Université Montpellier II
Master Informatique Modélisation, Optimisation Combinatoire et Algorithmique

Matière : Optimisation Combinatoire
Encadrant : Marin BOUGERET

Devoir à rendre

Guillerme DUVILLIE

Montpellier, le 4 décembre 2012

$\frac{3}{2}-$*dual approximation pour* $P||C_{max}$

L'objectif de cette partie est de présenter une $\frac{3}{2}-$dual approximation pour le problème d'ordonnancement $P||C_{max}$. Nous introduirons dans un premier temps le problème considéré, puis présenterons l'algorithme et dans un troisième temps nous nous consacrerons à l'analyse de ce dernier.

Le problème $P||C_{max}$

Ce problème est un problème classique de l'ordonnancement. Il cherche à répartir un nombre n de tâches sur un nombre m de machines identiques. À chaque tâche est associé un temps d'exécution identique sur toutes les machines, on cherche alors à minimiser le temps total d'exécution.

Définissons $P||C_{max}$ de manière plus formelle.

Entrée : Un ensemble T de n tâches à ordonnancer, un ensemble M de m machines, une fonction de poids p définie comme suit :

$$p : \begin{cases} \text{E} & \rightarrow & \mathbb{N} \\ i & \mapsto & p(i) = p_i \end{cases}$$

Sortie : Un ordonnancement des n tâches sur les m machines minimisant le temps d'exécution.

L'algorithme

Nous étudierons l'algorithme suivant :

Algorithm 1 Une $\frac{3}{2}-$ dual approximation

```
function DivLS(ω, I, m)
    if {j|p_j > ω} ≠ ∅ then
        return FAIL
    Big ← {j|p_j > ω/2}
    Sml ← {j|p_j ≤ ω/2}
    for i ∈ {1, ..., m} do
        x ← max(Big)
        Ordonnancer x sur i
        Big ← Big \ {x}
    if Big ≠ ∅ then
        return FAIL
    return Greedy(Sml)
function Greedy(Sml)
    while Sml ≠ ∅ && ∃ une machine l finissant avant ω do
        Ordonnancer x sur l
        Sml ← Sml \ {x}
    if Sml ≠ ∅ then
        return FAIL
```

Nous allons voir qu'il s'agit bien d'une $\rho-$dual approximation avec $\rho = \frac{3}{2}$.

Analyse

Voici quelques remarques et propriétés concernant l'algorithme **??**.

Lemme 1. *Si l'algorithme n'échoue pas, alors il retourne un ordonnancement.*

Démonstration. Si l'algorithme n'échoue pas, alors les ensembles Big et Sml sont vides. Par construction, on sait que $\text{Big} \cup \text{Sml} = I$, donc, dans ce cas, toutes les tâches on été affectées lors de l'appel à la fonction DivLS. Il s'agit donc bien d'un ordonnancement. □

Lemme 2. *Si l'algorithme échoue, alors $\omega < OPT$.*

Démonstration. Pour démontrer le lemme **??**, nous nous intéresserons à tous les passages de l'algorithme pouvant conduire à un échec de ce dernier. Ils sont au nombre de trois.

1. ligne 3 : l'algorithme échoue s'il existe une tâche dont le temps d'exécution est supérieur à ω. L'ordonnancement affectant toutes les tâches et les machines étant identiques, il va de soit que :

$$OPT \geq \max p_j \quad j \in E$$

   Donc s'il existe au moins une tâche j telle que $p_j > \omega$, alors $\max p_j > \omega \Rightarrow OPT > \omega$

2. ligne 11 : l'échec est provoqué par le fait que le cardinal de Big est supérieur au nombre de machines m. Or, par construction, Big contient toutes les tâches dont le temps d'exécution est supérieur à $\omega/2$. Donc, si $|\,\text{Big}\,| > m$, il faut alors affecter au moins deux tâches de Big à au moins une machine.
   Soient $a = \min(\text{Big})$ et $b = \min(\text{Big} \setminus \{a\})$, on a alors :

$$OPT \geq a + b \quad \Rightarrow \quad OPT > 2\left(\frac{\omega}{2}\right) \quad \Rightarrow \quad OPT > \omega$$

3. ligne 18 : l'algorithme échoue ici lors du traitement des petites tâches [1]. L'algorithme glouton est alors incapable de trouver une machine dont le temps de travail est inférieur à ω, ce qui nous indique :

$$\frac{1}{m} \sum_{i \in E} i > \omega$$

   Or on connaît la borne sur OPT suivante :

$$OPT \geq \frac{1}{m} \sum_{i \in E} i$$

   On a donc $OPT > \omega$.

□

Théorème 1. *L'algorithme* **??** *est une $\rho-$dual approximation et $\rho = \frac{3}{2}$.*

1. C'est à dire l'ensemble des tâches appartenant à Sml.

Démonstration. On s'intéresse à la valeur de la solution retournée par rapport à ω.

L'algorithme procède en deux étapes, le traitement des grosses tâches [2], puis le traitement des petites.

Lors de la première étape, une seule tâche, au plus, a été affectée par machine, le temps d'exécution de cette dernière étant majorée par ω [3].

Lors de la seconde étape, des tâches, dont le temps d'exécution est borné par $\omega/2$ sont rajoutées [4] aux machines dont le temps de travail est inférieur à ω. Considérons une machine l dont le temps de travail est égal à $\omega - 1$ avant ordonnancement d'une tâche j de pois $\omega/2$. Après ordonnancement de j, la machine l aura un temps de travail égal à $(3\omega/2) - 1$, elle ne pourra donc plus être candidate à l'ordonnancement d'une nouvelle tâche, ceci étant vrai quelque soit la machine considérée, le makespan retourné est inférieur à $3\omega/2$.

De plus les lemmes **??** et **??** complètent la preuve du théorème **??**. □

Non existence d'un ***FPTAS*** *pour MKP*

L'objectif de cette partie est de démontrer qu'il n'existe aucun **FPTAS** pour le problème du sac à dos multiple. Pour ce faire nous démontrerons que ceci est vrai pour deux sacs à dos, et nous admettrons que ce résultat se généralise à des instances à m sacs à dos, $m \in \mathbb{N}\backslash\{0, 1\}$.

Le problème du sac à dos multiple

Étant donné un ensemble E de n d'objets, caractérisés par leur volume et leur utilité, et un ensemble S de m sacs à dos, caractérisés par leur contenance maximale, ce problème cherche à déterminer des sous ensembles d'objets associés à chacun des sacs, de manière à ce que le volume total d'aucun des sous ensembles ne soit supérieur à la contenance maximale du sac qui lui est associée et de manière à maximiser l'utilité des objets sélectionnés.

Autrement dit, on cherche à prendre les objets les plus utiles possibles tout en ayant encore la possibilité de fermer tous les sacs.

On peut décrire ce problème de manière plus formelle.

Entrée : un ensemble E de n objets, un ensemble S de m sacs à dos, une fonction de poids [5] définie comme suit :

$$p : \left\{ \begin{array}{ccc} E & \to & \mathbb{N} \\ i & \mapsto & p(i) = p_i \end{array} \right.$$

une fonction de volume définie comme suit :

$$v : \left\{ \begin{array}{ccc} E & \to & \mathbb{N} \\ i & \mapsto & v(i) = v_i \end{array} \right.$$

2. Il s'agit des tâches appartenant à Big.
3. Puisque le contraire validerait la condition de la ligne 2, forçant l'algorithme à signaler un échec.
4. Dans la limite des stocks disponibles...
5. On assimilera l'utilité d'un objet à son poids.

et une fonction de capacité définie comme suit :

$$c : \begin{cases} S & \rightarrow & \mathbb{N} \\ l & \mapsto & c(l) = c_l \end{cases}$$

Sortie : Un ensemble de m sous-ensembles disjoints de E[6], tels que :

$$\sum_{i \in E_l} v_i \leq c_l \quad \forall l \in S$$

et maximisant la fonction objectif suivante :

$$\sum_{l \in S} \sum_{i \in E_l} p_i$$

La réduction gap

Nous utiliserons une réduction gap de MKP vers $2-$partition pour montrer qu'il n'existe aucun **FPTAS** pour MKP.

Théorème 2. *Il n'existe aucun algorithme **FPTAS** pour MKP sauf si $P = NP$.*

Démonstration. Considérons une instance de $2-$partition[7] à $2n$ éléments et de fonction de poids f.

Appelons A le poids des sous-ensemble de la $2-$partition recherchée, on a alors :

$$A = \frac{1}{2} \sum_{i=1}^{2n} f_i$$

On cherche à construire une instance de MKP avec deux sacs à dos à partir de l'instance définie de $2-$partitions. Pour ce faire, on fait correspondre à chaque élément i de l'ensemble de départ de $2-$partition un objet j de volume $v_j = f_i$ et de poids $p_j = 1$. De plus on fixe la capacité de chaque sac à dos ainsi : $c_1 = c_2 = A$.

Supposons à présent l'existence d'un algorithme $(1 - \epsilon)-$approché pour MKP s'exécutant en temps polynomial. Si l'algorithme réussit à affecter un sac à chacun des objets, l'optimal sera alors défini par :

$$OPT = \sum_{i=1}^{2n} p_i = 2n$$

Dans ce cas, l'affectation de chaque objet à un sac définirait une $2-$partition valide. Sinon l'optimal sera borné par au plus $2n - 1$. En sélectionnant ϵ inférieur à $\frac{1}{2n}$, il sera possible de différencier ces deux cas, et donc dans le cas ou $OPT = 2n$, on pourrait répondre OUI au problème de $2-$partition associé, sinon, la réponse à $2-$partition serait NON.

6. On notera E_l le sous ensemble associé au sac à dos l.
7. Il sera considéré que le lecteur connaît ce problème.

$$\begin{array}{rrcl} & (1-\epsilon)OPT & > & 2n-1 \\ \Rightarrow & -\epsilon & > & \dfrac{2n-1-OPT}{OPT} \quad \text{or} \quad OPT \geq 2n \\ \Rightarrow & -\epsilon & > & \dfrac{-1}{2n} \\ \Rightarrow & \epsilon & < & \dfrac{1}{2n} \end{array}$$

L'existence d'un **FPTAS** pour MKP permet la résolution de $2-$partition en temps polynomial, ce dernier étant $NP-$complet, ceci l'hypothèse de départ n'est vraie que si $P = NP$. $\square$