HW5 Can't get enough Algorithms

MPRI 2.11.1

6.11.2014 - **Due on Thursday 13.11 before 15:00**

You are asked to complete the exercise marked with a [★] and to send me your solutions at:
nicolas.schabanel@liafa.fr
(or drop it in my mail box on the 4th floor) on **Thursday 13.11 before 15:00**.

■ **Exercise 1 (Lasserre Support Lemma).** Consider a convex polytope K on n variables and $y \in \mathsf{LAS}_t(K)$. We want to show that if you induce on some k variables in y, we get an $\tilde{y} \in \mathsf{LAS}_{t-k}(K)$ whose support is included in y's support, where the support of a variable $z \in \mathsf{LAS}_\ell(K)$ denotes the set of indices $i \in [n]$ such that $z_{\{i\}} > 0$.

► ***Question* 1.1)** *Show that* $\mathrm{support}(\tilde{y}) \subseteq \mathrm{support}(y)$.
▷ Hint. *Recall the formula for inducing on variables:* $y_I^{J_0,J_1} = \sum_{H \subseteq J_0} (-1)^{|H|} y_{I \cup J_1 \cup H}$.

■ **Exercise 2 (Lasserre hierarchy: application to scheduling with precedence constraints).**
Let us consider the scheduling problem $P2 \,|\, \mathrm{prec}, p_j = 1 \,|\, C_{\max}$ where you are given n jobs $J_1, \ldots, J_n$ with unit processing time (all jobs last one time unit) and (acyclic) precedence constraints $\prec$ (i.e., $J_i \prec J_j$ means that job J_j cannot start before job J_i is completed), and whose goal is to find a *valid* schedule (no pair of jobs overlaps and jobs are scheduled in an order compatible with $\prec$) on two machines (no more than two jobs can be scheduled simultaneously) without *preemption* (once started, job execution cannot be interrupted) that minimizes the *makespan*, i.e. the completion time of the last completed job.

For the general case of m machines, the problem is NP-hard even with unit processing time. In the case of two machines and unit processing time (the case considered here), there is a matching-based polynomial time algorithm for solving this problem by Coffman and Graham (1972). However, here we want to demonstrate a neat application of the Lasserre hierarchy due to Svensson (2011).

► ***Question* 2.1)** *Show that we can restrict ourselves to valid schedules where all jobs start at integer times.*
▷ Hint. *Show that one can convert any valid schedule that does not schedule all jobs at integer time into a valid schedule that schedules all jobs at integer times without increasing its cost.*

So we consider T time slots of unit length and want to decide whether a makespan of T is possible. We formulate this problem as the following polytope:

$$K(T) = \begin{cases} \sum_{t=1}^{T} x_{j,t} = 1 & (\forall j \in [n]) \\ \sum_{j \in [n]} x_{j,t} \leqslant 2 & (\forall j \in [n]) \\ \sum_{u=1}^{t} x_{i,u} \geqslant \sum_{u=1}^{t+1} x_{j,u} & (\forall J_i \prec J_j)(\forall t \in [T]) \\ x_{j,t} \geqslant 0 & (\forall j \in [n])(\forall t \in [T]) \end{cases}$$

▶ ***Question* 2.2)** *Prove that $K(T)$ is a fractional relaxation whose integral solutions are the feasible solutions of the initial problem.*

▶ ***Question* 2.3)** *Consider the instance $J_1, \dots, J_6$ where $J_i \prec J_j$ for all $i \leqslant 3$ and $j \geqslant 4$. Show that the relaxation $(K(T))_{T\in\mathbb{N}}$ has a gap of $4/3$ for the makespan on the instance.*

We will show that one can construct a feasible integral solution for $K(T)$ from any $y \in \mathsf{LAS}_1(K(T))$ which thus solves exactly the initial problem in polynomial time.

▶ ***Question* 2.4)** *Show that if $\mathsf{LAS}_1(K(T)) = \varnothing$, then $\mathsf{OPT} \geqslant T + 1$.*

Assume now that $\mathsf{LAS}_1(K(T)) \neq \varnothing$ and let $y \in \mathsf{LAS}_1(K(T))$. For all $j \in [n]$, let $C_j = \max\{t \in [T] : y_{\{(j,t)\}} > 0\}$ be the fractional completion time of job J_j according to y. Renumber the jobs so that $C_1 \leqslant \cdots \leqslant C_n$. We then *list-schedule* the jobs in this order, i.e.: we go through the time slots, starting at the beginning and at each time slot, we consider the set of jobs that are unprocessed and have all their predecessors already completed, and then pick the job with the smallest index among those (if any). Let σ_j denote the time slot at which J_j is scheduled according to this algorithm. (Note that there might be only one job scheduled in some time slot and that J_{j+1} may be scheduled before J_j due to precedence constraints.)

▶ ***Question* 2.5)** *Show that if $J_i \prec J_j$ alors $C_i \leqslant C_j - 1$.*
▷ Hint. *Proceed by contradiction and induce y on $x_{iC_i} = 1$ (why can you?) and reason on the resulting $\tilde{y} \in \mathsf{LAS}_0(K(T)) = K(T)$ (use the support lemma).*

We now want to show the main result:

Lemma 1. *For all $j \in [n]$, $\sigma_j \leqslant C_j$.*

We proceed by contradiction and consider the smallest index j such that $\sigma_j \geqslant C_j + 1$. We consider two cases:

▶ ***Question* 2.6)** *Suppose that all time slots before σ_j are all occupied (two jobs are scheduled in every one of them), show a contradiction.*
▷ Hint. *Induce y on variable $x_{j\sigma_j}$.*

▶ ***Question* 2.7)** *Suppose that there is a job k that is scheduled alone in its time slot $\sigma_k < \sigma_j$ before J_j, show a contradiction.*
▷ Hint. *What can you say about the jobs scheduled after J_k and before J_j? Induce y on variable $x_{k,C_k} = 1$ to get a contradiction.*

▶ ***Question* 2.8)** *Conclude with an exact polynomial-time algorithm for the initial problem.*

■ **Exercise 3 (A $(1-\varepsilon)\ln m$-approximation for Set Cover with Lasserre hierarchy).** [★]
We consider the Set Cover problem: Given an universe U of m elements $e_1, \dots, e_m$ and a collection $\mathcal{S}$ of n subsets $S_1, \dots, S_n \subseteq U$ with costs $c_1, \dots, c_n$, find a subcollection $C \subseteq \mathcal{S}$ that covers U at minimum cost, i.e. such that $\cup_{S_i\in C} S_i = U$ and $\mathsf{cost}(C) = \sum_{S_i\in C} c_i$ is minimum.

Dinur and Steurer proved in 2013 that for any $\varepsilon > 0$, it is NP-hard to find a $(1-\varepsilon)\ln m$-approximation for this problem and on the other side, a simple greedy-based LP rounding approach by Chvátal (1972) gives a $\ln(m) + 1$-approximation. So it does not seem that there is much room for any improvement. But more precisely, what the recent result of Dinur and Steurer provides is that there is a $m^{O(1/\varepsilon)}$-time reduction from SAT to $(1-\varepsilon)\ln m$-gap instance of Set Cover. This hardness result however does not rule out a *subexponential time algorithm*, which would give a $(1-\varepsilon)\ln m$-approximation in time $2^{m^{O(\varepsilon)}}$, which is what we are about to prove, following the steps of Chlamtac, Friggstad and Georgiou (2012).

Let us first prove that Set Cover is better approximated when the sets are smaller. Consider the following linear program:

$$\begin{cases} \text{Minimize} & \displaystyle\sum_{i=1}^{n} c_i x_i \\ \text{subject to} & \displaystyle\sum_{S_i \ni j} x_i \geqslant 1 \ \ (\forall j \in [m]) \\ & x_i \geqslant 0 \ \ (\forall i \in [n]) \end{cases}$$

▶ ***Question* 3.1)** *Prove that it is a relaxation of the Set Cover problem.*

Let us now consider the following greedy algorithm. Imagine that a subset X of elements are already covered; for all $S_i \in \mathcal{S}$, we define the *effective cost* of S as its cost split evenly over the elements it would newly covered, i.e. $\text{cost}_{\text{eff}}(S_i) = \frac{c_i}{\#(S_i \setminus X)}$ (note that $\text{cost}_{\text{eff}}(S_i) = \infty$ if $S_i \subseteq X$). The greedy algorithm works as follows: start with an empty cover $C = \varnothing$, and as long as C does not cover all the elements, add to C a set S_i with minimum effective cost among all the unselected sets. If an element e_j is covered for the first time when adding the subset S_i, we remember in a variable $\text{price}(e_j) := \text{cost}_{\text{eff}}(S_i)$ the price paid by the algorithm to cover element e_j.

▶ ***Question* 3.2)** *Prove that at the end of the algorithm:* $\text{cost}(C) = \displaystyle\sum_{j=1}^{m} \text{price}(e_j)$.

In order to analyze this algorithm, let us denote by $r_j^i \in \{1, \dots, \#S_i\}$ the rank of every element $e_j \in S_i$ among the elements of S_i in the order in which they are covered by the algorithm (ties are broken arbitrarily).

▶ ***Question* 3.3)** *Prove that for all* $j \in [m]$, $\text{price}(e_j) \leqslant \displaystyle\sum_{S_i \ni j} \frac{c_i x_i}{\#S_i - r_j^i + 1}$, *where* x *is any feasible solution of the LP.*
▷ Hint. *Note that the algorithm selects always the subset with the cheapest effective cost, which is no more than any average of the effective costs of the available subsets.*

▶ ***Question* 3.4)** *Conclude that the cover* C *output by the algorithm costs at most* $H_k \cdot \text{OPT}_{LP}$ *where* $k = \max\{\#S_i : x_i > 0\}$ *is the size of the largest set used in an optimal LP solution* x, $H_k = \displaystyle\sum_{\ell=1}^{k} \frac{1}{\ell} \leqslant \ln k + 1$ *is the harmonic series, and* OPT_{LP} *denotes the optimal value of the linear relaxation.*

Let us now design our $(1 - \varepsilon) \ln m$-approximation. Assume for now that we know the value OPT of an optimal (integral) solution to Set Cover, and let us define the following convex relaxation polytope:

$$K = \begin{cases} \displaystyle\sum_{i=1}^{n} c_i x_i \leqslant \text{OPT} & \\ \displaystyle\sum_{S_i \ni j} x_i \geqslant 1 & (\forall j \in [m]) \\ x_i \geqslant 0 & (\forall i \in [n]) \end{cases}$$

The algorithm proceeds as follows. First, compute a feasible solution $y^{(0)} \in \text{LAS}_{m^\varepsilon}(K)$. Then, for $\ell = 1, \dots, m^\varepsilon$, we pick recursively the largest set S_i with $y^{(\ell-1)}_{\{i\}} > 0$, we renumber the sets so that $i = \ell$, select set S_i, and induce $y^{(\ell-1)}$ on variable $x_i = 1$ to get a new variable $y^{(\ell)} \in \text{LAS}_{m^\varepsilon - \ell}(K)$. We stop at anytime if all the elements are covered.

► ***Question* 3.5)** *Prove that* $c_1 + \cdots + c_{m^\varepsilon} \leqslant$ OPT.
▷ Hint. *To which polytope belongs* $y^{(m^\varepsilon)}$*?*

If there are still uncovered element, let $X = U \setminus (S_1 \cup \cdots \cup S_{m^\varepsilon})$ be the set of still uncovered elements. The algorithm then considers the collection $\mathcal{S}' = \{S_i \cap X : S_i \in \mathcal{S}, y_{\{i\}}^{(m^\varepsilon)} > 0\}$ induced by X and runs the greedy algorithm described earlier to cover the remaining elements.

► ***Question* 3.6)** *Show that the maximum size of a set in the collection* $\mathcal{S}'$ *is at most* $m^{1-\varepsilon}$.

► ***Question* 3.7)** *Prove that this algorithm yields a cover of cost at most* $(1-\varepsilon)\ln m \cdot$ OPT *for Set Cover. Show that its time complexity is* $2^{m^{O(\varepsilon)}} n^{O(1)}$.

Let us now conclude by showing how to guess a close enough value for OPT. Remark that OPT $\in [\min_i c_i, \sum_{i=1}^n c_i]$.

► ***Question* 3.8)** *Propose an algorithm that would try all powers* $(1+\varepsilon)^\ell \in [\min_i c_i, (1+\varepsilon)\sum_{i=1}^n c_i)$ *as a guess for* OPT. *What cover should it output? What is the cost of the output cover? What is its time complexity?*