M1 Informatique
X1II030

Université de Nantes
2019/2020

Feuille de Travaux Dirigés n° 3
Approximation et Inapproximation

Exercice 3.1 (Approximation de MINIMUM VERTEX COVER)
On rappelle la définition du problème MIN-VERTEX-COVER (MIN-VC).

MIN-VERTEX-COVER (MIN-VC) *Instance* : un graphe $G = (V, E)$, où $V = \{v_1, v_2 \dots v_n\}$ *Solution* : un ensemble $V' \subseteq V$ qui couvre toutes les arêtes de G *Mesure* : le nombre $\|V'\|$ de sommets dans V'

Voici la description d'un Programme Linéaire en nombres entiers (qu'on appellera (ILP)) :

$$\begin{array}{rll} \text{minimiser } & x_1 + x_2 + \ldots + x_n & \\ \text{sous les contraintes } & x_i + x_j \geq 1 & \forall (v_i, v_j) \in E(G) \\ & x_i \in \{0, 1\} & \forall v_i \in V(G) \end{array}$$

1. Indiquer comment, partant d'une solution de (ILP), on peut construire une solution de MIN-VC qui a le même optimum. Justifier.
2. Indiquer comment, partant d'une solution de MIN-VC, on peut construire une solution de (ILP) qui a le même optimum. Justifier.

Les Questions 1. et 2. ci-dessus montrent donc que les problèmes (ILP) et MIN-VC *sont équivalents*.
Voici maintenant la description d'un programme linéaire (qu'on appellera (RLP)). On remarquera qu'ici les y_i ne sont pas nécessairement des entiers, mais peuvent prendre *n'importe quelle valeur réelle* entre 0 et 1.

$$\begin{array}{rll} \text{minimiser } & y_1 + y_2 + \ldots + y_n & \\ \text{sous les contraintes } & y_i + y_j \geq 1 & \forall (v_i, v_j) \in E(G) \\ & 0 \leq y_i \leq 1 & \forall v_i \in V(G) \end{array}$$

Les problèmes de type (RLP) peuvent se résoudre en temps polynomial, alors que les problèmes de type (ILP) sont NP-complets. L'idée est donc d'utiliser le problème (RLP) pour obtenir une *approximation* du problème (ILP), et donc une approximation du problème MIN-VC.

3. Montrer que, pour tout graphe G, $opt(RLP) \leq opt(ILP)$.

Appelons $y^* = (y_1^*, y_2^* \dots y_n^*)$ une *solution optimale* obtenue par (RLP). On définit alors une solution $x^A = (x_1^A, x_2^A \dots x_n^A)$ pour (ILP) de la manière suivante : pour tout $1 \leq i \leq n$, $x_i^A = 1$ si $y_i^* \geq \frac{1}{2}$, et $x_i^A = 0$ sinon.

4. Montrer que si y^* est une solution optimale pour (RLP), alors x^A est une solution pour (ILP).
5. Montrer que pour tout $1 \leq i \leq n$, $x_i^A \leq 2y_i^*$.
6. Soit $opt(RLP) = y_1^* + y_2^* + \ldots y_n^*$, et $sol(ILP) = x_1^A + x_2^A + \ldots x_n^A$. Montrer que $sol(ILP) \leq r \cdot opt(RLP)$, où r est une valeur à déterminer.
7. Conclure quant à l'approximabilité de (ILP) et quant à l'appartenance de MIN-VC à APX.
8. Indiquer (en français) les grandes étapes de l'algorithme d'approximation de ratio r pour MIN-VC que nous venons d'étudier.

Exercice 3.2 (MIN-MAKESPAN)

Soit le problème de minimisation suivant, appelé MIN-MAKESPAN :

> MIN-MAKESPAN
> *Instance* : n tâches de durées $d_1, d_2 \dots d_n$; m machines identiques $M_1, M_2 \dots M_m$, chaque machine ne pouvant réaliser qu'une tâche à la fois
> *Solution* : Une affectation des n tâches aux m machines
> *Mesure* : le temps de terminaison T de l'ensemble des tâches (aussi appelé *makespan*)

Dans la Figure 1, on trouve un exemple de réalisation à $m = 3$ machines (M_1, M_2, M_3), avec $n = 11$ tâches dont les durées sont $d_1 = 2$, $d_2 = 7$, $d_3 = 1$, $d_4 = 3$, $d_5 = 2$, $d_6 = 6$, $d_7 = 2$, $d_8 = 3$, $d_9 = 6$, $d_{10} = 2$, $d_{11} = 5$, et pour lequel $T = 17$ (on ne prétend pas ici que cette réalisation soit optimale).

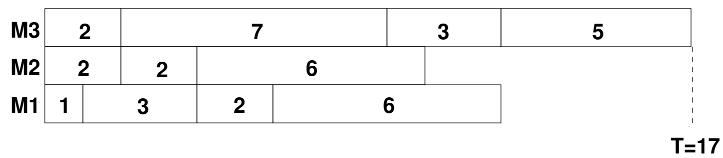

FIGURE 1 – Exemple d'affectation des tâches aux machines, aboutissant à un temps $T = 17$

Il a été démontré que MIN-MAKESPAN est NP-complet. Dans cet exercice, nous voulons montrer que MIN-MAKESPAN est dans APX. Pour cela, on propose l'algorithme suivant (appelé LSA, pour List Scheduling Algorithm) :

(a) prendre les tâches $1, 2 \dots n$ dans l'ordre dans lequel elles sont données ;

(b) pour tout $1 \leq i \leq n$, affecter la tâche i à la première machine disponible (si plusieurs machines sont disponibles, utiliser celle qui a le plus petit indice).

Pour toute instance I du problème MIN-MAKESPAN, on note $T_{opt}(I)$ la solution optimale, et $T_{\text{LSA}}(I)$ le temps obtenu par l'algorithme LSA.

1. Illustrer l'algorithme LSA sur l'exemple de la Figure 1, en présentant le résultat comme dans cette figure, et en indiquant clairement le temps T_{LSA} obtenu.
2. Démontrer que, pour toute instance I, $T_{opt}(I) \geq \frac{\sum_{i=1}^{n} d_i}{m}$.
3. Démontrer que, pour toute instance I, $T_{opt}(I) \geq \max_{i \in [1;n]} d_i$.

Pour toute instance I traitée par LSA, au moins une machine fonctionne en continu jusqu'au temps $T_{\text{LSA}}(I)$. Soit M_i une de ces machines, et soit j la dernière tâche effectuée par M_i. Appelons aussi $t_{deb}(j)$ le temps auquel la tâche numéro j démarre sur M_i.

4. Indiquer, sur l'exemple de la Question 1., les valeurs de i et j, ainsi que l'endroit où se situent (dans le schéma) M_i et $t_{deb}(j)$.
5. Démontrer que $\frac{(\sum_{1 \leq i \leq n} d_i) - d_j}{m} \geq t_{deb}(j)$.
6. Sachant que $T_{\text{LSA}}(I) = t_{deb}(j) + d_j$, en déduire une borne supérieure pour $T_{\text{LSA}}(I)$, et conclure que LSA est un algorithme d'approximation dont vous donnerez le ratio r.
7. Quel est le ratio d'approximation de LSA sur l'exemple de la Figure 1 ?

Supposons avoir l'instance I_0 suivante : m machines ; $n = m^2 + 1$ tâches, avec $d_1 = d_2 = \ldots = d_{m^2} = 1$ et $d_{m^2+1} = m$.

8. Que vaut $T_{\text{LSA}}(I_0)$? Justifier.
9. Que vaut $T_{opt}(I_0)$? Justifier.
10. Vers quoi tend le ratio $r_0 = \frac{T_{\text{LSA}}(I_0)}{T_{opt}(I_0)}$ lorsque m tend vers l'infini ? Que concluez-vous de ce résultat ?

Exercice 3.3 (Inapproximabilité du problème MINIMUM BIN-PACKING)

Soit le problème de minimisation suivant, appelé MINIMUM BIN-PACKING (ou MIN-BP) :

> MINIMUM BIN-PACKING (MIN-BP)
> *Instance* : Un ensemble $S = \{s_1, s_2 \dots s_n\}$ où chaque s_i possède un poids $w_i \in \mathbb{N}$, un entier C
> *Solution* : Une partition $P = \{S_1, S_2, \dots, S_m\}$ de S telle que pour tout $1 \leq j \leq m$, $\sum_{s_k \in S_j} w_k \leq C$
> *Mesure* : m, le nombre d'éléments de la partition P

1. Prenons l'exemple suivant : $n = 9$ (donc $S = \{s_1, s_2 \dots s_9\}$), les poids des s_i sont dans l'ordre $4; 1; 6; 3; 5; 3; 3; 3; 1$, et $C = 10$. Déterminer la valeur optimale du m rechercher, en justifiant de son optimalité.

Voici un autre problème, appelé PARTITION, qui lui est un problème *de décision* :

> PARTITION
> *Instance* : Un ensemble $S = \{s_1, s_2 \dots s_m\}$ où chaque s_i possède un poids $w_i \in \mathbb{N}$
> *Question* : Existe-t-il une partition de S en S_1 et S_2 telle que $\sum_{s_j \in S_1} w_j = \sum_{s_k \in S_2} w_k$?

2. Donner la réponse à PARTITION pour l'instance suivante : $m = 7$, $w_1 = 2$, $w_2 = 11$, $w_3 = 7$, $w_4 = 8$, $w_5 = 5$, $w_6 = 9$ et $w_7 = 4$.
3. Proposer une instance de PARTITION avec $m \geq 7$ et pour laquelle la réponse est NON.

On veut montrer, par contradiction et en utilisant le problème PARTITION, que le problème MIN-BP vu plus haut est APX-dur, et plus précisément qu'il n'est pas approximable sous un ratio $\frac{3}{2}$ (sous des hypothèses raisonnables de complexité, appelées HRC).

4. Indiquer comment transformer toute instance I de PARTITION en une instance I' de MIN-BP, de telle manière que l'optimal pour MIN-BP sur I' soit égal à 2 si et seulement si la réponse à PARTITION sur I est OUI.
5. Supposons que MIN-BP soit approximable avec un ratio $\frac{3}{2}$. Que peut-on alors déduire en ce qui concerne le problème PARTITION ?
6. Sachant que PARTITION est NP-complet, en déduire, sous HRC, que MIN-BP n'est pas approximable sous un ratio $\frac{3}{2}$.

Exercice 3.4 (Étude de MIN-VC3 – Examen 2017-2018)

On ne présente plus le problème d'optimisation MIN-VERTEX-COVER (ou MIN-VC) – voir Exercice 3.1 en cas d'oubli.

On appelle MIN-VC3 (respectivement MIN-VC4) le problème MIN-VC restreint aux graphes de degré maximum 3 (respectivement de degré maximum 4).

1. Montrer que dans un graphe G_Δ de degré maximum Δ et à n sommets, tout vertex cover de G_Δ est de taille au moins égale à $\frac{n}{\Delta+1}$.

Partant de n'importe quel graphe G_4 de degré maximum 4, on le transforme en un graphe G_3 de degré maximum 3 en transformant *chaque sommet de degré 4* comme indiqué dans la Figure 2 pour le sommet v. Les autres sommets (de degré 3 ou moins) ne sont pas modifiés.

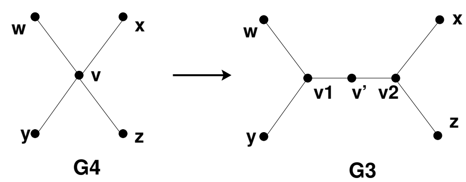

FIGURE 2 – Transformation d'un sommet v de degré 4 dans G_4, en v_1, v' et v_2 dans G_3

On appelle V_4 l'ensemble des sommets de G_4 de degré *exactement* 4, et $n_4 = |V_4|$ le nombre de ces sommets.

On part d'un vertex cover de G_4 qu'on appelle $\mathcal{V}_4$, et on souhaite construire à partir de lui un vertex cover $\mathcal{V}_3$ de G_3. Pour cela, on regarde chaque sommet v de G_4. Si v n'est pas dans V_4, on le met dans $\mathcal{V}_3$ uniquement s'il est dans $\mathcal{V}_4$. Si v est dans V_4, il y a deux cas possibles (voir Questions 2. et 3.) : v est dans $\mathcal{V}_4$, ou il n'y est pas.

2. Supposons que v est dans V_4 et aussi dans $\mathcal{V}_4$. Indiquer quels sont les deux sommets à ajouter à $\mathcal{V}_3$ pour obtenir un vertex cover de G_3. Justifier.
3. Supposons que v est dans V_4 mais pas dans $\mathcal{V}_4$. Indiquer quel est le sommet à ajouter à $\mathcal{V}_3$ pour obtenir un vertex cover de G_3. Justifier.
4. Montrer que l'ensemble $\mathcal{V}_3$ ainsi obtenu est un vertex cover de G_3.
5. Soit vc_3 la taille de $\mathcal{V}_3$ et vc_4 la taille de $\mathcal{V}_4$. Montrer que $vc_3 = vc_4 + n_4$.

On admettra que la réciproque est vraie, c'est-à-dire que s'il existe un vertex cover $\mathcal{V}'_3$ de taille vc'_3 dans G_3, alors on peut construire à partir de lui un vertex cover $\mathcal{V}'_4$ de taille $vc'_4 = vc'_3 - n_4$.

On appelle maintenant $opt(G_3)$ (resp. $opt(G_4)$) la taille du plus petit vertex cover de G_3 (resp. G_4).

6. Montrer que $opt(G_3) = opt(G_4) + n_4$.

Nous allons maintenant supposer que MIN-VC3 est dans PTAS, c'est-à-dire qu'il existe un algorithme $\mathcal{A}_3$ qui, pour tout graphe G_3 de degré maximum 3, détermine un vertex cover de taille $a_3(G_3) \leq (1+\varepsilon) \cdot opt(G_3)$.

Partant d'un graphe G_4 de degré maximum 4, on propose l'algorithme suivant, que l'on appellera $\mathcal{A}_4$:
(a) transformer G_4 en G_3 comme décrit à la Figure 2 ;
(b) appeler l'algorithme $\mathcal{A}_3$;
(c) transformer le vertex cover obtenu par $\mathcal{A}_3$ pour G_3 en un vertex cover pour G_4, comme décrit ci-dessus.

On appelle $a_3(G_3)$ (respectivement $a_4(G_4)$) la taille du vertex cover obtenu par $\mathcal{A}_3$ sur G_3 (resp. $\mathcal{A}_4$ sur G_4).

7. Montrer que $a_4(G_4) = a_3(G_3) - n_4$, et en déduire que $a_4(G_4) \leq (1+\varepsilon) \cdot opt(G_4) + \varepsilon \cdot n_4$.
8. En utilisant la Question 1., montrer que $a_4(G_4) \leq (1+6\varepsilon) \cdot opt(G_4)$.
9. Que signifie le résultat de la question précédente concernant le problème MIN-VC4 ?
10. Sachant que le problème MIN-VC4 est APX-dur, que peut-on en déduire pour MIN-VC3 ? (on attend ici une réponse précise et argumentée)