Complexité paramétrée (4)
Techniques algorithmiques

Christophe PAUL
(CNRS - LIRMM)

November 11, 2008

Programmation dynamique pour le Sac À Dos

- *Données :* Un ensemble $S = \{s_1, \ldots s_n\}$ et une fonction de valeur $v : S \rightarrow \mathbb{N}^+$ et une fonction $w : S \rightarrow \mathbb{N}^+$
- *Paramètre :* Un entier $k > 0$
- *Question :* Trouver un sous-ensemble S' de S de valeur maximum tel que
$$\sum_{s \in S'} w(s) \leqslant k \ ?$$

Programmation dynamique pour le Sac À Dos

- *Données :* Un ensemble $S = \{s_1, \dots s_n\}$ et une fonction de valeur $v : S \rightarrow \mathbb{N}^+$ et une fonction $w : S \rightarrow \mathbb{N}^+$
- *Paramètre :* Un entier $k > 0$
- *Question :* Trouver un sous-ensemble S' de S de valeur maximum tel que
$$\sum_{s \in S'} w(s) \leqslant k \ ?$$

Remarque sur la taille de la donnée

Les valeurs et les poids sont représentés par des $2n$ entiers codées en binaire. Nous supposerons donc que la taille de chaque entier est $\lceil \log W \rceil$.

Programmation dynamique pour le SAC À DOS

- *Données :* Un ensemble $S = \{s_1, \dots s_n\}$ et une fonction de valeur $v : S \rightarrow \mathbb{N}^+$ et une fonction $w : S \rightarrow \mathbb{N}^+$
- *Paramètre :* Un entier $k > 0$
- *Question :* Trouver un sous-ensemble S' de S de valeur maximum tel que
$$\sum_{s \in S'} w(s) \leqslant k \ ?$$

Remarque sur la taille de la donnée

Les valeurs et les poids sont représentés par des $2n$ entiers codées en binaire. Nous supposerons donc que la taille de chaque entier est $\lceil \log W \rceil$.

$\Rightarrow$ **Un algorithme de complexité $O(W.n)$ est donc exponentiel car $W = 2^{\log W}$**

Sac à Dos à valeurs binaires

On suppose que $w(s_i) \leqslant k$ pour tout $s_i \in S$.

- Soit la variable booléene $R[t, S'] =$ VRAI ssi il existe un sous-ensemble de S' de poids $t \in \mathbb{N}$

Sac à Dos à valeurs binaires

- Soit la variable booléene $R[t, S'] = \text{VRAI}$ ssi il existe un sous-ensemble de S' de poids $t \in \mathbb{N}$

On calcule le tableau suivant. On note $S_i = \{s_1, \dots s_i\}$

$R[t, S']$	$\emptyset$	S_1	S_2	$\dots$	S_i	$\dots$	S
1	F						
2	F	F	V	$\dots$	V	$\dots$	V
$\dots$	F						
j	F	V	V	$\dots$	V	$\dots$	V
$\dots$	F						
k	F						

SAC À DOS à valeurs binaires

- Soit la variable booléene $R[t, S'] = \text{VRAI}$ ssi il existe un sous-ensemble de S' de poids $t \in \mathbb{N}$

On calcule le tableau suivant. On note $S_i = \{s_1, \dots s_i\}$

$R[t, S']$	$\emptyset$	S_1	S_2	$\dots$	S_i	$\dots$	S
1	F						
2	F	F	V	$\dots$	V	$\dots$	V
$\dots$	F						
j	F	V	V	$\dots$	V	$\dots$	V
$\dots$	F						
k	F						

$$R[t, S_{i+1}] = R[t, S_i] \vee R[t - w(s_{i+1}), S_i]$$

Sac à Dos à valeurs binaires

- Soit la variable booléene $R[t, S'] =$VRAI ssi il existe un sous-ensemble de S' de poids $t \in \mathbb{N}$

On calcule le tableau suivant. On note $S_i = \{s_1, \ldots s_i\}$

$R[t, S']$	$\emptyset$	S_1	S_2	$\ldots$	S_i	$\ldots$	S
1	F						
2	F	F	V	$\ldots$	V	$\ldots$	V
$\ldots$	F						
j	F	V	V	$\ldots$	V	$\ldots$	V
$\ldots$	F						
k	F						

$$R[t, S_{i+1}] = R[t, S_i] \vee R[t - w(s_{i+1}), S_i]$$

Complexité: $O(W.n)$

Sac à Dos à valeurs entières

- Soit la variable **entière** $R[t,S'] \leqslant k$ stocke la valeur maximale d'un sous-ensemble de S' dont le poids est t.

On note S_i l'ensemble $\{s_1, \ldots s_i\}$

$R[t,S']$	$\emptyset$	$\ldots$	S_i	S_{i+1}	$\ldots$	S
1	0					
2	0	$\ldots$	$R[t-w(s_{i+1}),S_i]$		$\ldots$	
$\ldots$	0					
t	0	$\ldots$	$R[t,S_i]$	$R[t,S_i+1]$	$\ldots$	
$\ldots$	0					
k	0					

$$R[t,S_{i+1}] = \max\{R[t,S_i], R[t-w(s_{i+1}),S_i] + v(s_{i+1})\}$$

Remarque

- En fait, le problème du SAC À DOS est un problème **pseudo-polynomial**, i.e. il peut être résolu en temps polynomial si la donnée est codée en unaire.
- Ce n'est pas le cas par exemple de VERTEX COVER, qui est alors dit **fortement NP-complet**.

Lemme

Si un problème d'optimisation Π admet un FPTAS, alors il admet un algorithme pseudo-polynomial.

Compression itérative - Principe

Utilisée pour les problèmes de **minimisation** dont le paramètre est la taille k de la solution.

1. **Etape de compression :**
   Etant donnée une solution de taille $k+1$, trouver un algo **FPT** qui
   - soit on construit une solution de taille k
   - ou prouve qu'il n'y a pas de solution de taille k

Compression itérative - Principe

Utilisée pour les problèmes de **minimisation** dont le paramètre est la taille k de la solution.

1. **Etape de compression :**
   Etant donnée une solution de taille $k+1$, trouver un algo **FPT** qui
   - soit on construit une solution de taille k
   - ou prouve qu'il n'y a pas de solution de taille k
2. **Itération :**
   L'algorithme considère les sous-instances $X_i = X[x_1, \ldots x_i]$ les unes après les autres. Et à partir d'une solution S_i pour X_i:

Compression itérative - Principe

Utilisée pour les problèmes de **minimisation** dont le paramètre est la taille k de la solution.

1. **Etape de compression :**
   Etant donnée une solution de taille $k+1$, trouver un algo **FPT** qui
   - soit on construit une solution de taille k
   - ou prouve qu'il n'y a pas de solution de taille k
2. **Itération :**
   L'algorithme considère les sous-instances $X_i = X[x_1, \ldots x_i]$ les unes après les autres. Et à partir d'une solution S_i pour X_i:
   - construit une solution triviale pour G_{i+1} de taille $|S_{i+1}|$
   - applique l'étape de compression pour essayer d'améliorer cette solution

Compression itérative pour VERTEX COVER

Lemme : Soient $V = \{v_1, \dots v_n\}$ les sommets d'un graphe G et S_i un VERTEX COVER de taille k pour $G_i = G[v_1, \dots v_n]$.
On peut vérifier s'il existe un VERTEX COVER de taille k pour G_{i+1} et si c'est le cas le trouver en temps $O(2^{k+1}.m)$.

Compression itérative pour VERTEX COVER

Lemme : Soient $V = \{v_1, \dots v_n\}$ les sommets d'un graphe G et S_i un VERTEX COVER de taille k pour $G_i = G[v_1, \dots v_n]$.
On peut vérifier s'il existe un VERTEX COVER de taille k pour G_{i+1} et si c'est le cas le trouver en temps $O(2^{k+1}.m)$.

Conséquence

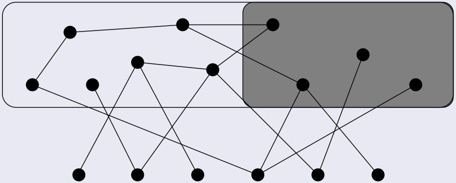
(VERTEX COVER, k) peut être résolu en temps $O(2^{k+1}.m.n)$ par la méthode de la compression itérative.

Etape de compression pour VERTEX COVER

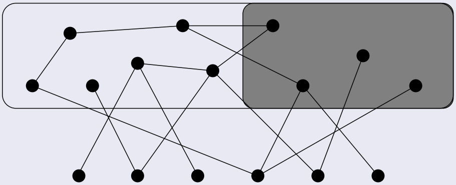
- Soit C un VERTEX COVER de taille k pour G_i, alors $C' = C \cup \{v_{i+1}\}$ est un VERTEX COVER de taille $k+1$ pour G_{i+1}

Etape de compression pour Vertex Cover

- Soit C un Vertex Cover de taille k pour G_i, alors $C' = C \cup \{v_{i+1}\}$ est un Vertex Cover de taille $k+1$ pour G_{i+1}
- Soit (C'_0, C'_1) une partition de C' avec
  C'_0 : l'ensemble des sommets qui seront absents dans C"
  C'_1 : l'ensemble des sommets qui seront gardés dans C"

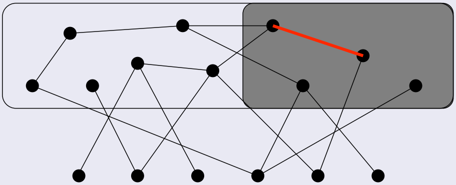


Etape de compression pour VERTEX COVER



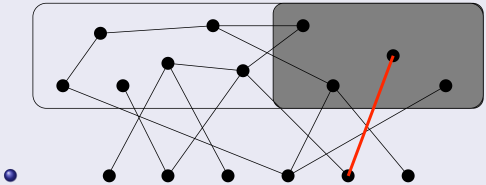
1. $C"_0 = \emptyset$
2. $\forall (u, v) \in E_{i+1}$ tq $u \notin C'_1$ et $v \notin C'_1$:

Etape de compression pour VERTEX COVER



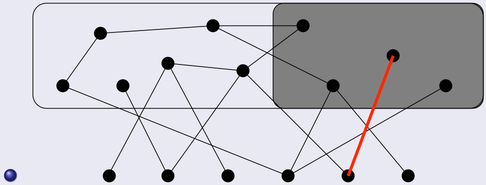
1. $C"_0 = \emptyset$
2. $\forall (u, v) \in E_{i+1}$ tq $u \notin C'_1$ et $v \notin C'_1$:
    1. Si $u \in C'_0$ et $v \in C'_0$, alors compression impossible

Etape de compression pour VERTEX COVER



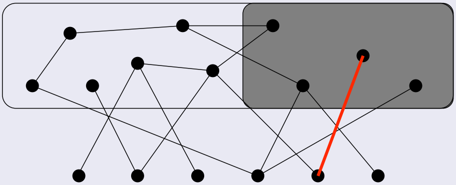
1. $C"_0 = \emptyset$
2. $\forall (u,v) \in E_{i+1}$ tq $u \notin C'_1$ et $v \notin C'_1$:
    1. Si $u \in C'_0$ et $v \in C'_0$, alors compression impossible
    2. Si $u \in C'_0$ et $v \in V_{i+1} \setminus C'$, alors $C''_0 = C''_0 \cup \{v\}$
    3. Si $u \in V_{i+1} \setminus C'$ et $v \in C'_0$, alors $C''_0 = C''_0 \cup \{u\}$

Etape de compression pour VERTEX COVER



1. $C''_0 = \emptyset$
2. $\forall (u,v) \in E_{i+1}$ tq $u \notin C'_1$ et $v \notin C'_1$:
   1. Si $u \in C'_0$ et $v \in C'_0$, alors compression impossible
   2. Si $u \in C'_0$ et $v \in V_{i+1} \setminus C'$, alors $C''_0 = C''_0 \cup \{v\}$
   3. Si $u \in V_{i+1} \setminus C'$ et $v \in C'_0$, alors $C''_0 = C''_0 \cup \{u\}$
3. Si $|C'_1 \cup C''_0| \leqslant k$, alors retourner $C'' = C'_1 \cup C''_0$

Etape de compression pour VERTEX COVER



- 1. $C''_0 = \emptyset$
  2. $\forall (u, v) \in E_{i+1}$ tq $u \notin C'_1$ et $v \notin C'_1$:
     1. Si $u \in C'_0$ et $v \in C'_0$, alors compression impossible
     2. Si $u \in C'_0$ et $v \in V_{i+1} \setminus C'$, alors $C''_0 = C''_0 \cup \{v\}$
     3. Si $u \in V_{i+1} \setminus C'$ et $v \in C'_0$, alors $C''_0 = C''_0 \cup \{u\}$
  3. Si $|C'_1 \cup C''_0| \leqslant k$, alors retourner $C'' = C'_1 \cup C''_0$
- Il suffit donc de tester les 2^{k+1} partitions possibles de C'

1. Arbre de recherche bornée
2. Programmation dynamique
3. Compression itérative
4. Color coding