Introduction à la complexité paramétrée

Christophe PAUL
(CNRS - LIRMM)

June 5, 2008

Bibliographie

- *Parameterized Complexity*, R. Downey and M. Fellows, 1999.
- *Invitation to Fixed-Parameter Algorithms*, R. Niedermeier, 2006.
- *Parameterized Complexity Theory*, J. Flum and M. Grohe, 2006.

"Mesurer la complexité seulement en fonction de la taille de la donnée signifie ignorer tout information structurelle sur l'instance donnée. . ."

J. Flum and M. Grohe

"Mesurer la complexité seulement en fonction de la taille de la donnée signifie ignorer tout information structurelle sur l'instance donnée..."

J. Flum and M. Grohe

"L'idée fondamentale est de restreindre l'explosion combinatoire, semble-t-il inévitable, qui est responsable de la croissance exponentielle du temps de calcul, à un paramètre spécifique au problème..."

R. Niedermeier

L'exemple de SAT

Mesure de la complexité en fonction de différents paramètres:

1. **taille des clauses :** k = nombre max. de littéraux par clause
   $k = 2$: SAT $\in$ **P**
   $k \geqslant 3$: SAT $\in$ **NP**-complet

L'exemple de SAT

Mesure de la complexité en fonction de différents paramètres:

1. **taille des clauses :** k = nombre max. de littéraux par clause
   $k = 2$: SAT $\in$ **P**
   $k \geqslant 3$: SAT $\in$ **NP**-complet
2. **nombre de variables :** n = nombre de variables
   Il y a 2^n affectations possibles
   Si on se restreint à 3-SAT, la complexité tombe à $O(1,49^n)$

L'exemple de SAT

Mesure de la complexité en fonction de différents paramètres:

1. **taille des clauses :** k = nombre max. de littéraux par clause
   $k = 2$: SAT $\in$ **P**
   $k \geqslant 3$: SAT $\in$ **NP**-complet
2. **nombre de variables :** n = nombre de variables
   Il y a 2^n affectations possibles
   Si on se restreint à 3-SAT, la complexité tombe à $O(1,49^n)$
3. **nombre de clauses :** m = nombre de clauses
   On obtient une complexité en $O(1,24^m)$

L'exemple de SAT

Mesure de la complexité en fonction de différents paramètres:

1. **taille des clauses :** $k =$ nombre max. de littéraux par clause
   $k = 2$: SAT $\in$ **P**
   $k \geqslant 3$: SAT $\in$ **NP**-complet
2. **nombre de variables :** $n =$ nombre de variables
   Il y a 2^n affectations possibles
   Si on se restreint à 3-SAT, la complexité tombe à $O(1,49^n)$
3. **nombre de clauses :** $m =$ nombre de clauses
   On obtient une complexité en $O(1,24^m)$
4. **longueur de la formule :** $l =$ nombre total de littéraux
   On obtient une complexité en $O(1,08^l)$

L'exemple de SAT

Mesure de la complexité en fonction de différents paramètres:

1. **taille des clauses :** k = nombre max. de littéraux par clause
   $k = 2$: SAT $\in$ **P**
   $k \geqslant 3$: SAT $\in$ **NP**-complet
2. **nombre de variables :** n = nombre de variables
   Il y a 2^n affectations possibles
   Si on se restreint à 3-SAT, la complexité tombe à $O(1,49^n)$
3. **nombre de clauses :** m = nombre de clauses
   On obtient une complexité en $O(1,24^m)$
4. **longueur de la formule :** l = nombre total de littéraux
   On obtient une complexité en $O(1,08^l)$
5. **structure de la formule :** paramètres basés par exemple sur la structure du graphe de dépendances...

L'exemple de Vertex Cover

- **Données :** Un graphe $G = (V, E)$ et un entier positif k
- **Question :** $\exists$? k sommets tels que toute arête est incidente à au moins un de ces k sommets

L'exemple de Vertex Cover

- **Données :** Un graphe $G = (V, E)$ et un entier positif k
- **Question :** $\exists$? k sommets tels que toute arête est incidente à au moins un de ces k sommets

Paramètre naturel : la taille de la solution k

L'exemple de VERTEX COVER

- **Données :** Un graphe $G = (V, E)$ et un entier positif k
- **Question :** $\exists$? k sommets tels que toute arête est incidente à au moins un de ces k sommets

Paramètre naturel : la taille de la solution k

Observation

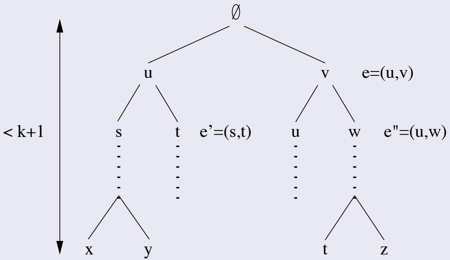
Si $S \subseteq V$ est une couverture et $e = (u, v)$ une arête de G, alors $u \in S$ ou $v \in S$

Un arbre de recherche borné pour Vertex Cover

VC(G, S) : Soit $e = (u, v)$ une arête de G,

1. $S = S \cup \{u\}$
   Si S ne couvre pas E et **si** $|S| < k$, VC$(G - u, S)$
2. $S = S \cup \{v\}$
   Si S ne couvre pas E et **si** $|S| < k$, VC$(G - v, S)$

Complexité : $2^k.n^{O(1)}$



L'exemple de ENSEMBLE INDÉPENDANT

- **Données :** Un graphe $G = (V, E)$ et un entier positif k
- **Question :** $\exists$? k sommets deux à deux non-adjacents

L'exemple de ENSEMBLE INDÉPENDANT

- **Données :** Un graphe $G = (V, E)$ et un entier positif k
- **Question :** $\exists$? k sommets deux à deux non-adjacents

- Algorithme "brut-force" : $O(n^k)$
  MAIS la base n'est pas une constante!

L'exemple de ENSEMBLE INDÉPENDANT

- **Données :** Un graphe $G = (V, E)$ et un entier positif k
- **Question :** $\exists$? k sommets deux à deux non-adjacents

- Algorithme "brut-force" : $O(n^k)$
  MAIS la base n'est pas une constante!

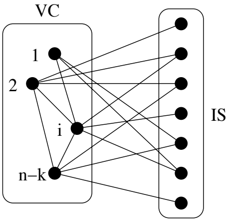
Observation

G possède un ENSEMBLE INDÉPENDANT de taille k ssi G possède un VERTEX COVER de taille $n - k$.

L'exemple de ENSEMBLE INDÉPENDANT

- **Données :** Un graphe $G = (V, E)$ et un entier positif k
- **Question :** $\exists$? k sommets deux à deux non-adjacents

- Algorithme "brut-force" : $O(n^k)$
  MAIS la base n'est pas une constante!
- L'arbre de recherche "à la VERTEX COVER" donne $2^{(n-k)}.n^{O(1)}$
  MAIS l'explosion combinatoire est toujours fonction de n



Observation

G possède un ENSEMBLE INDÉPENDANT de taille k ssi G possède un VERTEX COVER de taille $n - k$.

On recherche des problèmes qui

- sont **NP**;
- admettent un paramètre "*naturel*" k; et
- peuvent se résoudre par un algorithme $\mathcal{A}$ de complexité $f(k).n^{O(1)}$.

On recherche des problèmes qui

- sont **NP**;
- admettent un paramètre "*naturel*" k; et
- peuvent se résoudre par un algorithme $\mathcal{A}$ de complexité $f(k).n^{O(1)}$.

Remarque

La fonction f est quelconque et ne dépend que du paramètre k.
La fonction suivante est donc valide:

$$f(k) = 2^{2^{2^{2^{2^{2^{2^{2^{2^{2^{2^{2^{2^{k}}}}}}}}}}}}}$$

Définition

Soit Σ un alphabet fini.

1. Une **paramétrisation** de Σ^* est une fonction $\kappa : \Sigma^* \rightarrow \mathbb{N}$ calculable en temps polynomial.

Définition

Soit Σ un alphabet fini.

1. Une **paramétrisation** de Σ^* est une fonction $\kappa : \Sigma^* \longrightarrow \mathbb{N}$ calculable en temps polynomial.
2. Un **problème paramétré** (sur Σ) est une paire (Q, κ) tel que $Q \subseteq \Sigma^*$ et κ est une paramétrisation de Σ^*.

   $x \in \Sigma^*$ est une instance de Q et $\kappa(x)$ est le paramètre correspondant.

Définition

Soit Σ un alphabet fini.

1. Une **paramétrisation** de Σ^* est une fonction $\kappa : \Sigma^* \rightarrow \mathbb{N}$ calculable en temps polynomial.
2. Un **problème paramétré** (sur Σ) est une paire (Q, κ) tel que $Q \subseteq \Sigma^*$ et κ est une paramétrisation de Σ^*.

   $x \in \Sigma^*$ est une instance de Q et $\kappa(x)$ est le paramètre correspondant.

Exemple : Vertex Cover

- *Données :* Un graphe $G = (V, E)$
- *Paramètre :* Un entier $\kappa(G)$
- *Question :* G admet-il un Vertex Cover de taille $\kappa(G)$?

Définition

Soit Σ un alphabet fini et $\kappa : \Sigma^* \rightarrow \mathbb{N}$ une paramétrisation.

1. Un algorithme $\mathcal{A}$ sur Σ est un **algorithme paramétré** par κ s'il existe une fonction calculable $f : \mathbb{N} \rightarrow \mathbb{N}$ tel que $\forall x \in \Sigma^*$, la complexité de $\mathcal{A}$ est:
$$f(\kappa(x)).n^{O(1)}$$

Définition

Soit Σ un alphabet fini et $\kappa : \Sigma^* \to \mathbb{N}$ une paramétrisation.

1. Un algorithme $\mathcal{A}$ sur Σ est un **algorithme paramétré** par κ s'il existe une fonction calculable $f : \mathbb{N} \to \mathbb{N}$ tel que $\forall x \in \Sigma^*$, la complexité de $\mathcal{A}$ est:
$$f(\kappa(x)).n^{O(1)}$$
2. Un problème (Q, κ) est **FPT** (**Fixed Parameterized Tractable**) s'il existe un algorithme $\mathcal{A}$ paramétré par κ qui décide Q.

Définition

Soit Σ un alphabet fini et $\kappa : \Sigma^* \rightarrow \mathbb{N}$ une paramétrisation.

1. Un algorithme $\mathcal{A}$ sur Σ est un **algorithme paramétré** par κ s'il existe une fonction calculable $f : \mathbb{N} \rightarrow \mathbb{N}$ tel que $\forall x \in \Sigma^*$, la complexité de $\mathcal{A}$ est:
$$f(\kappa(x)).n^{O(1)}$$
2. Un problème (Q, κ) est **FPT** (**Fixed Parameterized Tractable**) s'il existe un algorithme $\mathcal{A}$ paramétré par κ qui décide Q.

Observation

Tout problème $\Pi \in$ **P** est **FPT**.

Définition

Soit (Q, κ) un problème paramétré et $l \in \mathbb{N}$. Le **niveau** t de (Q, κ) est le problème:

$$(Q, \kappa)_t = \{x \in Q \mid \kappa(x) = t\}$$

Définition

Soit (Q, κ) un problème paramétré et $l \in \mathbb{N}$. Le **niveau** t de (Q, κ) est le problème:

$$(Q, \kappa)_t = \{x \in Q \mid \kappa(x) = t\}$$

Observation

Soit (Q, κ) un problème paramétré et $t \in \mathbb{N}$. Si (Q, κ) est **FPT**, alors $(Q, \kappa)_t \in$ **P** .

Définition

Soit (Q, κ) un problème paramétré et $l \in \mathbb{N}$. Le **niveau** t de (Q, κ) est le problème:

$$(Q, \kappa)_t = \{x \in Q \mid \kappa(x) = t\}$$

Observation

Soit (Q, κ) un problème paramétré et $t \in \mathbb{N}$. Si (Q, κ) est **FPT**, alors $(Q, \kappa)_t \in$ **P** .

k-COLORATION $\in$ **FPT** ?

Définition

Soit (Q, κ) un problème paramétré et $l \in \mathbb{N}$. Le **niveau** t de (Q, κ) est le problème:

$$(Q, \kappa)_t = \{x \in Q \mid \kappa(x) = t\}$$

Observation

Soit (Q, κ) un problème paramétré et $t \in \mathbb{N}$. Si (Q, κ) est **FPT**, alors $(Q, \kappa)_t \in$ **P** .

k-COLORATION $\in$ **FPT** ?

Le problème 3-COLORATION$=$ (COLORATION, $\kappa)_3$ est **NP**-complet

$\Rightarrow$ k-COLORATION n'est pas **FPT**.

Retour sur VERTEX COVER - Règles de réduction

1. Si x est un sommet isolé, alors n'appartient à aucune solution optimale.

$$\textsc{Vertex Cover}(G,k) = \textsc{Vertex Cover}(G-x,k)$$

Retour sur Vertex Cover - Règles de réduction

1. Si x est un sommet isolé, alors n'appartient à aucune solution optimale.

$$\textsc{Vertex Cover}(G,k) = \textsc{Vertex Cover}(G-x,k)$$

2. Si x est un sommet de degré 1 voisin de y, alors il existe une solution optimale contenant y.

$$\textsc{Vertex Cover}(G,k) = \textsc{Vertex Cover}(G-\{y,x\},k-1)$$

Retour sur Vertex Cover - Règles de réduction

1. Si x est un sommet isolé, alors n'appartient à aucune solution optimale.

$$\textsc{Vertex Cover}(G,k) = \textsc{Vertex Cover}(G-x,k)$$

2. Si x est un sommet de degré 1 voisin de y, alors il existe une solution optimale contenant y.

$$\textsc{Vertex Cover}(G,k) = \textsc{Vertex Cover}(G-\{y,x\},k-1)$$

3. **Si x est de degré $\geqslant k+1$, alors si G possède une solution de taille k, elle contient le sommet x.**

$$\textsc{Vertex Cover}(G,k) = \textsc{Vertex Cover}(G-x,k-1)$$

Lemme [Buss]

Si G possède un VERTEX COVER de taille k, alors le graphe réduit possède au plus $k^2 + k$ sommets au plus k^2 arêtes.

Lemme [Buss]

Si G possède un VERTEX COVER de taille k, alors le graphe réduit possède au plus $k^2 + k$ sommets au plus k^2 arêtes.

Preuve

1. *Le graphe réduit possède au plus k^2 arêtes.*

Lemme [Buss]

Si G possède un Vertex Cover de taille k, alors le graphe réduit possède au plus $k^2 + k$ sommets au plus k^2 arêtes.

Preuve

1. *Le graphe réduit possède au plus k^2 arêtes.*

   Si S est un Vertex Cover, alors toute arête est incidente à un sommet de S.
   Or $d(x) \leqslant k$ et $|S| \leqslant k \Rightarrow k^2$ arêtes au plus

Lemme [Buss]

Si G possède un VERTEX COVER de taille k, alors le graphe réduit possède au plus $k^2 + k$ sommets au plus k^2 arêtes.

Preuve

1. *Le graphe réduit possède au plus k^2 arêtes.*
2. *Le graphe réduit possède au plus $k^2 + k$ sommets.*

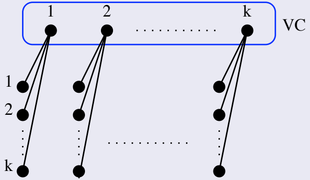
Lemme [Buss]

Si G possède un VERTEX COVER de taille k, alors le graphe réduit possède au plus $k^2 + k$ sommets au plus k^2 arêtes.

Preuve

1. *Le graphe réduit possède au plus k^2 arêtes.*
2. *Le graphe réduit possède au plus $k^2 + k$ sommets.*

   S possède au plus k sommets de degré $\leqslant k$
   $\Rightarrow$ il y a au plus $k^2 + k$ sommets



Définition

Soit (Q, κ) un problème paramétré sur l'alphabet Σ

Une fonction calculable en temps polynomiale

$$K : \Sigma^* \to \Sigma^*$$

est une **kernalisation** de (Q, κ) s'il existe une fonction

$$h : \mathbb{N} \to \mathbb{N}$$

telle que pour tout $x \in \Sigma^*$, on a

$$(x \in Q \Leftrightarrow K(x) \in Q)$$

$$|K(x)| \leqslant h(\kappa(x)) \qquad \text{et} \qquad \kappa(K(x)) \leqslant \kappa(x)$$

Théorème

Pour tout problème paramétré (Q, κ), les conditions suivantes sont équivalentes

1. $(Q, \kappa) \in$ **FPT**
2. Q est décidable et (Q, κ) possède un kernel.

Théorème

Pour tout problème paramétré (Q, κ), les conditions suivantes sont équivalentes

1. $(Q, \kappa) \in$ **FPT**
2. Q est décidable et (Q, κ) possède un kernel.

Preuve

- (2) $\Rightarrow$ (1) : Soit K la kernalisation de (Q, κ). Considérons l'algorithme $\mathcal{A}$ suivant:
  1. calculer $K(x)$
  2. décider si $K(x) \in Q$ avec un algorithme $\mathcal{A}'$

Théorème

Pour tout problème paramétré (Q, κ), les conditions suivantes sont équivalentes

1. $(Q, \kappa) \in$ **FPT**
2. Q est décidable et (Q, κ) possède un kernel.

Preuve

- $(2) \Rightarrow (1)$: Soit K la kernalisation de (Q, κ). Considérons l'algorithme $\mathcal{A}$ suivant:
  1. calculer $K(x)$
  2. décider si $K(x) \in Q$ avec un algorithme $\mathcal{A}'$

Le calcul de $K(x)$ se fait en $|x|^{O(1)}$
L'algorithme $\mathcal{A}'$ coûte $|K(x)|^{O(1)}$

Théorème

Pour tout problème paramétré (Q, κ), les conditions suivantes sont équivalentes

1. $(Q, \kappa) \in$ **FPT**
2. Q est décidable et (Q, κ) possède un kernel.

Preuve

- $(2) \Rightarrow (1)$: Soit K la kernalisation de (Q, κ). Considérons l'algorithme $\mathcal{A}$ suivant:
  1. calculer $K(x)$
  2. décider si $K(x) \in Q$ avec un algorithme $\mathcal{A}'$

Le calcul de $K(x)$ se fait en $|x|^{O(1)}$
L'algorithme $\mathcal{A}'$ coûte $|K(x)|^{O(1)}$
$\Rightarrow$ Puisque $|K(x| \leqslant h(\kappa(x))$, l'algorithme $\mathcal{A}$ est **FPT**.

Théorème

Pour tout problème paramétré (Q, κ), les conditions suivantes sont équivalentes

1. $(Q, \kappa) \in$ **FPT**
2. Q est décidable et (Q, κ) possède un kernel.

Preuve

- (1) $\Rightarrow$ (2) : Soit $\mathcal{A}$ un algorithme **FPT** pour (Q, κ) de complexité $f(\kappa(|x|)).p(|x|)$

Théorème

Pour tout problème paramétré (Q, κ), les conditions suivantes sont équivalentes

1. $(Q, \kappa) \in$ **FPT**
2. Q est décidable et (Q, κ) possède un kernel.

Preuve

- (1) $\Rightarrow$ (2) : Soit $\mathcal{A}$ un algorithme **FPT** pour (Q, κ) de complexité $f(\kappa(|x|)).p(|x|)$

  Si $Q = \emptyset$ ou $Q = \Sigma^*$, alors (Q, κ) a une kernalisation triviale. Soit $x_1 \in Q$ et $x_0 \in \Sigma^* \setminus Q$

  Soit l'algorithme $\mathcal{A}'$ suivant :

Théorème

Pour tout problème paramétré (Q, κ), les conditions suivantes sont équivalentes

1. $(Q, \kappa) \in$ **FPT**
2. Q est décidable et (Q, κ) possède un kernel.

Preuve

- $(1) \Rightarrow (2)$: Soit $\mathcal{A}$ un algorithme **FPT** pour (Q, κ) de complexité $f(\kappa(|x|)).p(|x|)$

  Si $Q = \emptyset$ ou $Q = \Sigma^*$, alors (Q, κ) a une kernalisation triviale. Soit $x_1 \in Q$ et $x_0 \in \Sigma^* \setminus Q$

  Soit l'algorithme $\mathcal{A}'$ suivant :

  1. Exécuter les $p(|x|).|x|$ premières étapes de $\mathcal{A}$
  2. $\mathcal{A}$ stoppe et accepte (réfute) $x \Rightarrow K(x) = x_1$ $(K(x) = x_0)$
  3. Sinon ($\mathcal{A}$ ne s'arrête pas en $p(|x|).|x|$ étapes) $K(x) = x$

Théorème

Pour tout problème paramétré (Q, κ), les conditions suivantes sont équivalentes

1. $(Q, \kappa) \in$ **FPT**
2. Q est décidable et (Q, κ) possède un kernel.

Preuve

- (1) $\Rightarrow$ (2) :
  1. Exécuter les $p(|x|).|x|$ premières étapes de $\mathcal{A}$
  2. $\mathcal{A}$ stoppe et accepte (réfute) $x \Rightarrow K(x) = x_1$ $(K(x) = x_0)$
  3. Sinon ($\mathcal{A}$ ne s'arrête pas en $p(|x|).|x|$ étapes) $K(x) = x$

  K est bien une kernalisation:
  - $K(x)$ est calculé en temps polynomial
  - $|K(x)| \leqslant |x_0| + |x_1| + f(\kappa(x))$ car

$$p(|x|).|x| \leqslant f(\kappa(x)).p(|x|))$$

Théorème

VERTEX COVER ne possède pas de noyau formé par un graphe de taille $1,36k$ sommets à moins que $\mathbf{P} \; = \mathbf{NP}$.

Théorème

VERTEX COVER ne possède pas de noyau formé par un graphe de taille $1,36k$ sommets à moins que $\mathbf{P} \ = \mathbf{NP}$.

Preuve

Supposons qu'un tel noyau existe. Alors cet ensemble de $1,36k$ sommets serait une approximation polynomiale de la solution optimale.
$\Rightarrow$ Impossible par des résultats de la théorie de l'approximabilité (Théorème PCP), à moins que $\mathbf{P} \ = \mathbf{NP}$.

Vertex Cover et programmation linéaire

Soit un graphe $G = (V, E)$. Alors le programme linéaire $L_{vc}(G)$ a une solution optimale semi-entière.

$$L_{vc}(G) = \min \sum_{v \in V} x_v \text{ tel que } \begin{cases} x_u + x_v \geqslant 1 & \forall uv \in E \\ 0 \leqslant x_v \leqslant 1 & \forall v \in V \end{cases}$$

Vertex Cover et programmation linéaire

Soit un graphe $G = (V, E)$. Alors le programme linéaire $L_{vc}(G)$ a une solution optimale semi-entière.

$$L_{vc}(G) = \min \sum_{v \in V} x_v \text{ tel que } \left\{ \begin{array}{ll} x_u + x_v \geqslant 1 & \forall uv \in E \\ 0 \leqslant x_v \leqslant 1 & \forall v \in V \end{array} \right.$$

Soit $(x_v)_{v \in V}$ une solution optimale demi-entière de $L_{vc}(G)$. Pour $r \in \{0, \frac{1}{2}, 1\}$, on note

- $V_r = \{v \in V \mid x_v = r\}$ et
- $G_r = G[V_r]$.

Lemme

Soient un graphe $G = (V, E)$ et $(x_v)_{v \in V}$ une solution optimale demi-entière de $L_{vc}(G)$. Alors

1. $VC(G_{\frac{1}{2}}) \geqslant |V_{\frac{1}{2}}| \ / \ 2$
2. $VC(G_{\frac{1}{2}}) = VC(G) - |V_1|$

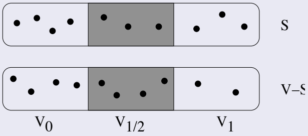
Lemme

Soient un graphe $G = (V, E)$ et $(x_v)_{v \in V}$ une solution optimale demi-entière de $L_{vc}(G)$. Alors

1. $VC(G_{\frac{1}{2}}) \geqslant |V_{\frac{1}{2}}| \ / \ 2$
2. $VC(G_{\frac{1}{2}}) = VC(G) - |V_1|$

Preuve

1. (i) Si S est un V.C. de G, alors $S_r = S \cap V_r$ est un V.C. de G_r.



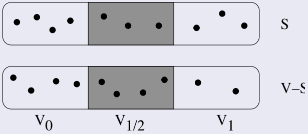
Lemme

Soient un graphe $G = (V, E)$ et $(x_v)_{v \in V}$ une solution optimale demi-entière de $L_{vc}(G)$. Alors

1. $VC(G_{\frac{1}{2}}) \geqslant |V_{\frac{1}{2}}| \ / \ 2$
2. $VC(G_{\frac{1}{2}}) = VC(G) - |V_1|$

Preuve

1. (i) Si S est un V.C. de G, alors $S_r = S \cap V_r$ est un V.C. de G_r.



(ii) Si S' est un V.C. de $G_{\frac{1}{2}}$, alors $S' \cup V_1$ est un V.C. de G.

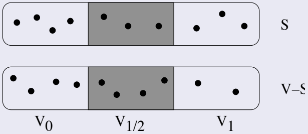
Lemme

Soient un graphe $G = (V, E)$ et $(x_v)_{v \in V}$ une solution optimale demi-entière de $L_{vc}(G)$. Alors

1. $VC(G_{\frac{1}{2}}) \geqslant |V_{\frac{1}{2}}| \ / \ 2$
2. $VC(G_{\frac{1}{2}}) = VC(G) - |V_1|$

Preuve

1. (i) Si S est un V.C. de G, alors $S_r = S \cap V_r$ est un V.C. de G_r.



(ii) Si S' est un V.C. de $G_{\frac{1}{2}}$, alors $S' \cup V_1$ est un V.C. de G.

$$|S'| + |V_1| \geqslant VC(G) \geqslant \sum_{v \in V} x_v = \frac{1}{2}|V_{\frac{1}{2}}| + |V_1|$$

Théorème

k-VERTEX COVER possède un kernel de taille au plus $2k$.

Théorème

k-VERTEX COVER possède un kernel de taille au plus $2k$.

Preuve

- $k - |V_1| < 0$: G ne possde pas de VC de taille k

Théorème

k-VERTEX COVER possède un kernel de taille au plus $2k$.

Preuve

- $k - |V_1| < 0$: G ne possde pas de VC de taille k
- $k - |V_1| = 0$: Si $G_{\frac{1}{2}}$ possède une arête alors G ne possède pas de solution sinon G possède un VC de taille k

Théorème

k-Vertex Cover possède un kernel de taille au plus $2k$.

Preuve

- $k - |V_1| < 0$: G ne possde pas de VC de taille k
- $k - |V_1| = 0$: Si $G_{\frac{1}{2}}$ possède une arête alors G ne possède pas de solution sinon G possède un VC de taille k
- $k - |V_1| > 0$ et $|V_{\frac{1}{2}} > 2(k - |V_1|)$: $G_{\frac{1}{2}}$ ne possède pas de VC de taille k (G non plus)

Théorème

k-VERTEX COVER possède un kernel de taille au plus $2k$.

Preuve

- $k - |V_1| < 0$: G ne possde pas de VC de taille k
- $k - |V_1| = 0$: Si $G_{\frac{1}{2}}$ possède une arête alors G ne possède pas de solution sinon G possède un VC de taille k
- $k - |V_1| > 0$ et $|V_{\frac{1}{2}} > 2(k - |V_1|)$: $G_{\frac{1}{2}}$ ne possède pas de VC de taille k (G non plus)
- Sinon, le noyau est l'instance $(G_{\frac{1}{2}}, k - |V_1|)$.

3-Hitting Set

Soient S un ensemble et $\mathcal{T}$ une famille de sous-ensembles de S telle que $\forall T \in \mathcal{T}$, $|T| \leqslant 3$. Existe-t-il un sous-ensemble $H \subseteq S$ de taille k tel que $\forall T \in \mathcal{T}$, $H \cap T \neq \emptyset$.

3-Hitting Set

Soient S un ensemble et $\mathcal{T}$ une famille de sous-ensembles de S telle que $\forall T \in \mathcal{T}$, $|T| \leqslant 3$. Existe-t-il un sous-ensemble $H \subseteq S$ de taille k tel que $\forall T \in \mathcal{T}$, $H \cap T \neq \emptyset$.

Théorème

Il existe des règles de réduction de complexité $O(kn + k^2)$, qui étant donnée une instance $\mathcal{I} = (S, \mathcal{C}, k)$ de 3-Hitting Set, avec $|S| = n$, soit

- calculent un noyau $(K, \mathcal{T}/K, k)$ tel que $K \subseteq S$ et $|K| \leqslant 6k^2$
- ou prouvent que $\mathcal{I}$ n'admet pas de solution.

Noyau quadratique pour 3-Hitting Set

1. Calculer une collection maximale $\mathcal{C} \subseteq \mathcal{T}$ de triplets deux à deux disjoints. Soit $G = \bigcup_{T \in \mathcal{C}} T$

Noyau quadratique pour 3-Hitting Set

1. Calculer une collection maximale $\mathcal{C} \subseteq \mathcal{T}$ de triplets deux à deux disjoints. Soit $G = \bigcup_{T \in \mathcal{C}} T$
   - Si $|\mathcal{C}| > k$, $\mathcal{I}$ est une instance négative;
   - Sinon G est de taille au plus $3k$.

Noyau quadratique pour 3-Hitting Set

1. Calculer une collection maximale $\mathcal{C} \subseteq \mathcal{T}$ de triplets deux à deux disjoints. Soit $G = \bigcup_{T \in \mathcal{C}} T$
2. $\forall x \in G$, $\mathcal{T}_x = \{T \in \mathcal{C} \mid x \in T\}$ et $\mathcal{T}'_x = \{T \setminus \{x\} \mid T \in \mathcal{T}_x\}$
   $\Rightarrow \mathcal{I}_x = (S \setminus \{x\}, \mathcal{T}'_x, k)$ est une instance de Vertex Cover

Noyau quadratique pour 3-HITTING SET

1. Calculer une collection maximale $\mathcal{C} \subseteq \mathcal{T}$ de triplets deux à deux disjoints. Soit $G = \bigcup_{T \in \mathcal{C}} T$
2. $\forall x \in G$, $\mathcal{T}_x = \{T \in \mathcal{C} \mid x \in T\}$ et $\mathcal{T}'_x = \{T \setminus \{x\} \mid T \in \mathcal{T}_x\}$
   $\Rightarrow \mathcal{I}_x = (S \setminus \{x\}, \mathcal{T}'_x, k)$ est une instance de VERTEX COVER
3. $\forall x \in G$,
   1. si $\mathcal{I}_x$ est une instance positive de VERTEX COVER, alors $\mathcal{I}_x$ possède un noyau K_x de taille $2k$
   2. sinon x doit appartenir à toute solution H de $\mathcal{I}$.

Noyau quadratique pour 3-HITTING SET

1. Calculer une collection maximale $\mathcal{C} \subseteq \mathcal{T}$ de triplets deux à deux disjoints. Soit $G = \bigcup_{T \in \mathcal{C}} T$
2. $\forall x \in G$, $\mathcal{T}_x = \{T \in \mathcal{C} \mid x \in T\}$ et $\mathcal{T}'_x = \{T \setminus \{x\} \mid T \in \mathcal{T}_x\}$
   $\Rightarrow \mathcal{I}_x = (S \setminus \{x\}, \mathcal{T}'_x, k)$ est une instance de VERTEX COVER
3. $\forall x \in G$,
   1. si $\mathcal{I}_x$ est une instance positive de VERTEX COVER, alors $\mathcal{I}_x$ possède un noyau K_x de taille $2k$
      $\Rightarrow K_x \subset M$
   2. sinon x doit appartenir à toute solution H de $\mathcal{I}$.
      $\Rightarrow x \in F$

$$|F \cup M| \leqslant 6k^2$$

4. $\mathcal{I}' = (K, \mathcal{C}/K, k)$ **est une noyau quadratique pour** $\mathcal{I}$

Problèmes ouverts (IWPEC 2006)

1. CLIQUE COVER : Etant donné un graphe G et un paramètre k, existe-t'il k cliques couvrant l'ensemble des arètes
   Question : Ce problème admet-il un kernel de taille polynomiale en k?

Problèmes ouverts (IWPEC 2006)

1. Clique Cover : Etant donné un graphe G et un paramètre k, existe-t'il k cliques couvrant l'ensemble des arètes
   Question : Ce problème admet-il un kernel de taille polynomiale en k?
2. Feedback Vertex Set : Etant donné un graphe G et un paramètre k, existe-t'il k sommets tels que leur suppression crée un graphe sans cycle ?
   Question : Ce problème admet-il un kernel de taille linéaire en k ? (connu k^3, linéaire dans les planaires)