Approximation Algorithms Exam

Nicolas Schabanel Guillaume Theyssier

January 14, 2005 — 9:00-12:00

Notes. Only the French version is reliable. Exercises are independent of each other and can be processed in any order; questions are not ordered by increasing difficulty; and most of the time one does not need to solve one particular question to solve the following. Let $\#A$ denote the number of elements in set A, $\lfloor x \rfloor = \max\{i \in \mathbb{Z} : i \leqslant x\}$ and $\lceil x \rceil = \min\{i \in \mathbb{Z} : i \geqslant x\}$ denote respectively the floor and ceil integer parts of $x \in \mathbb{R}$.

EXERCISE 1

Consider multicut problem on stars : given a star $G = (V \cup \{r\}, E = \{ur : u \in V\})$ with root r and with capacities $c : E \to \mathbb{Q}_+$ on edges, and a set of pairs of vertices $\mathcal{S} = \{\{s_1, t_1\}, \ldots, \{s_k, t_k\}\}$ such that $s_i \neq t_i$ for all i, find a minimum capacity subset of edges $C \subseteq E$ which disconnects all pairs $\{s_i, t_i\}$.

Question 1 *Give a (polynomial time) factor preserving reduction between this problem and minimum weight vertex cover on undirected graphs.*

Hint. Match leaves and vertices.

EXERCISE 2

Consider set cover problem : given an universal set $\mathcal{U} = \{u_1, \ldots, u_m\}$ and a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_n\}$ with non-negative costs $c : \mathcal{S} \to \mathbb{Q}^+$ on the subsets, find a minimum cost set cover, i.e., a minimum cost subset $C \subseteq \mathcal{S}$ such that $\bigcup_{S \in C} S = \mathcal{U}$. Let f denote the maximum frequency of an element : $f = \max_j \#\{i : u_j \in S_i\}$.

Question 2 *Explain with two sentences (only) why the following integer program* (IP) *computes an optimal solution for this problem. Give the simplest LP relaxation of* (IP), *which will be denoted* (LP).

$$(\text{IP}) \begin{cases} \text{Minimize} & \displaystyle\sum_{i=1}^{n} c_i x_i \\ \text{subject to} & \displaystyle\sum_{i\,:\,u_j \in S_i} x_i \geqslant 1 \quad (\forall j) \\ & x_i \in \{0,1\} \qquad (\forall i) \end{cases}$$

Consider the following algorithm : compute an optimal solution x^* of (LP), and select every set S_i such that $x_i^* \geqslant 1/f$.

Question 3 *Show that the solution computed by this algorithm is always a valid set cover.*

Question 4 *What is the approximation factor guaranteed by this algorithm ? Prove it. To which classic technics does this algorithm belong ?*

EXERCISE 3
In this exercise, we want to count (approximately) the number of solutions to a DNF boolean formula.

> **Problem 1 (Counting DNF solutions)** Let $f = C_1 \vee \cdots \vee C_m$ be a DNF boolean formula on n variables $x_1, \ldots, x_n$. Each clause C_j is of the form $C_j = l_1 \wedge \cdots \wedge l_{k_j}$, where each l_i is a litteral (i.e., a variable or its negation). We assume that every clause is satisfiable and non redundant (i.e., contains each variable at most once, negated or not).
> Compute $\#f$, the number of truth assignments of variables (x_i) that satisfy f.

We want to evaluate $\#f$ by sampling randomly the 2^n possible truth assignments of variables (x_i). Let us first consider uniform sampling : draw a uniform random truth assignment τ in $\{0,1\}^n$, and set $X = 2^n$ if τ satisfies f and $X = 0$ otherwise.

Question 5 *Show that* $\mathbb{E}[X] = \#f$. *How many random bits does every draw of X use ?*

However, X does not estimate $\#f$ correctly in polynomial time, because even if $\#f > 0$, the probability for X to be non-zero can be exponentially small. Thus, a polynomial number of draws of X is not enough to estimate $\#f$ up to a constant factor. Indeed :

Question 6 *Assume n is even and consider the following formula : $f = x_1 \wedge x_2 \wedge \cdots \wedge x_{n/2}$. What is the value of $\#f$? What is the probability for X to be non-zero ?*

Let $X_1, \ldots, X_k$ be the values of k independent draws of X. What is the probability that one (at least) of the draws is non-zero ? Show that if k is polynomial in n, the probability that $X_1 = \cdots = X_k = 0$ is $1 - o(1)$.

We then decide to use a biased variable which samples only satisfying assignments. Let S_j denote the set of truth assignments of (x_i) that satisfy clause C_j (which has k_j litterals). Note that $\#f = \#(\bigcup_j S_j)$.

Question 7 *What is the size of S_j ?*

Let $c(\tau)$ denote the number of clauses satisfied by truth assignement τ. Note that the sum of $c(\tau)$ over all possible truth assignements τ is $M = \sum_{j=1}^{m} \#S_j$.

Question 8 *Design a (polynomial time, in n and m) randomized algorithm that draws a truth assignment τ with probability $c(\tau)/M$, i.e., with probability proportional to $c(\tau)$. How many random bits does your algorithm use (as a function of n and m) ?*
Hint. Remark that you can first draw a set S_j and then choose τ in S_j. Explain in details your random sampling procedure.

Consider the random variable Y defined as follows : draw a truth assignment τ with probability $c(\tau)/M$, and set $Y = M/c(\tau)$.

Question 9 *Show that* $\mathbb{E}[Y] = \#f$.

Let $\sigma^2(Z) = \mathbb{E}[(Z - \mathbb{E}[Z])^2]$ denote the *variance* of a random variable Z.

Question 10 *Show that* $\sigma^2(Y) \leqslant ((m-1)\,\mathbb{E}[Y])^2$, *where* m *is the number of clauses in* f.
Hint. Show that Y belongs to interval $[M/m, M]$.

Recall Chebychev inequality which claims that for all random variable Z and all $a \in \mathbb{R}_+$,

$$\Pr\{\, |Z - \mathbb{E}[Z]| \geqslant a \,\} \leqslant \frac{\sigma^2(Z)}{a^2}.$$

Question 11 *Let* Z_1 *and* Z_2 *be two independent random variables and* $Z = Z_1 + Z_2$. *Show that* $\mathbb{E}[Z_1 Z_2] = \mathbb{E}[Z_1]\,\mathbb{E}[Z_2]$ *and that* $\sigma^2(Z) = \sigma^2(Z_1) + \sigma^2(Z_2)$.

Let $Y_1, \ldots, Y_k$ be the values of k independent draws of Y, and set $Z = (Y_1 + \cdots + Y_k)/k$.

Question 12 *Show that for all* $\epsilon > 0$, *and all* $k \geqslant 4(m-1)^2/\epsilon^2$,

$$\Pr\{|Z - \#f| \leqslant \epsilon \#f\} \geqslant 3/4.$$

Question 13 *Give, for all* $\epsilon > 0$, *a polynomial time (in* n, m *and* $1/\epsilon$*) randomized algorithm which outputs a value* v *such that* $(1-\epsilon)\,\#f \leqslant v \leqslant (1+\epsilon)\,\#f$ *with constant probability (independent of* n, m *and* ϵ*).*

Question 14 *Can we use this algorithm to solve SAT ? Explain why. Can we obtain from it a PTAS for Max-SAT ? Explain why.*

EXERCISE 4
Consider the following problem.

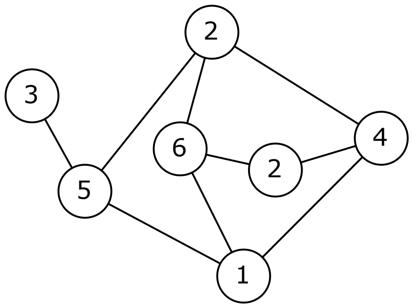
> **Problem 2 (Multicoloring)** Let $G = (V, E)$ be a finite undirected graph with demands $d : V \to \mathbb{N}$ on the vertices.
> A *multicoloring* of G is a function $C : V \to \wp(\mathbb{N})$ that assigns to each vertex $u \in V$ a subset $C(u)$ of $d(u)$ distinct colors (for all $u \in V$, $C(u) \subset \mathbb{N}$ and $\#C(u) = d(u)$), such that two neighboring vertices do not share any color (for all $uv \in E$, $C(u) \cap C(v) = \varnothing$). The *size* of a multicoloring is the number of colors used : $\mathrm{size}(C) = \#(\bigcup_{u \in V} C(u))$. The problem consists in finding a minimum size multicoloring.

Given a clique K in G, denote by $d(K)$ the sum of the demands of the vertices in K. We define the *clique number* as $\omega(G) = \max_{K \text{ clique of } G} d(K)$.

Question 15 *Show that* $\omega(G)$ *is a lower bound on* OPT.

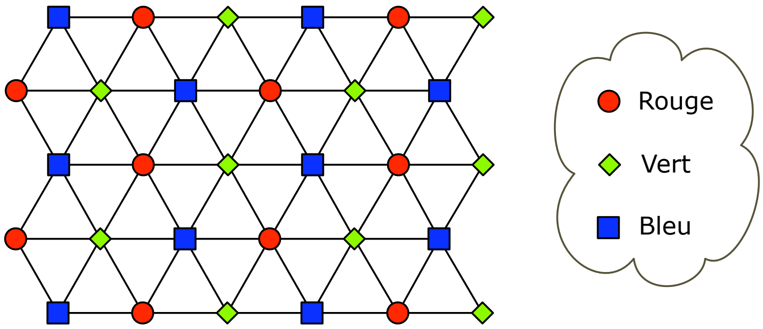
Question 16 *Assume in this question that G is bipartite.*[1] *What is the value of $\omega(G)$? Design a (polynomial time) optimal multicoloring algorithm for graph G.*
Hint. Show that $\mathrm{OPT} = \omega(G)$.

Give the colors assignment computed by your algorithm on the following bipartite graph (the demands are written in the vertices).



We now focus on multicoloring subgraphs induced by finite subsets of vertices in the triangular lattice. Surprisingly enough, this restriction of the multicoloring problem is already NP-complete.

The *triangular lattice* $\mathcal{T}$ is (mathematically speaking) the sublattice of the plane generated by the three vectors $(0,1)$, $(-1/2, \sqrt{3}/2)$ and $(1/2, \sqrt{3}/2)$. It admits the following "Red-Green-Blue" 3-coloring which will be very useful next :



Assume from now on that G is a finite induced subgraph of the triangular lattice. A vertex of G is red, if its color is red in the 3-coloring of $\mathcal{T}$ (same for blue and green).

Let $p = \lceil \omega(G)/3 \rceil$ and $q = \omega(G) - 2p$. We define four sets of colors : the red colors $\{1, \ldots, p\}$, the green colors $\{p+1, \ldots, 2p\}$, the blue colors $\{2p+1, \ldots, 3p\}$ and the black colors $\{3p+1, \ldots, 3p+q\}$.

The first step of the algorithm assigns to each red (resp. blue, green) vertex u, the $\min(p, d(u))$ smallest red (resp. blue, green) colors. Denote by H the subgraph of G induced by the still unsatisfied vertices.

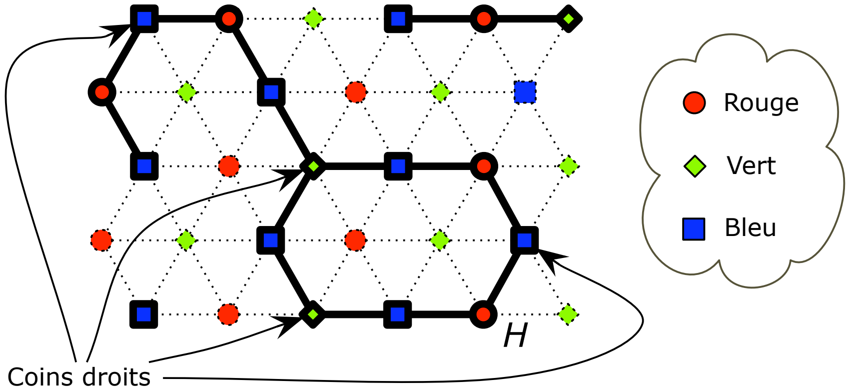
Question 17 *Show that H is triangle-free.*

A vertex in H is a *corner* if it has (at least) two neighbors in H of the same color.

Question 18 *Show that every corner in H has at most three neighbors, and that its neighbors are all of the same color.*

[1] A graph G is *bipartite* if it is 2-colorable in the classic sense, i.e., if there exists a partition (X, Y) of the vertices in V such that every edge in G has one end in X and the other one in Y.

We say that a red (resp. green, blue) corner is *right* if its neighbors in H are green (resp., blue, red) — i.e., if the horizontal edge in the subgraph induced by its neighbors points in the right direction. The figure bellow gives the right corners of the graph H in bold.



Question 19 *Show that the right corners in H form an independent set (i.e., that the subgraph induced by these vertices in H is totally disconnected).*

The second step of the algorithm assigns to each red (resp., green, blue) right corner u de H, the $d(u) - p$ largest blue (resp., red, green) colors. We say that the red right corner *borrows* blue colors from its blue neighbors.

Question 20 *Show that this second step satisfies the demands of all the right corners without conflicts with their neighbors (in G) — i.e., that the borrowed colors were indeed available.*
Hint. Consider the triangle consisting of the red corner, its blue neighbor in G and one of its green neighbors.

Denote by K the subgraph in H induced by the still unsatisfied vertices.

Question 21 *Show that K consists of isolated vertices and trees.*

Question 22 *Show how to satisfy the remaining demand of each isolated vertex in K with black colors, and possibly by borrowing colors from its neighbors (this is the third step of the algorithm).*

Denote by L the subgraph of K induced by the still unsatisfied vertices.

Question 23 *Show how to satisfy the remaining demands of the vertices in L with black colors.*

Question 24 *What is the approximation ratio achieved by this algorithm ?*

Question 25 *Give an infinite family of instances I_n such that* $\mathrm{OPT}(I_n) \geqslant \frac{9}{8}\omega(I_n)$.
Hint. Look for a small triangle-free odd cycle in $\mathcal{T}$, and assigns an uniform demand n to each vertex.

★
★ ★