

Complexité et algorithmes paramétrés (1)

-

Introduction

Christophe PAUL
CNRS - LIRMM

EJC Informatique Mathématique du GDR IM
Perpignan, avril 2013

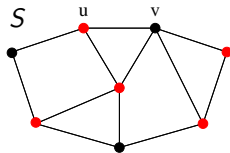
Un peu d'histoire de la complexité : NP-Complétude

- ▶ 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 ?



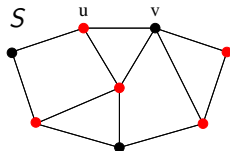
Idée "admise" : les problèmes NP-complets sont tous équivalents !

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 ?



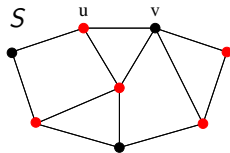
- ▶ **ENSEMBLE INDÉPENDANT** : Existe-t'il un ensemble S de k sommets deux à deux non-adjacents dans un graphe ?

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 :

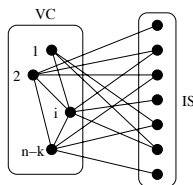
- ▶ Le problème COLORATION est NP-Complet pour $k = 3$.
- ▶ Les problèmes VERTEX COVER et ENSEMBLE INDÉPENDANT sont polynomiaux pour k fixé : algorithme naïf en $O(n^k)$.

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 $O(n^k)$** .

Observation : G possède un VERTEX COVER de taille $k \Leftrightarrow G$ possède un indépendant de taille $n - k$

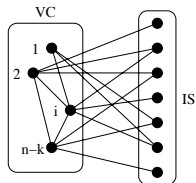


Difficulté des problèmes NP-Complets

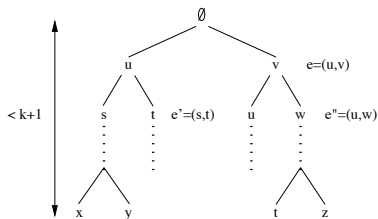
Observation :

- ▶ Le problème COLORATION est NP-Complet pour $k = 3$.
- ▶ Les problèmes VERTEX COVER et ENSEMBLE INDÉPENDANT sont polynomiaux pour k fixé : **algorithme naïf en $O(n^k)$** .

Observation : G possède un VERTEX COVER de taille $k \Leftrightarrow G$ possède un indépendant de taille $n - k$



Arbre de Recherche Bornée

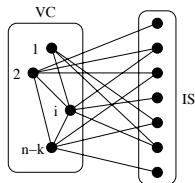


Difficulté des problèmes NP-Complets

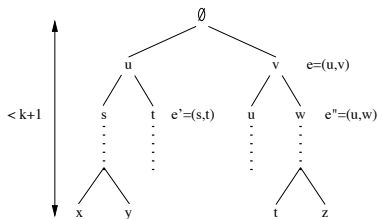
Observation :

- ▶ 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 $n - k$



Arbre de Recherche Bornée



VERTEX COVER : $O(2^k \cdot (m + n))$

ENSEMBLE INDÉPENDANT :

$O(2^{(n-k)} \cdot (m + n))$

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.

Complexité paramétrée

Mesure de la complexité en fonction de

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

→ 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)

→ 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)

→ 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)

→ 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)

→ 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 :

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

Complexité paramétrée

Soit $\kappa\text{-Q}$ un problème paramétré et $l \in \mathbb{N}$. Le **niveau** t de $\kappa\text{-Q}$ est le problème :

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

Observation : Soit $\kappa\text{-Q}$ un problème paramétré et $t \in \mathbb{N}$.
Si $\kappa\text{-Q}$ est **FPT**, alors $\kappa\text{-Q}_t \in \mathbf{P}$.

Complexité paramétrée

Soit $\kappa\text{-Q}$ un problème paramétré et $l \in \mathbb{N}$. Le **niveau** t de $\kappa\text{-Q}$ est le problème :

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

Observation : Soit $\kappa\text{-Q}$ un problème paramétré et $t \in \mathbb{N}$.
Si $\kappa\text{-Q}$ est **FPT**, alors $\kappa\text{-Q}_t \in \mathbf{P}$.

Exercice 1 : $k\text{-COLORATION}$ paramétré par k est-il **FPT** ?

Complexité paramétrée

Soit $\kappa\text{-Q}$ un problème paramétré et $l \in \mathbb{N}$. Le **niveau** t de $\kappa\text{-Q}$ est le problème :

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

Observation : Soit $\kappa\text{-Q}$ un problème paramétré et $t \in \mathbb{N}$.
Si $\kappa\text{-Q}$ est **FPT**, alors $\kappa\text{-Q}_t \in \mathbf{P}$.

Exercice 1 : $k\text{-COLORATION}$ paramétré par k est-il **FPT** ?

Le problème $3\text{-COLORATION} = (\text{COLORATION}, \kappa)_3$ est **NP-complet**
 $\Rightarrow k\text{-COLORATION}$ n'est pas **FPT**.

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 $\mathbf{P} = \mathbf{NP}$)

DNF-SAT

Exercice 2 :

1. Montrer que $(\text{EXACT-MAX-3-DNF-SAT}, k)$ n'est pas **FPT** (sauf si $\mathbf{P} = \mathbf{NP}$)
 - ▶ *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 $\mathbf{P} = \mathbf{NP}$)
 - ▶ *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 satisfait 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 $\mathbf{P} = \mathbf{NP}$)
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 $\mathbf{P} = \mathbf{NP}$)
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 $\mathbf{P} = \mathbf{NP}$)
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 $m \geq 8k$, il existe une 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 $\mathbf{P} = \mathbf{NP}$)
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 $m \geq 8k$, il existe une affectation qui satisfait au moins k termes*

Si $m \geq 8k$ alors $\mathbb{E}[X] \geq k$. 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 $\mathbf{P} = \mathbf{NP}$)
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 $m \geq 8k$, il existe une affectation qui satisfait au moins k termes*

Si $m \geq 8k$ alors $\mathbb{E}[X] \geq k$. 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 $8^{8k} \cdot n^{O(1)}$

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 $\kappa\text{-Q}$ 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 $h : \mathbb{N} \rightarrow \mathbb{N}$ telle que

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

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

⇒ 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

⇒ 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

⇐ Puisque $|G'| \leq h(\kappa(k))$, 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

⇒ 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

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

⇒ 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

⇒ 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

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

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

- ▶ si $n = |G| \leq f(k)$, 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

⇒ 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

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

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

- ▶ si $n = |G| \leq f(k)$, alors l'instance est déjà réduite
- ▶ sinon $f(k) \cdot n^c \leq n \cdot n^c = n^{c+1}$: 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

⇒ 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

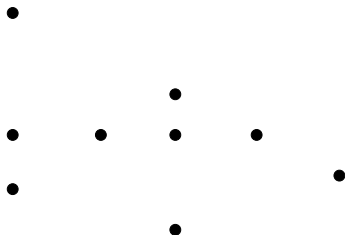
⇐ Puisque $|G'| \leq h(\kappa(k))$, l'algorithme \mathcal{A} est **FPT**.

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

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

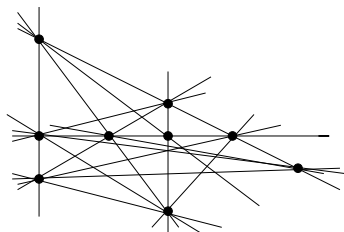
Observation : la taille du noyau obtenu est **exponentiel** en k

Noyau polynomial : un problème géométrique



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

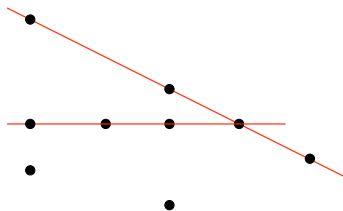
Noyau polynomial : un problème géométrique



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

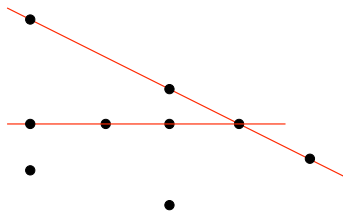


Existe-t-il k lignes couvrant l'ensemble S de points ?

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

⇒ supprimer L et soustraire 1 à k

Noyau polynomial : un problème géométrique



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

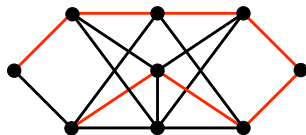
Observation 2 : Si une ligne L contient au moins $k + 1$ points, alors elle appartient à la solution (si elle existe) (e.g. ici $k = 3$)

⇒ supprimer L et soustraire 1 à k

⇒ 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-Comple**t (réduction à CHEMIN HAMILTONIEN)
mais peut-être résolu en temps $O(c^k \cdot n^{O(1)})$ 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
 $(G, k) = (G_1, k) \oplus (G_2, k) \oplus \dots \oplus (G_t, k)$



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 paramétrisations**) :

- ▶ 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 paramétrisations**) :

- ▶ 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** ?

Réductions paramétrées et W -hiérarchie (1)

Soient $\kappa\text{-P}$ et $\tau\text{-Q}$ deux problèmes paramétrés.

Une **réduction paramétrée** de $\kappa\text{-P}$ vers $\tau\text{-Q}$ est un algorithme $\mathcal{R} : (x, \kappa(x)) \rightarrow (y, \tau(y))$ tel que

1. $(x, \kappa(x))$ est positive $\Leftrightarrow (y, \tau(y))$ est positive,

On notera : $\kappa\text{-P} \leq_{fpt} \tau\text{-Q}$

Réductions paramétrées et W -hiérarchie (1)

Soient $\kappa\text{-P}$ et $\tau\text{-Q}$ deux problèmes paramétrés.

Une **réduction paramétrée** de $\kappa\text{-P}$ vers $\tau\text{-Q}$ est un algorithme $\mathcal{R} : (x, \kappa(x)) \rightarrow (y, \tau(y))$ tel que

1. $(x, \kappa(x))$ est positive $\Leftrightarrow (y, \tau(y))$ est positive,
2. $(y, \tau(y))$ est construite en temps **FPT**, i.e. $f(\kappa(x)) \cdot |x|^{O(1)}$,

On notera : $\kappa\text{-P} \leq_{fpt} \tau\text{-Q}$

Réductions paramétrées et W -hiérarchie (1)

Soient $\kappa\text{-P}$ et $\tau\text{-Q}$ deux problèmes paramétrés.

Une **réduction paramétrée** de $\kappa\text{-P}$ vers $\tau\text{-Q}$ est un algorithme $\mathcal{R} : (x, \kappa(x)) \rightarrow (y, \tau(y))$ tel que

1. $(x, \kappa(x))$ est positive $\Leftrightarrow (y, \tau(y))$ est positive,
2. $(y, \tau(y))$ est construite en temps **FPT**, i.e. $f(\kappa(x)) \cdot |x|^{O(1)}$,
3. et $\tau(y) \leq g(\kappa(x))$

On notera : $\kappa\text{-P} \leq_{\text{fpt}} \tau\text{-Q}$

Réductions paramétrées et W -hiérarchie (1)

Soient κ -**P** et τ -**Q** deux problèmes paramétrés.

Une **réduction paramétrée** de κ -**P** vers τ -**Q** est un algorithme $\mathcal{R} : (x, \kappa(x)) \rightarrow (y, \tau(y))$ tel que

1. $(x, \kappa(x))$ est positive $\Leftrightarrow (y, \tau(y))$ est positive,
2. $(y, \tau(y))$ est construite en temps **FPT**, i.e. $f(\kappa(x)) \cdot |x|^{O(1)}$,
3. et $\tau(y) \leq g(\kappa(x))$

On notera : κ -**P** \leq_{fpt} τ -**Q**

Observation : Si κ -**P** \leq_{fpt} τ -**Q** et τ -**Q** appartient à la classe **FPT**, alors κ -**P** appartient à la classe **FPT**.

Réductions paramétrées et W -hiérarchie (1)

Soient κ -**P** et τ -**Q** deux problèmes paramétrés.

Une **réduction paramétrée** de κ -**P** vers τ -**Q** est un algorithme $\mathcal{R} : (x, \kappa(x)) \rightarrow (y, \tau(y))$ tel que

1. $(x, \kappa(x))$ est positive $\Leftrightarrow (y, \tau(y))$ est positive,
2. $(y, \tau(y))$ est construite en temps **FPT**, i.e. $f(\kappa(x)) \cdot |x|^{O(1)}$,
3. et $\tau(y) \leq g(\kappa(x))$

On notera : κ -**P** \leq_{fpt} τ -**Q**

Observation : Si κ -**P** \leq_{fpt} τ -**Q** et τ -**Q** appartient à la classe **FPT**, alors κ -**P** appartient à la classe **FPT**.

Observation : La réduction de VERTEX COVER vers ENSEMBLE INDÉPENDANT n'est pas une transformation paramétrée polynomiale !

Réductions paramétrées et W -hiérarchie

MULTICOLORED CLIQUE

- ▶ *Données* : Un graphe $G = (V, E)$ et une coloration propre paramétré par le nombre k de couleurs.
- ▶ *Question* : Existe-t'il une clique K de taille k transversale aux couleurs? (K intersecte chacune des couleurs)

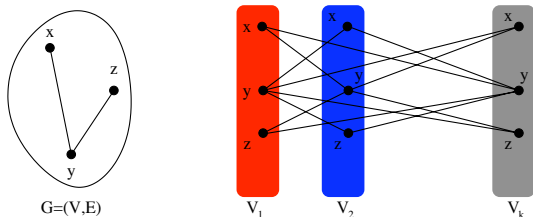
Lemme : $k\text{-CLIQUE} \leq_{fpt} k\text{-MULTICOLORED CLIQUE}$

Réductions paramétrées et W -hiérarchie

MULTICOLORED CLIQUE

- ▶ *Données* : Un graphe $G = (V, E)$ et une coloration propre paramétré par le nombre k de couleurs.
- ▶ *Question* : Existe-t'il une clique K de taille k transversale aux couleurs? (K intersecte chacune des couleurs)

Lemme : k -CLIQUE \leq_{fpt} k -MULTICOLORED CLIQUE



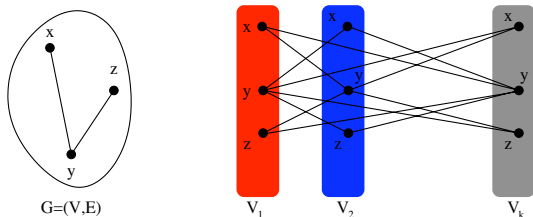
- ▶ clique de taille k dans G ssi clique multi-colorée dans H .

Réductions paramétrées et W -hiérarchie

MULTICOLORED CLIQUE

- ▶ *Données* : Un graphe $G = (V, E)$ et une coloration propre paramétré par le nombre k de couleurs.
- ▶ *Question* : Existe-t'il une clique K de taille k transversale aux couleurs? (K intersecte chacune des couleurs)

Lemme : k -CLIQUE \leq_{fpt} k -MULTICOLORED CLIQUE



- ▶ clique de taille k dans G ssi clique multi-colorée dans H .
- ▶ Transformation en temps polynomial et paramètres identiques.

Réductions paramétrées et W -hiérarchie

Exercice 3 : Montrer que k -MULTICOLORED CLIQUE \leq_{fpt} k' -CLIQUE

Réductions paramétrées et W -hiérarchie

Exercice 3 : Montrer que k -MULTICOLORED CLIQUE \leq_{fpt} k' -CLIQUE

Définition d'une hiérarchie de classes de complexité à partir de la satisfiabilité de circuits booléens.

$$\mathbf{FPT} \subseteq \mathbf{W}[1] \subseteq \mathbf{W}[2] \subseteq \dots \subseteq \mathbf{W}[t] \subseteq \dots \subseteq \mathbf{W}[P] \subseteq \mathbf{XP}$$

Hypothèses :

- ▶ k -CLIQUE et k -ENSEMBLE INDÉPENDANT sont $W[1]$ -complet
- ▶ k -ENSEMBLE DOMINANT est $W[2]$ -complet

A venir

- ▶ Kernelization - bornes inférieures et supérieures
- ▶ Techniques standards pour les algorithmes paramétrés
- ▶ Décomposition arborescente et programmation dynamique