o son adjacents entre ells, o son adjacents a un tercer vertex diferent. Això vol dir que, o existeix un camí de mida 1 entre ells (son adjacents) o existeix un camí de mida 2 (son adjacents a un tercer vèrtex diferent).

(a) No sé.

(b) Donarem un algorisme i veurem que el seu cost és $O(f(k) * p(n))$, demostrant així que pertany a FPT.
L'algorisme és el seguent:

Algorithm 1 $SIG(G, k)$

```
if k < 0 then return False
else if (u, w, t, v) = find3Path(G) then
    return SIG(G − {u}, k − 1) or SIG(G − {v}, k − 1)
else if (u, v) = find∞Path(G) then
    return SIG(G − {u}, k − 1) or SIG(G − {v}, k − 1)
end if
return True
```

L'algorisme funciona perquè si existeix un camí de mida 3 entre u i v, aleshores almenys un dels dos s'haurà d'eliminar. A part, si existeixen dos vèrtexs que no estan connectats (estan a distancia ∞), un dels dos s'haura d'eliminar també. I, com que comprova tots els casos, sempre retornarà el resultat correcte.

A cada crida de l'algorisme, com a mínim es redueix k en 1 i, com que cada crida en fa màxim 2 de noves, en total se'n faran 2^k com a molt.
El cost d'una crida recursiva és obviament polinòmic respecte el nombre de vèrtex del graf ja que es pot comprovar si hi ha un camí de mida 3 o dos vèrtexs no estan connectats fent un BFS desde cada vèrtex en $O(n * (n + m)) \equiv O(n^3)$.

Per tant, com que el cost total de l'algorisme és $O(2^k * n^3)$, queda demostrat que aquest problema pertany a FPT.

Problema 27: Induced matching - Paula

Enunciat

Given a graph G = (V, E), an induced matching of G is a matching F $\subseteq$ E, such that the edge set of the induced subgraph G[V (F)] is F itself. The size of an induced matching is the number of edges in it. The Induced Matching problem is given a graph G and an integer k, to decide whether G has an induced matching of size at least k.
Consider the following reduction rules:

- Remove isolated vertices.
- If there is a non isolated edge (u, v) (i.e., an edge such that u or v (or both) have degree bigger than 1), delete the edges incident with u and those incident with v

(a) Show that by applying the above rules until none of them can be applied, we get an induced matching of G.

(b) Can the previous preprocessing be used to define a kernelization for induced matching parameterized by d + k where d is the maximum degree of the graph G?

Solució

a)
Un *matching* és un conjunt de vèrtex tal que cadascun d'ells és adjacent únicament a un altre vèrtex, és a dir, tots els vèrtex tenen grau exactament 1.
Un *induced matching* és un matching tal que tots els seus nodes conserven totes les arestes del graf original entre nodes del induced matching. És a dir, entre dos vèrtex del induced matching no pot ser que existeixi una aresta que els connecti en el graf original i que no formi part del induced matching.
Demostrem que aplicant les regles de reducció anteriors fins que no es puguin aplicar més, acabem tenint un *induced matching* de G. L'objectiu d'aquestes es mantenir tots els vèrtex de grau exactament 1 i eliminar la resta.
En primer lloc, al aplicar la primera regla eliminem tots els vèrtex aïllats que són de grau zero. En segon lloc, al aplicar la segona regla trobem totes les arestes tal que com a mínim algun dels seus dos vèrtex és de grau major a 1 i eliminem totes les arestes incidents a aquests dos. D'aquesta forma ens assegurem que el graf resultant és un induced matching ja que totes les arestes són de grau exactament 1. A més, no pot existir cap aresta al graf original que uneixi dos vèrtex del induced matching i no haguem eliminat amb les regles, ja que aquesta aresta seria incident a dos vèrtex de grau major a 1 i per tant aquests haurien sigut eliminats.
D'aquesta forma ens assegurem que el graf final sigui un *induced matching*.

b)
No. Ho demostrem a partir d'un contraexemple:
Suposem que k = 1 i que tenim un graf G que és un camí de tres nodes que té la següent forma:
N1 – N2 – N3
Veiem que d = 2, ja que el grau màxim del vèrtex és el del node N2.
La solució d'aquest problema és "sí" ja que existeix un induced matching a G de mida com a mínim 1. El trobem si eliminem algun dels dos vèrtex de les cantonades, per exemple el N3, i veiem que l'induced matching resultant és: N1 – N2
Veiem que els dos vèrtex són de grau exactament 1 i l'aresta que uneix N2 i N3 al graf original no es té en compte ja que N3 no forma part del induced matching.
En cas que apliquessim les regles de reducció mencionades anteriorment, passaria el següent:
- trobariem una aresta no aïllada tal que algun dels seus dos nodes té grau major a 1, per exemple (N1, N2).
- eliminariem totes les arestes incidents a aquests dos nodes, per tant les arestes (N1,N2) i (N2,N3).
El graf resultant serien els tres nodes N1, N2 i N3 aïllats.

- eliminariem els tres nodes ja que estan aïllats
El graf resultant d'aplicar les regles és un graf buit sense nodes i arestes. És a dir, l'induced matching és de mida 0. Per tant, la resposta al problema seria "no" ja que no en el graf resultant no existeix un inducuded matching de mida com a mínim k (on k = 1).
Aquest contraexemple demostra que aplicant les regles de reducció anteriors trobem una instància que dona una solució diferent a la instància original del problema.

Problema 33: Streaming - Valls i Ordoñez

Enunciat

Hemos visto como usar reservoir sampling para muestreo de un stream del que no conocemos su longitud. Queremos extender este resultado a la estimación de valores de una función definida sobre el stream.

Tenemos un stream $s = a_1, a_2, \ldots, a_m$ formado por valores enteros en $[n] = \{0, \ldots, n-1\}$. Para i $\in$ $[n]$, f_i denota la frecuencia de i en s. Queremos estimar el valor

$$g(s) = \sum_{i \in [n]} g(f_i)$$

donde g es una función con valores reales, con $g(0) = 0$.

El siguiente algoritmo combina la obtención de una muestra con un conteo parcial:

- Obtener x, una muestra (con distribución uniforme) sobre [m]
- $r = |\{i \geq x \mid a_i = a_x\}|$
- Return $m(g(r) - g(r-1))$

(a) Demostrad que el algoritmo propuesto proporciona un estimador de la función g.

(b) Proporcionad una implementación del algoritmo propuesto para un stream de datos del que no conocéis su longitud m.

Solució

Algorithm 2 FREQÜÈNCIA()

```
(m, r, x) ← (0, 0, 0)
for element i ∈ s do
    m ← m + 1
    amb probabilitat 1/m do:
            x ← i
            r ← 0
    if i = x then
        r ← r + 1
    end if
end for
return m · (g(r) − g(r − 1))
```

El que fa l'algorisme és comptar el nombre d'elements que rebem per tal de calcular la mida m. Per a cada element a_i, ens quedem amb aquell element per representar el conjunt de la mostra amb probabilitat $1/m'$, on m' és la mida del stream fins a aquell moment. Quan tenim un índex x seleccionat, calculem el nombre d'aparicions de a_x des d'aquell instant fins que ja no quedin dades al stream. En cas que a meitat de l'execució decidim canviar x de la mostra per un de nou, reiniciem el comptador de repeticions a 0.

Siguin A i R els valors aleatoris de les variables a_x i r un cop l'algorisme ha processat tot el stream s i sigui X l'output del programa, Considerem el cas $A = j$, per algun $j \in [n]$, sota aquesta suposició, el valor R haurà de ser algun dels valors del conjunt $\{1, \ldots, f_j\}$, ja que no podem saber en quina de les aparicions de j vam escollir aquell valor com a mostra (i, per tant, vam resetejar el comptador r).

$E[X|A=j]$ representa l'esperança de la sortida del programa si hem agafat com a mostra l'element j. Com l'element j pot haver estat escollit amb la mateixa probabilitat en qualsevol de les aparicions del stream, no sabem quin és el valor del comptador r, però sabem que aquest comptador valdrà en el millor dels casos f_j si hem agafat j com a mostra en la primera aparició del stream. En cas pitjor, valdrà 1 si l'hem seleccionat en l'última iteració. Com hem pogut escollir j en qualsevol de les aparicions amb la mateixa probabilitat, cada possible output amb un valor concret de r té la mateixa probabilitat de ser la sortida del programa. Aquesta probabilitat serà doncs $1/f_j$. En el l'últim pas s'aplica una suma telescòpica maquíssima.
$E[X]$ és l'esperança de la sortida del programa havent seleccionant qualsevol del valors de la mostra (entre 0 i $n-1$ en qualsevol de les seves aparicions. Això és equivalent a agafar l'esperança per a cada possible valor que agafem com a mostra multiplicat per la probabilitat que haguem agafat aquell valor com a mostra. De nou, com agafem qualsevol element del stream amb la mateixa probabilitat, la probabilitat que la mostra sigui el valor i serà f_i/m. És a dir, el nombre de cops que apareix l'element i respecte la mida total del stream.

$$E[X|A=j] = E[m\cdot(g(R)-g(R-1))|A=j] = \sum_{i=1}^{f_j}\frac{1}{f_j}\cdot m\cdot(g(i)-g(i-1)) = \frac{m}{f_j}\cdot g(f_i) - g(0)$$

$$E[X] = \sum_{i=0}^{n-1} Pr[A=i]\cdot E[X|A=i] = \sum_{i=0}^{n-1}\frac{f_i}{m}\cdot\frac{m}{f_i}\cdot g(f_i) - g(0) = \sum_{i=0}^{n-1} g(f_i) = g(s)$$

Tal i com hem vist a classe, si ens quedem l'element que ens arriba per el stream com a representant de la mostra amb probabilitat $1/m'$, on m' és la mida del stream fins a aquell moment, qualsevol element del stream tindrà la mateixa probabilitat de ser agafat com a mostra. Es pot demostrar per inducció tal i com es veu a les transparències. Un cop sabem que l'element que ens quedem com a mostra té la mateixa probabilitat que la resta de ser escollit, apliquem el raonament de l'apartat a per deduir que l'esperança de la sortida del nostre programa és l'esperada.

Referències/Inspiracions

`https://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf` (pàgina 31)
`https://courses.engr.illinois.edu/cs498abd/fa2020/slides/07-lec.pdf`