Complexité paramétrée (1)
Introduction

(Méthodes Exactes - GMIN30C)

Christophe PAUL
(CNRS - LIRMM)

20 Septembre 2013

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.

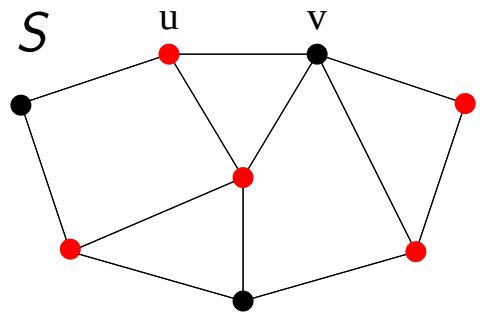
Un peu d'histoire de la complexité : NP-Complétude

- Théorème de Cook (1971) : SAT est NP-COMPLET
  (2-SAT $\in$ **P**)
- 21 problèmes NP-complets de Karp (1972) dont

Idée "admise" : les problèmes NP-complets sont tous équivalents !

Un peu d'histoire de la complexité : NP-Complétude

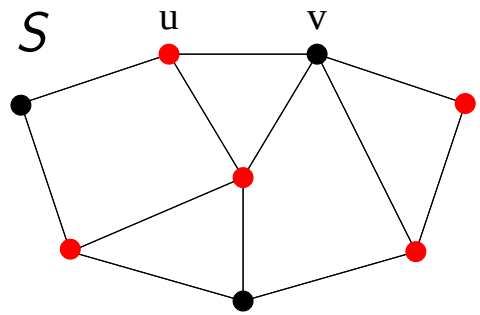
- Théorème de Cook (1971) : SAT est NP-COMPLET
  (2-SAT $\in$ **P**)
- 21 problèmes NP-complets de Karp (1972) dont
  - VERTEX COVER : Existe-t'il un ensemble S de k sommets couvrant toutes les arêtes d'un graphe ?



Idée "admise" : les problèmes NP-complets sont tous équivalents !

Un peu d'histoire de la complexité : NP-Complétude

- Théorème de Cook (1971) : SAT est NP-COMPLET
  (2-SAT $\in$ **P**)
- 21 problèmes NP-complets de Karp (1972) dont
  - VERTEX COVER : Existe-t'il un ensemble S de k sommets couvrant toutes les arêtes d'un graphe ?

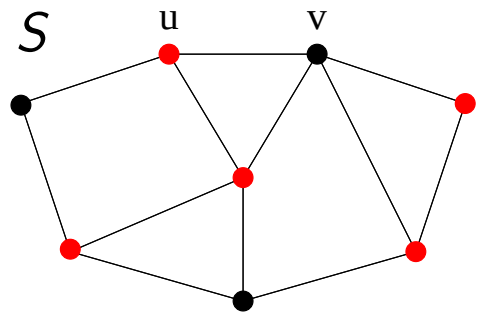


  - ENSEMBLE INDÉPENDANT : Existe-t'il un ensemble S de k sommets deux à deux non-adjacents dans un graphe ?

Idée "admise" : les problèmes NP-complets sont tous équivalents !

Un peu d'histoire de la complexité : NP-Complétude

- Théorème de Cook (1971) : SAT est NP-COMPLET
  (2-SAT $\in$ **P**)
- 21 problèmes NP-complets de Karp (1972) dont
  - VERTEX COVER : Existe-t'il un ensemble S de k sommets couvrant toutes les arêtes d'un graphe ?



  - ENSEMBLE INDÉPENDANT : Existe-t'il un ensemble S de k sommets deux à deux non-adjacents dans un graphe ?
  - COLORATION : Les sommets d'un graphe peuvent-ils être colorés avec k couleurs de sorte que pour toute arête xy, la couleur de x et celle de y sont différentes ?

Idée "admise" : les problèmes NP-complets sont tous équivalents !

Difficulté des problèmes NP-Complets

Observation :

- Le problème Coloration est NP-Complet pour $k = 3$.
- Les problèmes Vertex Cover et Ensemble Indépendant sont polynomiaux pour k fixé : algorithme naïf en $O(n^k)$.

Difficulté des problèmes NP-Complets

Observation :

- Le problème Coloration est NP-Complet pour $k = 3$.
- Les problèmes Vertex Cover et Ensemble Indépendant sont polynomiaux pour k fixé : algorithme naïf en $O(n^k)$.

Règle de branchement : soient (G, k) une instance et $e = (u, v)$ une arête de G, alors brancher sur

$$(G - u, k - 1) \text{ et } (G - v, k - 1)$$

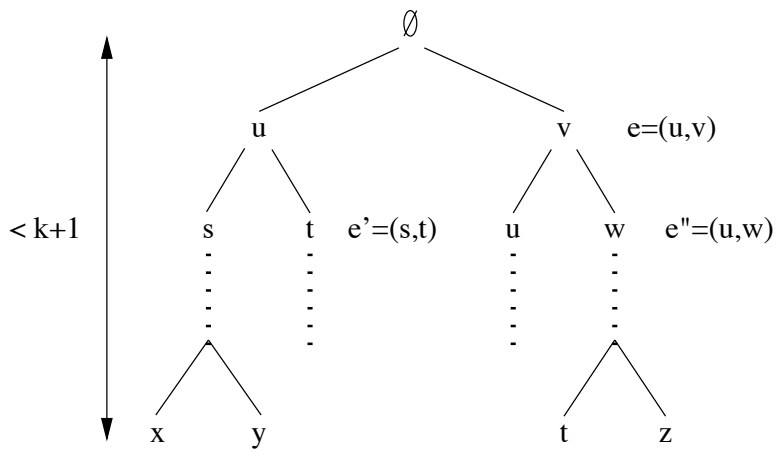
Difficulté des problèmes NP-Complets

Observation :

- Le problème Coloration est NP-Complet pour $k = 3$.
- Les problèmes Vertex Cover et Ensemble Indépendant sont polynomiaux pour k fixé : algorithme naïf en $O(n^k)$.

Règle de branchement : soient (G, k) une instance et $e = (u, v)$ une arête de G, alors brancher sur

$$(G - u, k - 1) \text{ et } (G - v, k - 1)$$



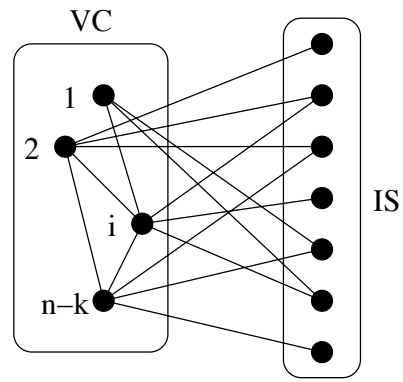
Vertex Cover : $O(2^k.(m + n))$

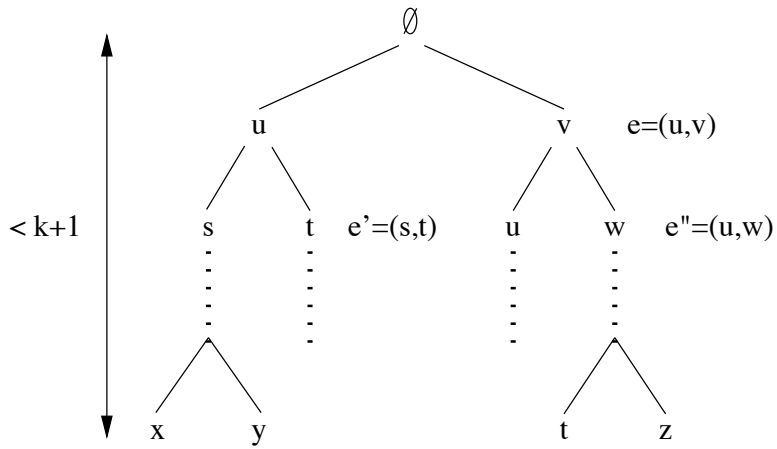
Difficulté des problèmes NP-Complets

Observation :

- Le problème Coloration est NP-Complet pour $k = 3$.
- Les problèmes Vertex Cover et Ensemble Indépendant sont polynomiaux pour k fixé : algorithme naïf en $O(n^k)$.

Observation : G possède un Vertex Cover de taille $k \Leftrightarrow G$ possède un indépendant de taille $n - k$





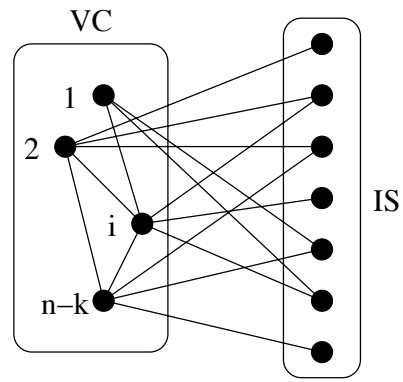
Vertex Cover : $O(2^k.(m+n))$

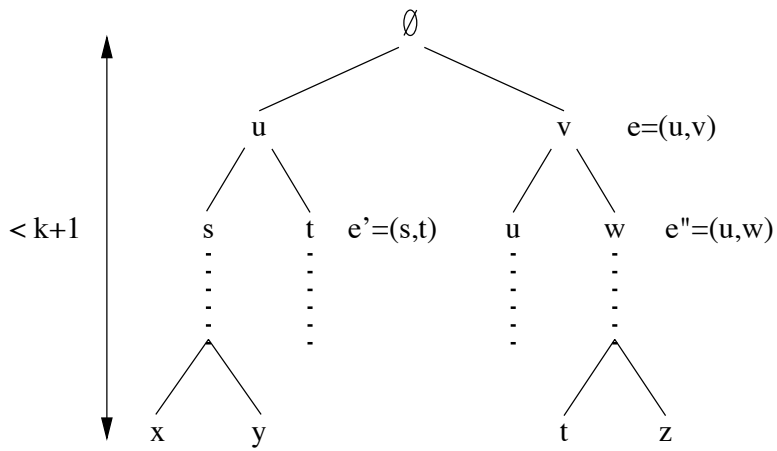
Difficulté des problèmes NP-Complets

Observation :

- Le problème Coloration est NP-Complet pour $k = 3$.
- Les problèmes Vertex Cover et Ensemble Indépendant sont polynomiaux pour k fixé : algorithme naïf en $O(n^k)$.

Observation : G possède un Vertex Cover de taille $k \Leftrightarrow G$ possède un indépendant de taille $n - k$





Vertex Cover : $O(2^k.(m+n))$

Ensemble Indépendant : $O(2^{(n-k)}.(m+n))$

Difficulté des problèmes NP-Complets

On constate donc que :

- k-Coloration appartient à la classe **Para-NP-Complet**
- k-Ensemble Indépendant appartient à la classe **XP**
- k-Vertex Cover appartient à la classe **FPT**

Difficulté des problèmes NP-Complets

On constate donc que :

- k-Coloration appartient à la classe **Para-NP-Complet**
- k-Ensemble Indépendant appartient à la classe **XP**
- k-Vertex Cover appartient à la classe **FPT**

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

J. Flum and M. Grohe, *Parameterized Complexity*, 2006.

"L'idée fondamentale [de la complexité paramétrée] 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, *Invitation to fixed parameter algorithms*, 2006.

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...

Complexité paramétrée

Mesure de la complexité en fonction de

- la taille n de la donnée
- un paramètre k (indépendant de n) :

$\longrightarrow$ Une paramétrisation d'un problème associe à chaque instance un paramètre k.

Complexité paramétrée

Mesure de la complexité en fonction de

- la taille n de la donnée
- un paramètre k (indépendant de n) :
  - la taille de la solution (paramètre naturel)

$\longrightarrow$ Une paramétrisation d'un problème associe à chaque instance un paramètre k.

Complexité paramétrée

Mesure de la complexité en fonction de

- la taille n de la donnée
- un paramètre k (indépendant de n) :
  - la taille de la solution (paramètre naturel)
  - le degré maximum, la largeur arborescente, la taille d'un coupe-cycle minimum... (paramètres structuraux)

$\longrightarrow$ Une paramétrisation d'un problème associe à chaque instance un paramètre k.

Complexité paramétrée

Mesure de la complexité en fonction de

- la taille n de la donnée
- un paramètre k (indépendant de n) :
  - la taille de la solution (paramètre naturel)
  - le degré maximum, la largeur arborescente, la taille d'un coupe-cycle minimum... (paramètres structuraux)

$\longrightarrow$ Une paramétrisation d'un problème associe à chaque instance un paramètre k.

Définition : Un problème paramétré est **FPT** (Fixed Parameter Tractable) s'il est résolu par un algorithme de complexité $f(k) \cdot n^{O(1)}$

Complexité paramétrée

Mesure de la complexité en fonction de

- la taille n de la donnée
- un paramètre k (indépendant de n) :
  - la taille de la solution (paramètre naturel)
  - le degré maximum, la largeur arborescente, la taille d'un coupe-cycle minimum... (paramètres structuraux)

$\longrightarrow$ Une paramétrisation d'un problème associe à chaque instance un paramètre k.

Définition : Un problème paramétré est **FPT** (Fixed Parameter Tractable) s'il est résolu par un algorithme de complexité $f(k) \cdot n^{O(1)}$

Observation : La fonction suivante est valide :

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

Problèmes, algorithmes paramétrés

Définition

Soit Σ un alphabet fini.

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

Définition

Soit Σ un alphabet fini.

1. Une **paramétrisation** de Σ^* est une fonction $\kappa : \Sigma^* \to \mathbb{N}$ (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 $\kappa : \Sigma^* \to \mathbb{N}$ (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 $\kappa : \Sigma^* \to \mathbb{N}$ 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 :

$$f(\kappa(x)).n^{O(1)}$$

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 $\kappa : \Sigma^* \to \mathbb{N}$ 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.

Observation

Tout problème $\Pi \in \mathbf{P}$ est **FPT**.

Définition

Soit (Q, κ) un problème paramétré et $t \in \mathbb{N}$. 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 $t \in \mathbb{N}$. 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 $(Q, \kappa)_t \in \mathbf{P}$.

Définition

Soit (Q, κ) un problème paramétré et $t \in \mathbb{N}$. 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 $(Q, \kappa)_t \in \mathbf{P}$.

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

Définition

Soit (Q, κ) un problème paramétré et $t \in \mathbb{N}$. 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$=$ $(\text{COLORATION}, \kappa)_3$ est **NP**-complet

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

Définition alternative

Soit (Q, κ) un problème paramétré. Les assertions suivantes sont équivalentes

1. $(Q, \kappa) \in \mathbf{FPT}$
2. Il existe un algorithme de décision pour $x \in Q$ de complexité
$$g(\kappa(x)) + f(\kappa(x)) \cdot p(|x| + \kappa(x))$$
$f(.)$ et $g(.)$ des fonctions calculables et $p(.)$ un polynôme
3. Il existe un algorithme de décision pour $x \in Q$ de complexité
$$g(\kappa(x)) + p(|x|)$$
$g(.)$ une fonction calculable et $p(.)$ un polynôme

Preuve : Soient $|x| = n$ et $\kappa(x) = k$

Définition alternative

Soit (Q, κ) un problème paramétré. Les assertions suivantes sont équivalentes

1. $(Q, \kappa) \in \mathbf{FPT}$
2. Il existe un algorithme de décision pour $x \in Q$ de complexité
$$g(\kappa(x)) + f(\kappa(x)) \cdot p(|x| + \kappa(x))$$
$f(.)$ et $g(.)$ des fonctions calculables et $p(.)$ un polynôme
3. Il existe un algorithme de décision pour $x \in Q$ de complexité
$$g(\kappa(x)) + p(|x|)$$
$g(.)$ une fonction calculable et $p(.)$ un polynôme

Preuve : Soient $|x| = n$ et $\kappa(x) = k$

$3 \Rightarrow 2$ est trivial

Définition alternative

Soit (Q, κ) un problème paramétré. Les assertions suivantes sont équivalentes

1. $(Q, \kappa) \in \mathbf{FPT}$
2. Il existe un algorithme de décision pour $x \in Q$ de complexité
$$g(\kappa(x)) + f(\kappa(x)) \cdot p(|x| + \kappa(x))$$
$f(.)$ et $g(.)$ des fonctions calculables et $p(.)$ un polynôme
3. Il existe un algorithme de décision pour $x \in Q$ de complexité
$$g(\kappa(x)) + p(|x|)$$
$g(.)$ une fonction calculable et $p(.)$ un polynôme

Preuve : Soient $|x| = n$ et $\kappa(x) = k$

$2 \Rightarrow 1$ Soit $p(n) = n^d$

$$\begin{aligned} g(k) + f(k) \cdot (n+k)^d &\leqslant g(k) + f(k) \cdot n^d \cdot (k+1)^d \\ &\leqslant (g(k) + f(k) \cdot (k+1)^d) \cdot n^d \\ &\leqslant h(k) \cdot n^d \end{aligned}$$

Définition alternative

Soit (Q, κ) un problème paramétré. Les assertions suivantes sont équivalentes

1. $(Q, \kappa) \in \mathbf{FPT}$
2. Il existe un algorithme de décision pour $x \in Q$ de complexité
$$g(\kappa(x)) + f(\kappa(x)) \cdot p(|x| + \kappa(x))$$
$f(.)$ et $g(.)$ des fonctions calculables et $p(.)$ un polynôme
3. Il existe un algorithme de décision pour $x \in Q$ de complexité
$$g(\kappa(x)) + p(|x|)$$
$g(.)$ une fonction calculable et $p(.)$ un polynôme

Preuve : Soient $|x| = n$ et $\kappa(x) = k$

$1 \Rightarrow 3$ Il suffit d'observer que $f(k) \cdot p(n) \leqslant f(k)^2 + p(n)^2$

DNF-SAT

Une formule booléenne Φ est sous forme normale 3-disjonctive (3-DNF) si $\Phi = \bigvee_{i \in I} (\lambda_{i,1} \wedge \lambda_{i,2} \wedge \lambda_{i,3})$

1. MAX-3-DNF-SAT (problème d'optimisation)
   - *Donnée :* une formule 3-DNF Φ.
   - *Objectif :* maximiser $1 + t$, avec t le nbre de termes satisfaits.

DNF-SAT

Une formule booléenne Φ est sous forme normale 3-disjonctive (3-DNF) si $\Phi = \bigvee_{i \in I} (\lambda_{i,1} \wedge \lambda_{i,2} \wedge \lambda_{i,3})$

1. MAX-3-DNF-SAT (problème d'optimisation)
   - *Donnée :* une formule 3-DNF Φ.
   - *Objectif :* maximiser $1 + t$, avec t le nbre de termes satisfaits.
2. (MAX-3-DNF-SAT, k) (paramétrisation standard)
   - *Donnée :* une formule 3-DNF Φ.
   - *Question :* Existe-t-il une affectation des variables satisfaisant au moins k termes ?

DNF-SAT

Une formule booléenne Φ est sous forme normale 3-disjonctive (3-DNF) si $\Phi = \bigvee_{i \in I} (\lambda_{i,1} \wedge \lambda_{i,2} \wedge \lambda_{i,3})$

1. MAX-3-DNF-SAT (problème d'optimisation)
   - *Donnée :* une formule 3-DNF Φ.
   - *Objectif :* maximiser $1 + t$, avec t le nbre de termes satisfaits.
2. (MAX-3-DNF-SAT, k) (paramétrisation standard)
   - *Donnée :* une formule 3-DNF Φ.
   - *Question :* Existe-t-il une affectation des variables satisfaisant au moins k termes ?
3. (EXACT-MAX-3-DNF-SAT, k)
   - *Donnée :* une formule 3-DNF Φ.
   - *Question :* Existe-t-il une affectation des variables satisfaisant exactement $k - 1$ termes ?

DNF-SAT

Exercice 2 :

1. Montrer que $(\text{EXACT-MAX-3-DNF-SAT}, k)$ n'est pas **FPT** (sauf si **P** $=$ **NP**)

DNF-SAT

Exercice 2 :

1. Montrer que $(\text{EXACT-MAX-3-DNF-SAT}, k)$ n'est pas **FPT** (sauf si **P** $=$ **NP**)

   - *Montrer que c'est* **NP***-complet de décider s'il existe une affectation ne satisfaisant aucun terme*

DNF-SAT

Exercice 2 :

1. Montrer que $(\text{EXACT-MAX-3-DNF-SAT}, k)$ n'est pas **FPT** (sauf si **P** $=$ **NP**)

   - *Montrer que c'est* **NP***-complet de décider s'il existe une affectation ne satisfaisant aucun terme*

   Aucun terme n'est satisfait dans Φ

   $\Leftrightarrow$

   Tous les termes sont satistaits dans $\neg\Phi$

   $\hookrightarrow$ Réduction à 3-SAT

DNF-SAT

Exercice 2 :

1. Montrer que $(\text{EXACT-MAX-3-DNF-SAT}, k)$ n'est pas **FPT** (sauf si **P** $=$ **NP**)
2. Montrer que $(\text{MAX-3-DNF-SAT}, k)$ est **FPT**.

DNF-SAT

Exercice 2 :

1. Montrer que $(\text{EXACT-MAX-3-DNF-SAT}, k)$ n'est pas **FPT** (sauf si **P** $=$ **NP**)
2. Montrer que $(\text{MAX-3-DNF-SAT}, k)$ est **FPT**.
   - *Montrer que le nombre moyen de termes satisfaits dans une affectation aléatoire est au moins $\frac{m}{8}$ où m est le nombre total de termes*

DNF-SAT

Exercice 2 :

1. Montrer que $(\text{EXACT-MAX-3-DNF-SAT}, k)$ n'est pas **FPT** (sauf si **P** $=$ **NP**)
2. Montrer que $(\text{MAX-3-DNF-SAT}, k)$ est **FPT**.
   - *Montrer que le nombre moyen de termes satisfaits dans une affectation aléatoire est au moins $\frac{m}{8}$ où m est le nombre total de termes*

     Soit X_i v.a. binaire telle que $X_i = 1$ ssi le terme λ_i est vrai

$$P(X_i = 1) = \frac{1}{8} \Rightarrow \mathbb{E}[X = \sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} \mathbb{E}[X_i] = \frac{m}{8}$$

DNF-SAT

Exercice 2 :

1. Montrer que $(\text{EXACT-MAX-3-DNF-SAT}, k)$ n'est pas **FPT** (sauf si **P** $=$ **NP**)
2. Montrer que $(\text{MAX-3-DNF-SAT}, k)$ est **FPT**.
   - *Montrer que le nombre moyen de termes satisfaits dans une affectation aléatoire est au moins $\frac{m}{8}$ où m est le nombre total de termes*

     Soit X_i v.a. binaire telle que $X_i = 1$ ssi le terme λ_i est vrai

     $$P(X_i = 1) = \tfrac{1}{8} \Rightarrow \mathbb{E}[X = \sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} \mathbb{E}[X_i] = \frac{m}{8}$$
   - *Conclure que pour $m \geqslant 8k$, il existe une affectation qui satisfisait au moins k termes*

DNF-SAT

Exercice 2 :

1. Montrer que $(\text{EXACT-MAX-3-DNF-SAT}, k)$ n'est pas **FPT** (sauf si **P** $=$ **NP**)
2. Montrer que $(\text{MAX-3-DNF-SAT}, k)$ est **FPT**.
   - *Montrer que le nombre moyen de termes satisfaits dans une affectation aléatoire est au moins $\frac{m}{8}$ où m est le nombre total de termes*
     Soit X_i v.a. binaire telle que $X_i = 1$ ssi le terme λ_i est vrai
     $$P(X_i = 1) = \frac{1}{8} \Rightarrow \mathbb{E}[X = \sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} \mathbb{E}[X_i] = \frac{m}{8}$$
   - *Conclure que pour $m \geqslant 8k$, il existe une affectation qui satisfisait au moins k termes*
     Si $m \geqslant 8k$ alors $\mathbb{E}[X] \geqslant k$. Donc il existe affectation qui satisfait au moins k termes.

DNF-SAT

Exercice 2 :

1. Montrer que $(\text{EXACT-MAX-3-DNF-SAT}, k)$ n'est pas **FPT** (sauf si **P** $=$ **NP**)
2. Montrer que $(\text{MAX-3-DNF-SAT}, k)$ est **FPT**.
   - *Montrer que le nombre moyen de termes satisfaits dans une affectation aléatoire est au moins $\frac{m}{8}$ où m est le nombre total de termes*
     Soit X_i v.a. binaire telle que $X_i = 1$ ssi le terme λ_i est vrai
     $$P(X_i = 1) = \frac{1}{8} \Rightarrow \mathbb{E}[X = \sum_{i=1}^{m} X_i] = \sum_{i=1}^{m} \mathbb{E}[X_i] = \frac{m}{8}$$
   - *Conclure que pour $m \geqslant 8k$, il existe une affectation qui satisfisait au moins k termes*
     Si $m \geqslant 8k$ alors $\mathbb{E}[X] \geqslant k$. Donc il existe affectation qui satisfait au moins k termes.
   - *On peut donc supposer que $m < 8k$*
     Une recherche exhaustive permet de répondre à la question en temps $8^{8k}.n^{O(1)}$

Kernelization - réduction à un noyau

Observation : nous venons de montrer que en temps polynomial

- soit une instance de $(\text{MAX-3-DNF-SAT}, k)$ est positive ($m > 8k$)
- soit sa taille est bornée par une fonction (polynomiale) en k

Kernelization - réduction à un noyau

Observation : nous venons de montrer que en temps polynomial

- soit une instance de $(\text{MAX-3-DNF-SAT}, k)$ est positive ($m > 8k$)
- soit sa taille est bornée par une fonction (polynomiale) en k

Le problème $(\text{MAX-3-DNF-SAT}, k)$ admet un noyau (linéaire)

Kernelization - réduction à un noyau

Observation : nous venons de montrer que en temps polynomial

- soit une instance de $(\text{MAX-3-DNF-SAT}, k)$ est positive ($m > 8k$)
- soit sa taille est bornée par une fonction (polynomiale) en k

Le problème $(\text{MAX-3-DNF-SAT}, k)$ admet un noyau (linéaire)

Définition : Soit κ-**Q** un problème paramétré. Un algorithme polynomial qui étant donnée une instance (G, k) retourne une instance (G', k') est une kernelization s'il existe une fonction $h : \mathbb{N} \to \mathbb{N}$ telle que

- (G, k) est une instance positive $\Leftrightarrow$ (G', k') est une instance positive
- $|G| \leqslant h(k)$
- $k' \leqslant k$

Kernelization - réduction à un noyau

Théorème : Un problème paramétré κ-**Q** est **FPT** ssi il est décidable et admet un noyau.

Kernelization - réduction à un noyau

Théorème : Un problème paramétré κ-**Q** est **FPT** ssi il est décidable et admet un noyau.

Preuve

$\Rightarrow$ Soit $\mathcal{K}$ la kernelisation de κ-**Q**. Considérons l'algorithme $\mathcal{A}$

1. calculer $G' = \mathcal{K}(G)$ en temps polynomial en $|G|$
2. décider si $G' \in Q$ avec un algorithme $\mathcal{A}'$ exponentiel exact

Kernelization - réduction à un noyau

Théorème : Un problème paramétré κ-**Q** est **FPT** ssi il est décidable et admet un noyau.

Preuve

$\Rightarrow$ Soit $\mathcal{K}$ la kernelisation de κ-**Q**. Considérons l'algorithme $\mathcal{A}$

1. calculer $G' = \mathcal{K}(G)$ en temps polynomial en $|G|$
2. décider si $G' \in Q$ avec un algorithme $\mathcal{A}'$ exponentiel exact

$\Leftarrow$ Puisque $|G'| \leqslant h(\kappa(k))$, l'algorithme $\mathcal{A}$ est **FPT**.

Kernelization - réduction à un noyau

Théorème : Un problème paramétré κ-**Q** est **FPT** ssi il est décidable et admet un noyau.

Preuve

$\Rightarrow$ Soit $\mathcal{K}$ la kernelisation de κ-**Q**. Considérons l'algorithme $\mathcal{A}$

1. calculer $G' = \mathcal{K}(G)$ en temps polynomial en $|G|$
2. décider si $G' \in Q$ avec un algorithme $\mathcal{A}'$ exponentiel exact

$\Leftarrow$ Puisque $|G'| \leqslant h(\kappa(k))$, l'algorithme $\mathcal{A}$ est **FPT**.

$\Rightarrow$ Soit $\mathcal{A}$ un algorithme **FPT** pour (Q, κ) de complexité $f(k) \cdot n^c$ pour une constante $c > 0$

Kernelization - réduction à un noyau

Théorème : Un problème paramétré κ-**Q** est **FPT** ssi il est décidable et admet un noyau.

Preuve

$\Rightarrow$ Soit $\mathcal{K}$ la kernelisation de κ-**Q**. Considérons l'algorithme $\mathcal{A}$

1. calculer $G' = \mathcal{K}(G)$ en temps polynomial en $|G|$
2. décider si $G' \in Q$ avec un algorithme $\mathcal{A}'$ exponentiel exact

$\Leftarrow$ Puisque $|G'| \leqslant h(\kappa(k))$, l'algorithme $\mathcal{A}$ est **FPT**.

$\Rightarrow$ Soit $\mathcal{A}$ un algorithme **FPT** pour (Q, κ) de complexité $f(k) \cdot n^c$ pour une constante $c > 0$

- si $n = |G| \leqslant f(k)$, alors l'instance est déjà réduite

Kernelization - réduction à un noyau

Théorème : Un problème paramétré κ-**Q** est **FPT** ssi il est décidable et admet un noyau.

Preuve

$\Rightarrow$ Soit $\mathcal{K}$ la kernelisation de κ-**Q**. Considérons l'algorithme $\mathcal{A}$

1. calculer $G' = \mathcal{K}(G)$ en temps polynomial en $|G|$
2. décider si $G' \in Q$ avec un algorithme $\mathcal{A}'$ exponentiel exact

$\Leftarrow$ Puisque $|G'| \leqslant h(\kappa(k))$, l'algorithme $\mathcal{A}$ est **FPT**.

$\Rightarrow$ Soit $\mathcal{A}$ un algorithme **FPT** pour (Q, κ) de complexité $f(k) \cdot n^c$ pour une constante $c > 0$

- si $n = |G| \leqslant f(k)$, alors l'instance est déjà réduite
- sinon $f(k) \cdot n^c \leqslant n \cdot n^c = n^{c+1}$: donc $\mathcal{A}$ est polynomial en $|G|$

Kernelization - réduction à un noyau

Théorème : Un problème paramétré κ-**Q** est **FPT** ssi il est décidable et admet un noyau.

Preuve

$\Rightarrow$ Soit $\mathcal{K}$ la kernelisation de κ-**Q**. Considérons l'algorithme $\mathcal{A}$

1. calculer $G' = \mathcal{K}(G)$ en temps polynomial en $|G|$
2. décider si $G' \in Q$ avec un algorithme $\mathcal{A}'$ exponentiel exact

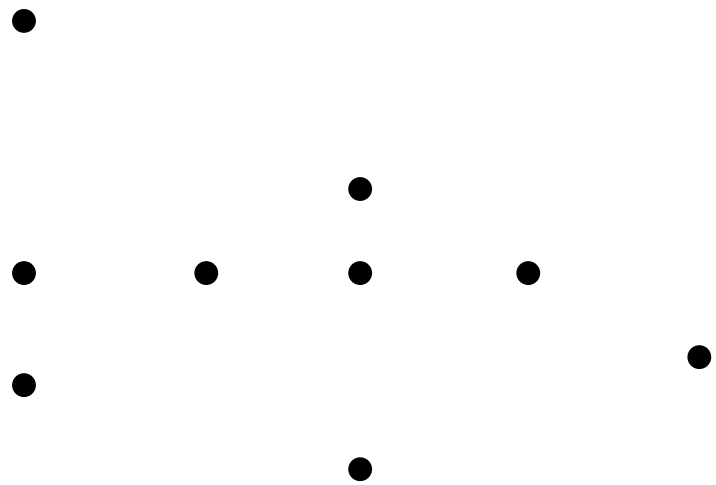
$\Leftarrow$ Puisque $|G'| \leqslant h(\kappa(k))$, l'algorithme $\mathcal{A}$ est **FPT**.

$\Rightarrow$ Soit $\mathcal{A}$ un algorithme **FPT** pour (Q, κ) de complexité $f(k) \cdot n^c$ pour une constante $c > 0$

- si $n = |G| \leqslant f(k)$, alors l'instance est déjà réduite
- sinon $f(k) \cdot n^c \leqslant n \cdot n^c = n^{c+1}$: donc $\mathcal{A}$ est polynomial en $|G|$

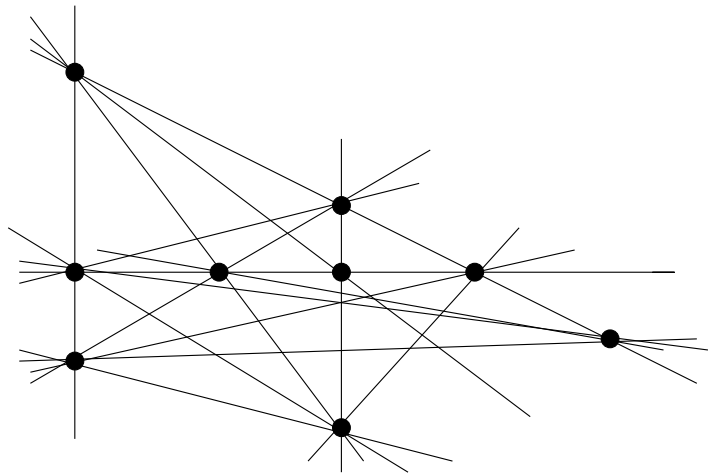
Observation : la taille du noyau obtenu est exponentiel en k

Noyau polynomial : un problème géométrique



Existe-t-il k lignes couvrant l'ensemble S de points ?

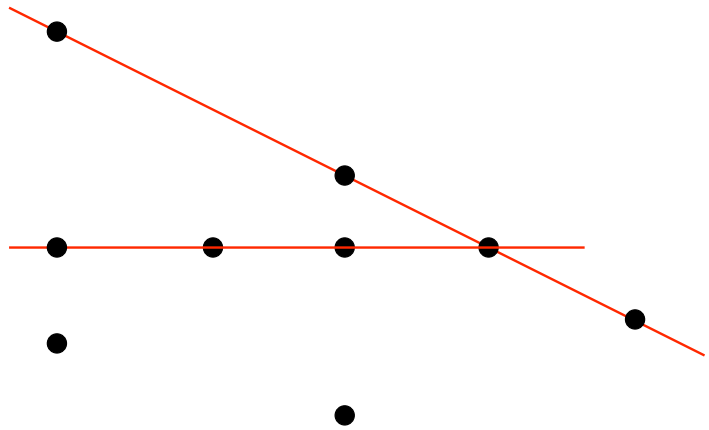
Noyau polynomial : un problème géométrique



Existe-t-il k lignes couvrant l'ensemble S de points ?

Observation 1 : On peut se restreindre aux lignes générées par les paires de points de S

Noyau polynomial : un problème géométrique

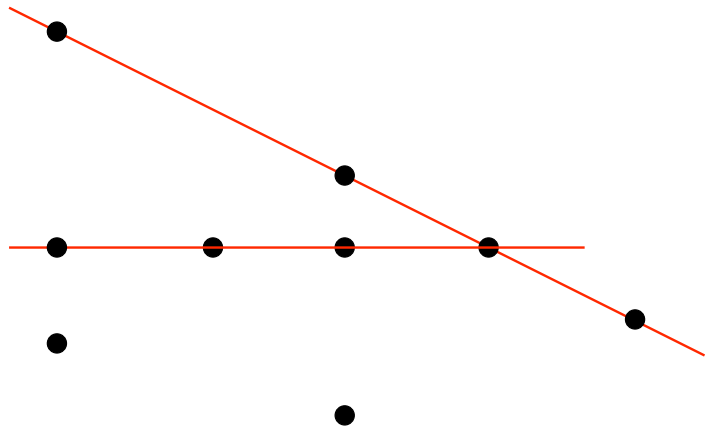


Existe-t-il k lignes couvrant l'ensemble S de points ?

Observation 2 : Si une ligne L contient au moins $k+1$ points, alors elle appartient à la solution (si elle existe) (e.g. ici $k = 3$)

$\Rightarrow$ supprimer L et soustraire 1 à k

Noyau polynomial : un problème géométrique



Existe-t-il k lignes couvrant l'ensemble S de points ?

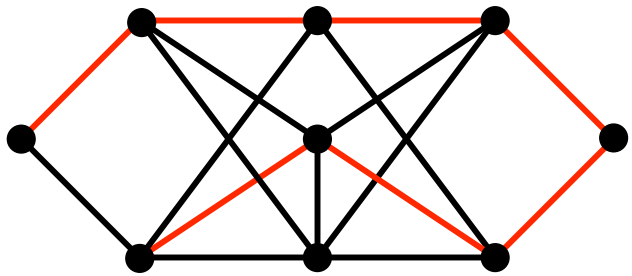
Observation 2 : Si une ligne L contient au moins $k+1$ points, alors elle appartient à la solution (si elle existe) (e.g. ici $k = 3$)

$\Rightarrow$ supprimer L et soustraire 1 à k

$\Rightarrow$ l'instance réduite contient au plus k^2 points.

Non-existence de noyau polynomial : Longest Path

- Un graphe $G = (V, E)$ et un paramètre $k \in \mathbb{N}$
- G contient-il un chemin de taille k ?

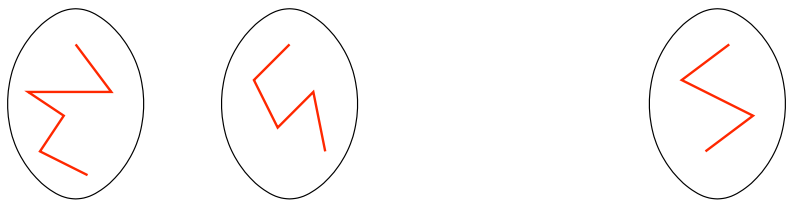


Longest Path est NP-Complet (réduction à Chemin Hamiltonien) mais peut-être résolu en temps $O(c^k.n^{O(1)})$ grâce à Color Coding.

Non-existence de noyau polynomial : Longest Path

Hypothèse : Il existe un algorithme de kernelization $\mathcal{A}$ pour Longest Path qui retourne un noyau polynomial de taille $t = k^c$ bits.

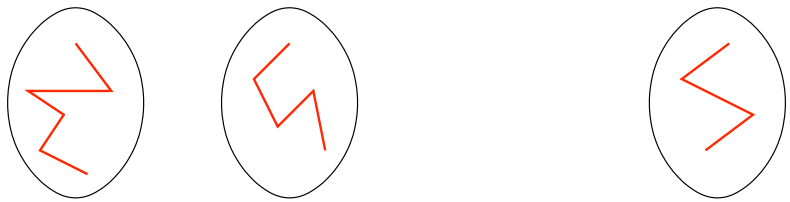
- construisons une instance (G, k) avec t instances différentes
$$(G, k) = (G_1, k) \oplus (G_2, k) \oplus \ \ldots \ \oplus (G_t, k)$$



Non-existence de noyau polynomial : Longest Path

Hypothèse : Il existe un algorithme de kernelization $\mathcal{A}$ pour Longest Path qui retourne un noyau polynomial de taille $t = k^c$ bits.

- construisons une instance (G, k) avec t instances différentes
$$(G, k) = (G_1, k) \oplus (G_2, k) \oplus \ \ldots \ \oplus (G_t, k)$$



Observation : (G, k) admet un chemin de longueur k ssi $\exists i$ tq G_i admet un chemin de taille k.

Question :
Est-il possible de décider si une des instances possède un chemin de longueur k en disposant de moins de 1 bits par instance en moyenne ?

Non-existence de noyau polynomial

Théorème : Sauf si $co - NP \subseteq NP/poly$, le problème paramétré Longest Path n'admet pas de noyau polynomial.

Non-existence de noyau polynomial

Théorème : Sauf si $co - NP \subseteq NP/poly$, le problème paramétré Longest Path n'admet pas de noyau polynomial.

Plusieurs outils existent pour démontrer des bornes inférieures sur les tailles de noyaux (sous des hypothèses de complexité classiques)

- OU-composition [Bodlaender et al.] et ET-composition [Drucker]
- Transformations polynomiales paramétrés [Bodlaender et al.]
- Composition croisée [Bodlaender et al.]
- ...

Synthèse

Nous avons donc constaté que (sous des hypothèses standards) certains problèmes NP-Complets :

- sont NP-Complets à paramètre fixé
  k-Coloration — **Para-NP-Complet**
- peuvent être résolus en temps $O(n^k)$
  k-Ensemble Indépendant — **XP**
- peuvent être résolus en temps $f(k) \cdot n^{O(1)}$
  k-Vertex Cover — **FPT**
- n'admettent pas de noyau polynomial
  k-Longest Path — **No-poly-Kernel**
- admettent un noyau polynomial
  k-Line-Cover — **poly-Kernel**

Synthèse

Nous avons donc constaté que (sous des hypothèses standards) certains problèmes NP-Complets (pour certaines paramétérisations) :

- sont NP-Complets à paramètre fixé
  k-Coloration — **Para-NP-Complet**
- peuvent être résolus en temps $O(n^k)$
  k-Ensemble Indépendant — **XP**
- peuvent être résolus en temps $f(k) \cdot n^{O(1)}$
  k-Vertex Cover — **FPT**
- n'admettent pas de noyau polynomial
  k-Longest Path — **No-poly-Kernel**
- admettent un noyau polynomial
  k-Line-Cover — **poly-Kernel**

Observation : Le problème Coloration peut être **FPT** pour d'autres paramétrisations !

Synthèse

Nous avons donc constaté que (sous des hypothèses standards) certains problèmes NP-Complets (pour certaines paramétérisations) :

- sont NP-Complets à paramètre fixé
  k-Coloration — **Para-NP-Complet**
- peuvent être résolus en temps $O(n^k)$
  k-Ensemble Indépendant — **XP**
- peuvent être résolus en temps $f(k) \cdot n^{O(1)}$
  k-Vertex Cover — **FPT**
- n'admettent pas de noyau polynomial
  k-Longest Path — **No-poly-Kernel**
- admettent un noyau polynomial
  k-Line-Cover — **poly-Kernel**

Question : Est-il possible de montrer que k-Ensemble Indépendant n'appartient pas à la classe **FPT** ?