

Complexité paramétrée (3)

Recherche de noyaux - bornes inférieures

Christophe PAUL
(CNRS - LIRMM)

Noyaux exponentiels

EDGE CLIQUE-COVER
LONGEST PATH

Algorithmes de distillation et de composition

Conjecture de OU-distillation
Non-existence de noyau polynomial
Exemples

Transformations paramétrées polynomiales

Définition et exemple
Réductions non-triviales

Compositions croisées

Définition et théorèmes
Dominating set

Autres techniques

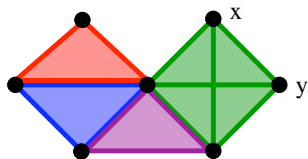
ET-composition
Bornes inférieures polynomiales

Observation

Le théorème prouvant, pour un problème paramétré (Q, κ) , l'équivalence entre l'existence d'un noyau et son appartenance à la classe de complexité FPT, ne permet que de déduire des **noyaux de tailles exponentiels**.

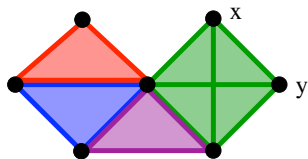
- Peut-on démontrer que certains problèmes n'admettent pas de noyau polynomial ?

L'exemple de EDGE CLIQUE-COVER



- ▶ Un graphe $G = (V, E)$ et un paramètre $k \in \mathbb{N}$
- ▶ Peut-on couvrir les arêtes E par au plus k cliques ?

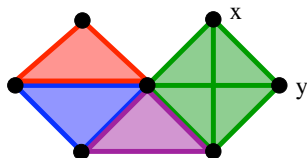
L'exemple de EDGE CLIQUE-COVER



Théorème [Gramm, Guo, Hüffner, Niedermeier]

EDGE CLIQUE-COVER admet un noyau de taille 2^k sommets.

L'exemple de EDGE CLIQUE-COVER



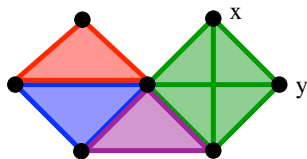
Théorème [Gramm, Guo, Hüffner, Niedermeier]

EDGE CLIQUE-COVER admet un noyau de taille 2^k sommets.

Règles de réduction

1. Supprimer les sommets isolés
2. S'il existe deux sommets x et y tels que $N[x] = N[y]$, supprimer l'un des deux sommets.

L'exemple de EDGE CLIQUE-COVER



Théorème [Gramm, Guo, Hüffner, Niedermeier]

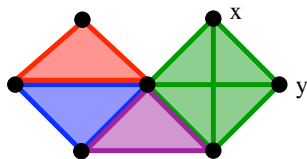
EDGE CLIQUE-COVER admet un noyau de taille 2^k sommets.

Preuve : Supposons qu'il existe k cliques $C_1 \dots C_k$ couvrant E .
A chaque sommet x on associe un vecteur C de k bits tel que

$$C[x, i] = 1 \iff x \in C_i$$

Observation : $\forall i \in [k] \ C[x, i] = C[y, i] \text{ ssi } N[x] = N[y]$

L'exemple de EDGE CLIQUE-COVER



Théorème [Gramm, Guo, Hüffner, Niedermeier]

EDGE CLIQUE-COVER admet un noyau de taille 2^k sommets.

Preuve : Supposons qu'il existe k cliques $C_1 \dots C_k$ couvrant E .
A chaque sommet x on associe un vecteur C de k bits tel que

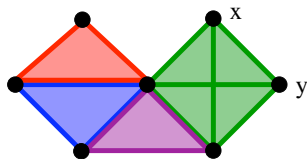
$$C[x, i] = 1 \iff x \in C_i$$

Observation : $\forall i \in [k] \ C[x, i] = C[y, i] \text{ ssi } N[x] = N[y]$

Problème ouvert

EDGE CLIQUE-COVER admet-il un noyau polynomial ?

L'exemple de EDGE CLIQUE-COVER



Théorème [Gramm, Guo, Hüffner, Niedermeier]

EDGE CLIQUE-COVER admet un noyau de taille 2^k sommets.

Preuve : Supposons qu'il existe k cliques $C_1 \dots C_k$ couvrant E .
A chaque sommet x on associe un vecteur C de k bits tel que

$$C[x, i] = 1 \iff x \in C_i$$

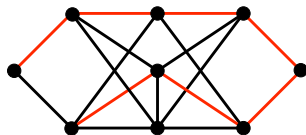
Observation : $\forall i \in [k] \ C[x, i] = C[y, i] \text{ ssi } N[x] = N[y]$

Problème ouvert

EDGE CLIQUE-COVER admet-il un noyau polynomial ? **NON**

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 (cf. chemin hamiltonien) mais peut-être résolu en temps $O(c^k \cdot n^{O(1)})$ grâce à COLOR CODING.

LONGEST PATH

Hypothèse Il existe un algorithme de kernelization \mathcal{A} pour LONGEST PATH qui retourne un noyau polynomial.

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



LONGEST PATH

Hypothèse Il existe un algorithme de kernelization \mathcal{A} pour LONGEST PATH qui retourne un noyau polynomial.

- ▶ 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 que \mathcal{A} identifie en temps polynomial quel G_i , parmi les 2^{2^n} possibles, possède un chemin de longueur k ?

Noyaux exponentiels

EDGE CLIQUE-COVER

LONGEST PATH

Algorithmes de distillation et de composition

Conjecture de OU-distillation

Non-existence de noyau polynomial

Exemples

Transformations paramétrées polynomiales

Définition et exemple

Réductions non-triviales

Compositions croisées

Définition et théorèmes

Dominating set

Autres techniques

ET-composition

Bornes inférieures polynomiales

Algorithmes de distillation

Un **algorithme de OU-distillation** \mathcal{A} pour un problème de décision Q (i.e. non paramétré) sur l'alphabet Σ est un algorithme qui :

- ▶ reçoit une séquence (x_1, \dots, x_t) , avec $x_i \in \Sigma^*$, $\forall i \in [t]$;

Algorithmes de distillation

Un **algorithme de OU-distillation** \mathcal{A} pour un problème de décision Q (i.e. non paramétré) sur l'alphabet Σ est un algorithme qui :

- ▶ reçoit une séquence (x_1, \dots, x_t) , avec $x_i \in \Sigma^*$, $\forall i \in [t]$;
- ▶ possède une complexité **polynomiale** en $\sum_{i=1}^t |x_i|$,

Algorithmes de distillation

Un **algorithme de OU-distillation** \mathcal{A} pour un problème de décision Q (i.e. non paramétré) sur l'alphabet Σ est un algorithme qui :

- ▶ reçoit une séquence (x_1, \dots, x_t) , avec $x_i \in \Sigma^*$, $\forall i \in [t]$;
- ▶ possède une complexité **polynomiale** en $\sum_{i=1}^t |x_i|$,
- ▶ retourne $y \in \Sigma^*$ tel que
 1. $y \in Q \Leftrightarrow \exists i \in [t], x_i \in Q$;
 2. $|y|$ est **polynomial** en $\max_{i \in [t]} |x_i|$.

Algorithmes de distillation

Un **algorithme de OU-distillation** \mathcal{A} pour un problème de décision Q (i.e. non paramétré) sur l'alphabet Σ est un algorithme qui :

- ▶ reçoit une séquence (x_1, \dots, x_t) , avec $x_i \in \Sigma^*$, $\forall i \in [t]$;
- ▶ possède une complexité **polynomiale** en $\sum_{i=1}^t |x_i|$,
- ▶ retourne $y \in \Sigma^*$ tel que
 1. $y \in Q \Leftrightarrow \exists i \in [t], x_i \in Q$;
 2. $|y|$ est **polynomial** en $\max_{i \in [t]} |x_i|$.

Conjecture [Bodlaender, Downey, Fellows, Hermelin]

Aucun problème NP-Complet n'est **OU-distillable**.

Algorithmes de distillation

Un **algorithme de OU-distillation** \mathcal{A} pour un problème de décision Q (i.e. non paramétré) sur l'alphabet Σ est un algorithme qui :

- ▶ reçoit une séquence (x_1, \dots, x_t) , avec $x_i \in \Sigma^*$, $\forall i \in [t]$;
- ▶ possède une complexité **polynomiale** en $\sum_{i=1}^t |x_i|$,
- ▶ retourne $y \in \Sigma^*$ tel que
 1. $y \in Q \Leftrightarrow \exists i \in [t], x_i \in Q$;
 2. $|y|$ est **polynomial** en $\max_{i \in [t]} |x_i|$.

Conjecture [Bodlaender, Downey, Fellows, Hermelin]

Aucun problème NP-Complet n'est **OU-distillable**.

Théorème [Fortnow et Santhanam]

S'il existe un problème NP-complet OU-distillable, alors $PH = \Sigma_p^3$

Algorithme de OU-composition

Un **algorithme de OU-composition** pour un problème paramétré $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$ est un algorithme \mathcal{A} qui

- ▶ reçoit une séquence $((x_1, k), \dots, (x_t, k))$, avec $(x_i, k) \in \Sigma^* \times \mathbb{N}$
 $\forall i \in [t]$;

Algorithme de OU-composition

Un algorithme de OU-composition pour un problème paramétré $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$ est un algorithme \mathcal{A} qui

- ▶ reçoit une séquence $((x_1, k), \dots, (x_t, k))$, avec $(x_i, k) \in \Sigma^* \times \mathbb{N} \forall i \in [t]$;
- ▶ a une complexité polynomiale en $\sum_{i=1}^t |x_i| + k$,

Algorithme de OU-composition

Un algorithme de OU-composition pour un problème paramétré $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$ est un algorithme \mathcal{A} qui

- ▶ reçoit une séquence $((x_1, k), \dots, (x_t, k))$, avec $(x_i, k) \in \Sigma^* \times \mathbb{N} \forall i \in [t]$;
- ▶ a une complexité polynomiale en $\sum_{i=1}^t |x_i| + k$,
- ▶ retourne $(y, k') \in \Sigma^* \times \mathbb{N}$ tel que
 1. $(y, k') \in (Q, \kappa) \Leftrightarrow \exists i \in [t], (x_i, k) \in (Q, \kappa)$;
 2. k' est polynomial en k .

Algorithme de OU-composition

Un algorithme de OU-composition pour un problème paramétré $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$ est un algorithme \mathcal{A} qui

- ▶ reçoit une séquence $((x_1, k), \dots, (x_t, k))$, avec $(x_i, k) \in \Sigma^* \times \mathbb{N} \forall i \in [t]$;
- ▶ a une complexité polynomiale en $\sum_{i=1}^t |x_i| + k$,
- ▶ retourne $(y, k') \in \Sigma^* \times \mathbb{N}$ tel que
 1. $(y, k') \in (Q, \kappa) \Leftrightarrow \exists i \in [t], (x_i, k) \in (Q, \kappa)$;
 2. k' est polynomial en k .

Observation : L'existence d'un noyau polynomial pour LONGEST PATH impliquerait l'existence d'un algorithme de OU-composition.

Théorème [Bodlaender, Downey, Fellows, Hermelin]

Soit $(Q, \kappa) \in \Sigma^* \times \mathbb{N}$ un problème paramétré OU-composable tel que le problème $\tilde{Q} \in \Sigma^*$ (non-paramétré) est NP-Complet.

Si (Q, κ) admet un noyau polynomial, alors \tilde{Q} est OU-distillable.

Théorème [Bodlaender, Downey, Fellows, Hermelin]

Soit $(Q, \kappa) \in \Sigma^* \times \mathbb{N}$ un problème paramétré OU-composable tel que le problème $\tilde{Q} \in \Sigma^*$ (non-paramétré) est NP-Complet.

Si (Q, κ) admet un noyau polynomial, alors \tilde{Q} est OU-distillable.

► $(x, \kappa(x))$ instance de $(Q, \kappa) \longrightarrow x\#1^k$ instance de \tilde{Q}

Théorème [Bodlaender, Downey, Fellows, Hermelin]

Soit $(Q, \kappa) \in \Sigma^* \times \mathbb{N}$ un problème paramétré OU-composable tel que le problème $\tilde{Q} \in \Sigma^*$ (non-paramétré) est NP-Complet.

Si (Q, κ) admet un noyau polynomial, alors \tilde{Q} est OU-distillable.

▶ $(x, \kappa(x))$ instance de $(Q, \kappa) \longrightarrow x\#1^k$ instance de \tilde{Q}

▶ Puisque \tilde{Q} est un problème NP-Complet, il existe deux transformations polynomiales

$$\Phi : \tilde{Q} \longrightarrow \text{SAT} \quad \Psi : \text{SAT} \longrightarrow \tilde{Q}$$

Théorème [Bodlaender, Downey, Fellows, Hermelin]

Soit $(Q, \kappa) \in \Sigma^* \times \mathbb{N}$ un problème paramétré OU-composable tel que le problème $\tilde{Q} \in \Sigma^*$ (non-paramétré) est NP-Complet.

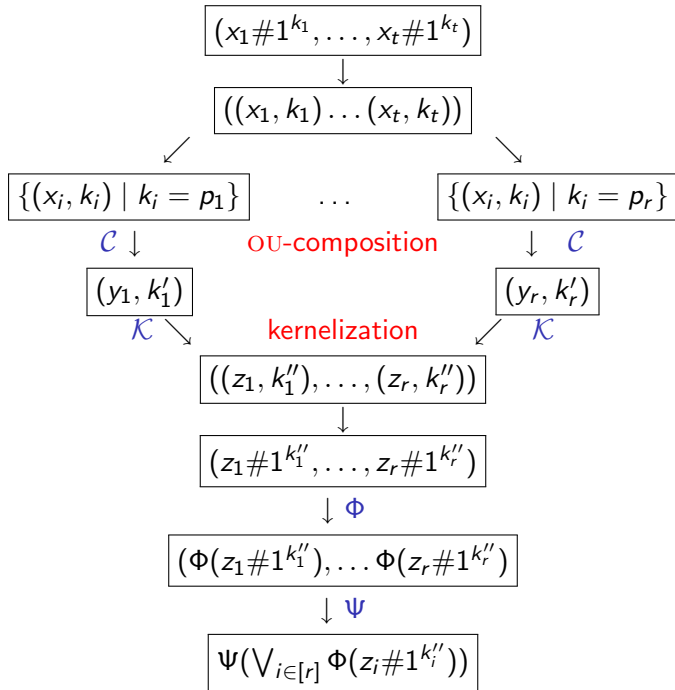
Si (Q, κ) admet un noyau polynomial, alors \tilde{Q} est OU-distillable.

▶ $(x, \kappa(x))$ instance de $(Q, \kappa) \longrightarrow x\#1^k$ instance de \tilde{Q}

▶ Puisque \tilde{Q} est un problème NP-Complet, il existe deux transformations polynomiales

$$\Phi : \tilde{Q} \longrightarrow \text{SAT} \quad \Psi : \text{SAT} \longrightarrow \tilde{Q}$$

▶ on va construire un algorithme \mathcal{A} de OU-distillation pour \tilde{Q} à partir de Φ , Ψ , de l'algorithme de OU-composition \mathcal{C} et de l'algorithme \mathcal{K} calculant le noyau de (Q, κ) .



L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

$$\blacktriangleright \Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''})) \in \tilde{Q} \iff \exists i \in [r], (x_1 \# 1^{k_1}) \in \tilde{Q}$$

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

- ▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_1 \# 1^{k_1}) \in \tilde{Q}$
- ▶ La complexité de l'algorithme est polynomiale en $\sum_{i \in [t]} |x_i|$

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

- ▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_i \# 1^{k_i}) \in \tilde{Q}$
- ▶ La complexité de l'algorithme est polynomiale en $\sum_{i \in [t]} |x_i|$
- ▶ Il reste à prouver que la taille de l'instance de \tilde{Q} retournée est polynomiale en $n = \max_{i \in [t]} |x_i \# 1^{k_i}|$

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

- ▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_i \# 1^{k_i}) \in \tilde{Q}$
- ▶ La complexité de l'algorithme est polynomiale en $\sum_{i \in [t]} |x_i|$
- ▶ Il reste à prouver que la taille de l'instance de \tilde{Q} retournée est polynomiale en $n = \max_{i \in [t]} |x_i \# 1^{k_i}|$
 - ▶ $r \leq k = \max_{i \in [r]} k_r \leq n$

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

- ▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_i \# 1^{k_i}) \in \tilde{Q}$
- ▶ La complexité de l'algorithme est polynomiale en $\sum_{i \in [t]} |x_i|$
- ▶ Il reste à prouver que la taille de l'instance de \tilde{Q} retournée est polynomiale en $n = \max_{i \in [t]} |x_i \# 1^{k_i}|$
 - ▶ $r \leq k = \max_{i \in [r]} k_r \leq n$
 - ▶ $\forall i \in [r], k'_i$ est borné par un polynôme en $k_i \leq k \leq n$
(\mathcal{C} est un algorithme de OU-composition)

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

- ▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_i \# 1^{k_i}) \in \tilde{Q}$
- ▶ La complexité de l'algorithme est polynomiale en $\sum_{i \in [t]} |x_i|$
- ▶ Il reste à prouver que la taille de l'instance de \tilde{Q} retournée est polynomiale en $n = \max_{i \in [t]} |x_i \# 1^{k_i}|$
 - ▶ $r \leq k = \max_{i \in [r]} k_r \leq n$
 - ▶ $\forall i \in [r], k_i'$ est borné par un polynôme en $k_i \leq k \leq n$
(\mathcal{C} est un algorithme de OU-composition)
 - ▶ Donc pour tout $i \in [r]$, la taille de $z_i \# 1^{k_i''}$ est bornée par un polynôme en n (\mathcal{K} est une kernalization)

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

- ▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_i \# 1^{k_i}) \in \tilde{Q}$
- ▶ La complexité de l'algorithme est polynomiale en $\sum_{i \in [t]} |x_i|$
- ▶ Il reste à prouver que la taille de l'instance de \tilde{Q} retournée est polynomiale en $n = \max_{i \in [t]} |x_i \# 1^{k_i}|$
 - ▶ $r \leq k = \max_{i \in [r]} k_r \leq n$
 - ▶ $\forall i \in [r], k_i'$ est borné par un polynôme en $k_i \leq k \leq n$
(\mathcal{C} est un algorithme de OU-composition)
 - ▶ Donc pour tout $i \in [r]$, la taille de $z_i \# 1^{k_i''}$ est bornée par un polynôme en n (\mathcal{K} est une kernalization)
 - ▶ Donc la taille de $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''})))$ est bornée par un polynôme en n (Φ et Ψ sont des transformations polynomiales)

Conséquences

Corollaire

Sauf si $PH = \Sigma_p^3$, LONGEST PATH n'admet pas de noyau polynomial.

Conséquences

Corollaire

Sauf si $PH = \Sigma_p^3$, LONGEST PATH n'admet pas de noyau polynomial.

Sous-arborescence avec k feuilles

Etant donné un graphe orienté $D = (V, \vec{E})$, existe-t-il une arborescence \vec{T} dans D avec k feuilles ? \rightarrow algo en $O(4^k n^{O(1)})$

Conséquences

Corollaire

Sauf si $PH = \Sigma_p^3$, LONGEST PATH n'admet pas de noyau polynomial.

Sous-arborescence avec k feuilles

Etant donné un graphe orienté $D = (V, \vec{E})$, existe-t-il une arborescence \vec{T} dans D avec k feuilles ? \rightarrow algo en $O(4^k n^{O(1)})$

Lemme

Sauf si $PH = \Sigma_p^3$, le problème k -SOUS-ARBORESCENCE n'admet pas de noyau polynomial.

Conséquences

Corollaire

Sauf si $PH = \Sigma_p^3$, LONGEST PATH n'admet pas de noyau polynomial.

Sous-arborescence avec k feuilles

Etant donné un graphe orienté $D = (V, \vec{E})$, existe-t-il une arborescence \vec{T} dans D avec k feuilles ? \rightarrow algo en $O(4^k n^{O(1)})$

Lemme

Sauf si $PH = \Sigma_p^3$, le problème k -SOUS-ARBORESCENCE n'admet pas de noyau polynomial.

Remarque La version enracinée (on fixe une racine r) admet un noyau cubique !

Noyaux exponentiels

EDGE CLIQUE-COVER

LONGEST PATH

Algorithmes de distillation et de composition

Conjecture de OU-distillation

Non-existence de noyau polynomial

Exemples

Transformations paramétrées polynomiales

Définition et exemple

Réductions non-triviales

Compositions croisées

Définition et théorèmes

Dominating set

Autres techniques

ET-composition

Bornes inférieures polynomiales

Transformations polynomiales et paramétrées

Soient (P, π) et (Q, κ) deux problèmes paramétrés.

Une **transformation polynomiale et paramétrée (TPP)** de (P, π) vers (Q, κ) est un algorithme polynomial \mathcal{A} qui :

- ▶ à toute instance (x, k) de (P, π) associe une instance (x', k') de (Q, κ) ;
- ▶ $(x, k) \in (P, \pi) \iff (x', k') \in (Q, \kappa)$ et $k' \leq \text{poly}(k)$

On note $(P, \pi) \leq_{TPP} (Q, \kappa)$

Transformations polynomiales et paramétrées

Soient (P, π) et (Q, κ) deux problèmes paramétrés.

Une **transformation polynomiale et paramétrée (TPP)** de (P, π) vers (Q, κ) est un algorithme polynomial \mathcal{A} qui :

- ▶ à toute instance (x, k) de (P, π) associe une instance (x', k') de (Q, κ) ;
- ▶ $(x, k) \in (P, \pi) \iff (x', k') \in (Q, \kappa)$ et $k' \leq \text{poly}(k)$

On note $(P, \pi) \leq_{TPP} (Q, \kappa)$

Théorème [Bodlaender, Thomassé, Yeo]

Soient (P, π) et (Q, κ) deux problèmes paramétrés tels que P est NP-Complet et Q appartient à NP.

Si $(P, \pi) \leq_{TPP} (Q, \kappa)$ et si (P, π) n'admet pas de noyau polynomial, alors (Q, κ) n'admet pas de noyau polynomial.

Exercice : Faire la preuve du théorème précédent.

(Idée : *en supposant que (Q, κ) admet un noyau polynomial, alors on construit un algorithme polynomial qui réduit (P, π) à un noyau polynomial.*)

Exercice : Faire la preuve du théorème précédent.

Utilisation

- ▶ construction de noyaux polynomiaux
- ▶ preuve de non-existence de noyau polynomial:

Exercice : Faire la preuve du théorème précédent.

Utilisation

- ▶ construction de noyaux polynomiaux
- ▶ preuve de non-existence de noyau polynomial:

PATH PACKING

→ Tester si un graphe contient k chemins sommets disjoints de taille k

Exercice : Faire la preuve du théorème précédent.

Utilisation

- ▶ construction de noyaux polynomiaux
- ▶ preuve de non-existence de noyau polynomial:

PATH PACKING

→ Tester si un graphe contient k chemins sommets disjoints de taille k

Remarque : PATH-PACKING n'est pas OU-composable !

Exercice : Faire la preuve du théorème précédent.

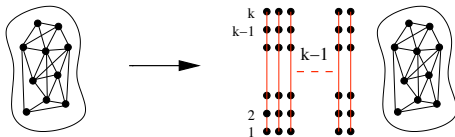
Utilisation

- ▶ construction de noyaux polynomiaux
- ▶ preuve de non-existence de noyau polynomial:

PATH PACKING

→ Tester si un graphe contient k chemins sommets disjoints de taille k

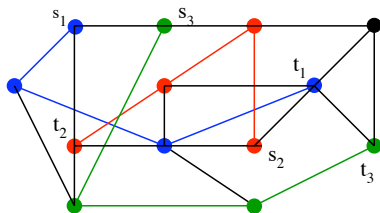
Remarque : PATH-PACKING n'est pas OU-composable !



LONGEST PATH \leq_{TPP} PATH PACKING

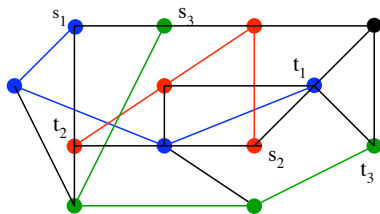
DISJOINT PATHS

- ▶ Un graphe G et k paires de sommets $(s_1, t_1), \dots, (s_k, t_k)$
- ▶ G contient-il k chemins P_1, \dots, P_k , **sommets disjoints** tels que P_i est un chemin entre s_i et t_i ($i \in [k]$) ?



DISJOINT PATHS

- ▶ Un graphe G et k paires de sommets $(s_1, t_1), \dots, (s_k, t_k)$
- ▶ G contient-il k chemins P_1, \dots, P_k , **sommets disjoints** tels que P_i est un chemin entre s_i et t_i ($i \in [k]$) ?



Remarque

- ▶ DISJOINT PATHS est FPT et NPC [Roberston et Seymour]
- ▶ Mais DISJOINT PATHS n'est pas OU-composable

DISJOINT PATHS

Méthode

- ▶ Introduction d'un problème intermédiaire (P, π)
- ▶ Montrer que (P, π) est FPT et OU-composable, que P (non-paramétré) est NP-Complet
- ▶ Montrer que $(P, \pi) \leq_{TPP} \text{DISJOINT PATHS}$

DISJOINT PATHS

Méthode

- ▶ Introduction d'un problème intermédiaire (P, π)
- ▶ Montrer que (P, π) est FPT et OU-composable, que P (non-paramétré) est NP-Complet
- ▶ Montrer que $(P, \pi) \leq_{TPP} \text{DISJOINT PATHS}$

DISJOINT FACTORS

Un mot W sur $\Sigma = \{1, \dots, k\}$ possède la **propriété des facteurs disjoints** si W contient k facteurs disjoints F_1, \dots, F_k tels que $\forall i \in [k]$, F_i commence et termine par la lettre i et $|F_i| \geq 2$.

DISJOINT PATHS

Méthode

- ▶ Introduction d'un problème intermédiaire (P, π)
- ▶ Montrer que (P, π) est FPT et OU-composable, que P (non-paramétré) est NP-Complet
- ▶ Montrer que $(P, \pi) \leq_{TPP} \text{DISJOINT PATHS}$

DISJOINT FACTORS

Un mot W sur $\Sigma = \{1, \dots, k\}$ possède la **propriété des facteurs disjoints** si W contient k facteurs disjoints F_1, \dots, F_k tels que $\forall i \in [k]$, F_i commence et termine par la lettre i et $|F_i| \geq 2$.

1 2 4 3 2 4 3 1 3 2 4 2 3 4 2 3 1 4 1

DISJOINT PATHS

Exercice

1. Montrer que DISJOINT FACTORS est FPT
(idée : par programmation dynamique en temps $O(nk \cdot 2^k)$)
2. Montrer que DISJOINT FACTORS est NP-Complet
(idée : réduction depuis 3-SAT)

DISJOINT PATHS

Exercice

1. Montrer que DISJOINT FACTORS est FPT
(idée : par programmation dynamique en temps $O(nk \cdot 2^k)$)
2. Montrer que DISJOINT FACTORS est NP-Complet
(idée : réduction depuis 3-SAT)

Lemme [BTY] : DISJOINT FACTORS est OU-composable

DISJOINT PATHS

Exercice

1. Montrer que DISJOINT FACTORS est FPT
(idée : par programmation dynamique en temps $O(nk \cdot 2^k)$)
2. Montrer que DISJOINT FACTORS est NP-Complet
(idée : réduction depuis 3-SAT)

Lemme [BTY] : DISJOINT FACTORS est OU-composable

$$\boxed{((W_1, k), (W_2, k))}$$



$$\boxed{((k+1).W_1.(k+1).W_2.(k+1), (k+1))}$$

DISJOINT PATHS

Exercice

1. Montrer que DISJOINT FACTORS est FPT
(idée : par programmation dynamique en temps $O(nk \cdot 2^k)$)
2. Montrer que DISJOINT FACTORS est NP-Complet
(idée : réduction depuis 3-SAT)

Lemme [BTY] : DISJOINT FACTORS est OU-composable

$$\left((W_1, k), (W_2, k), (W_3, k), (W_4, k) \right)$$

$$(k+1)W_1(k+1)W_2(k+1)$$

$$(k+1)W_3.(k+1)W_4(k+1)$$

$$(k+2)(k+1)W_1(k+1)W_2(k+1)(k+2)(k+1)W_3(k+1)W_4(k+1)(k+2)$$

DISJOINT PATHS

Exercice

1. Montrer que DISJOINT FACTORS est FPT
(idée : par programmation dynamique en temps $O(nk \cdot 2^k)$)
2. Montrer que DISJOINT FACTORS est NP-Complet
(idée : réduction depuis 3-SAT)

Lemme [BTY] : DISJOINT FACTORS est OU-composable

$$((W_1, k) \dots (W_t, k)) \longrightarrow (W', k + \lceil \log_2 t \rceil)$$

Remarques:

DISJOINT PATHS

Exercice

1. Montrer que DISJOINT FACTORS est FPT
(idée : par programmation dynamique en temps $O(nk \cdot 2^k)$)
2. Montrer que DISJOINT FACTORS est NP-Complet
(idée : réduction depuis 3-SAT)

Lemme [BTY] : DISJOINT FACTORS est OU-composable

$$((W_1, k) \dots (W_t, k)) \longrightarrow (W', k + \lceil \log_2 t \rceil)$$

Remarques:

- ▶ $t \leq 2^k$, sinon le problème peut se résoudre en temps polynomial car $2^k \leq \sum_1^t |W_i|$.

DISJOINT PATHS

Exercice

1. Montrer que DISJOINT FACTORS est FPT
(idée : par programmation dynamique en temps $O(nk \cdot 2^k)$)
2. Montrer que DISJOINT FACTORS est NP-Complet
(idée : réduction depuis 3-SAT)

Lemme [BTY] : DISJOINT FACTORS est OU-composable

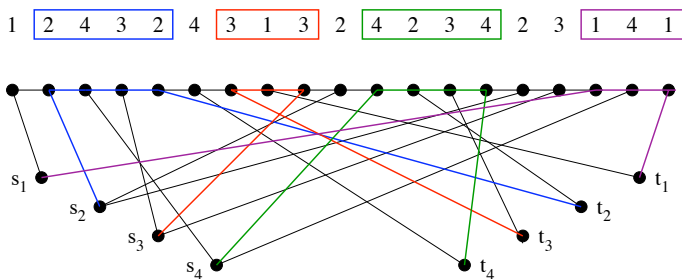
$$((W_1, k) \dots (W_t, k)) \longrightarrow (W', k + \lceil \log_2 t \rceil)$$

Remarques:

- ▶ $t \leq 2^k$, sinon le problème peut se résoudre en temps polynomial car $2^k \leq \sum_1^t |W_i|$.
- ▶ $k + \lceil \log_2 t \rceil \leq 2k$

DISJOINT PATHS

Lemme [BTY] : DISJOINT FACTORS \leq_{TPP} DISJOINT PATHS



Théorème [Bodlaender, Thomassé, Yeo]

DISJOINT PATHS n'admet pas de noyau polynomial à moins que

$$PH = \Sigma_p^3$$

Noyaux exponentiels

EDGE CLIQUE-COVER

LONGEST PATH

Algorithmes de distillation et de composition

Conjecture de OU-distillation

Non-existence de noyau polynomial

Exemples

Transformations paramétrées polynomiales

Définition et exemple

Réductions non-triviales

Compositions croisées

Définition et théorèmes

Dominating set

Autres techniques

ET-composition

Bornes inférieures polynomiales

COMPOSITIONS CROISÉES

Idée de base / objectifs

- ▶ unifier les deux techniques existantes (OU-composition et TPP)
- ▶ permettre de composer un problème vers un autre problème
- ▶ être capable de composer des problèmes NP-complets vers des problèmes FPT

COMPOSITIONS CROISÉES

Idée de base / objectifs

- ▶ unifier les deux techniques existantes (OU-composition et TPP)
- ▶ permettre de composer un problème vers un autre problème
- ▶ être capable de composer des problèmes NP-complets vers des problèmes FPT

Une relation d'équivalence \mathcal{R} sur Σ^* est une **relation d'équivalence polynomiale** si

1. \exists un algorithme qui décide si $x\mathcal{R}y$ en temps $(|x| + |y|)^{O(1)}$
2. $\forall S \subseteq \Sigma^*$, \mathcal{R} partitionne S en au plus $(\max_{x \in S} |X|)^{O(1)}$ classes

COMPOSITIONS CROISÉES

Idée de base / objectifs

- ▶ unifier les deux techniques existantes (OU-composition et TPP)
- ▶ permettre de composer un problème vers un autre problème
- ▶ être capable de composer des problèmes NP-complets vers des problèmes FPT

Une relation d'équivalence \mathcal{R} sur Σ^* est une **relation d'équivalence polynomiale** si

1. \exists un algorithme qui décide si $x\mathcal{R}y$ en temps $(|x| + |y|)^{O(1)}$
 2. $\forall S \subseteq \Sigma^*$, \mathcal{R} partitionne S en au plus $(\max_{x \in S} |X|)^{O(1)}$ classes
- ▶ **Exemple** : nombre de sommets et valeur du paramètres égaux

COMPOSITIONS CROISÉES

Soient $L \subseteq \Sigma^*$ et $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$. Il existe une **composition croisée** de L vers Q si il existe une relation d'équivalence polynomiale \mathcal{R} sur Σ^* et un algorithme \mathcal{A} qui

- ▶ reçoit une séquence (x_1, \dots, x_t) , tel que $\forall 1 \leq i < j \leq t, x_i \mathcal{R} x_j$
- ▶ a une complexité **polynomiale** en $\sum_{i=1}^t |x_i|$
- ▶ retourne $(x, k) \in \Sigma^* \times \mathbb{N}$ tel que
 1. $(x, k) \in (Q, \kappa) \Leftrightarrow \exists i \in [t], x_i \in L$
 2. k est polynomial en $\max_{i=1}^t |x_i| + \log t$.

COMPOSITIONS CROISÉES

Soient $L \subseteq \Sigma^*$ et $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$. Il existe une **composition croisée** de L vers Q si il existe une relation d'équivalence polynomiale \mathcal{R} sur Σ^* et un algorithme \mathcal{A} qui

- ▶ reçoit une séquence (x_1, \dots, x_t) , tel que $\forall 1 \leq i < j \leq t, x_i \mathcal{R} x_j$
- ▶ a une complexité **polynomiale** en $\sum_{i=1}^t |x_i|$
- ▶ retourne $(x, k) \in \Sigma^* \times \mathbb{N}$ tel que
 1. $(x, k) \in (Q, \kappa) \Leftrightarrow \exists i \in [t], x_i \in L$
 2. k est polynomial en $\max_{i=1}^t |x_i| + \log t$.

Théorème [Bodlaender, Jansen, Kratsch]

Soit \mathcal{A} une **composition croisée** d'un problème NP-complet $L \subseteq \Sigma^*$ vers un problème paramétré (Q, κ) tel que $\tilde{Q} \in \text{NP-complet}$. Si (Q, κ) admet un **noyau polynomial**, alors L admet une OU-distillation.

COMPOSITIONS CROISÉES

Soient $L \subseteq \Sigma^*$ et $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$. Il existe une **composition croisée** de L vers Q si il existe une relation d'équivalence polynomiale \mathcal{R} sur Σ^* et un algorithme \mathcal{A} qui

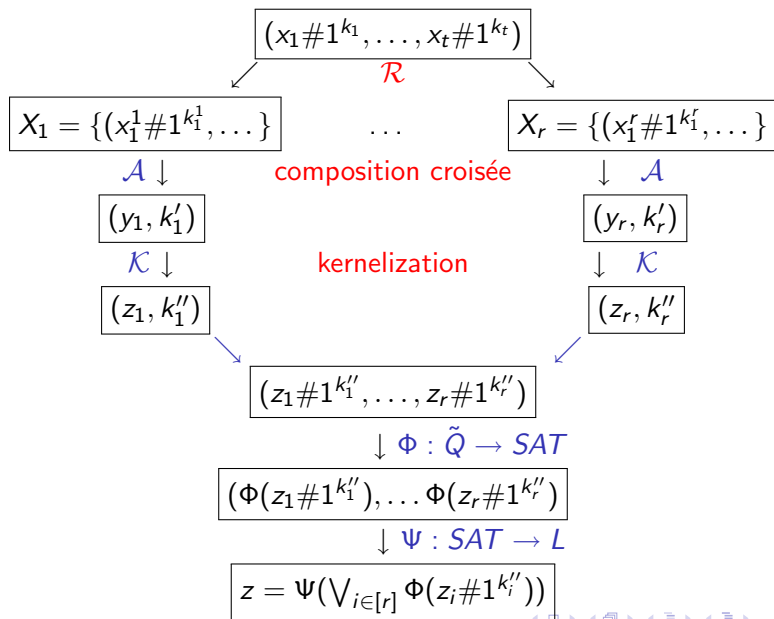
- ▶ reçoit une séquence (x_1, \dots, x_t) , tel que $\forall 1 \leq i < j \leq t, x_i \mathcal{R} x_j$
- ▶ a une complexité **polynomiale** en $\sum_{i=1}^t |x_i|$
- ▶ retourne $(x, k) \in \Sigma^* \times \mathbb{N}$ tel que
 1. $(x, k) \in (Q, \kappa) \Leftrightarrow \exists i \in [t], x_i \in L$
 2. k est polynomial en $\max_{i=1}^t |x_i| + \log t$.

Théorème [Bodlaender, Jansen, Kratsch]

Soit \mathcal{A} une **composition croisée** d'un problème NP-complet $L \subseteq \Sigma^*$ vers un problème paramétré (Q, κ) tel que $\tilde{Q} \in \text{NP-complet}$. Si (Q, κ) admet un **noyau polynomial**, alors L admet une OU-distillation.

- ▶ preuve similaire à celle de la OU-composition

COMPOSITIONS CROISÉES



L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

- ▶ On pose $m = \max_{i \in [t]} |x_i|$ et $t \leq (|\Sigma| + 1)^m$ (sinon certaines instances sont des copies que l'on supprime)

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

- ▶ On pose $m = \max_{i \in [t]} |x_i|$ et $t \leq (|\Sigma| + 1)^m$ (sinon certaines instances sont des copies que l'on supprime)
- ▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_1 \# 1^{k_1}) \in \tilde{Q}$

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

- ▶ On pose $m = \max_{i \in [t]} |x_i|$ et $t \leq (|\Sigma| + 1)^m$ (sinon certaines instances sont des copies que l'on supprime)
- ▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_1 \# 1^{k_1}) \in \tilde{Q}$
- ▶ La complexité de l'algorithme est polynomiale en
$$\sum_{i \in [t]} |x_i|$$

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

- ▶ On pose $m = \max_{i \in [t]} |x_i|$ et $t \leq (|\Sigma| + 1)^m$ (sinon certaines instances sont des copies que l'on supprime)
- ▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_1 \# 1^{k_1}) \in \tilde{Q}$
- ▶ La complexité de l'algorithme est polynomiale en
$$\sum_{i \in [t]} |x_i|$$
- ▶ Il reste à prouver que la taille de l'instance z de \tilde{Q} est polynomiale en $n = \max_{i \in [t]} |x_i \# 1^{k_i}|$

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

▶ On pose $m = \max_{i \in [t]} |x_i|$ et $t \leq (|\Sigma| + 1)^m$ (sinon certaines instances sont des copies que l'on supprime)

▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_i \# 1^{k_i}) \in \tilde{Q}$

▶ La complexité de l'algorithme est polynomiale en

$$\sum_{i \in [t]} |x_i|$$

▶ Il reste à prouver que la taille de l'instance z de \tilde{Q} est polynomiale en $n = \max_{i \in [t]} |x_i \# 1^{k_i}|$

▶ r est polynomiale en m

\mathcal{R} est une relation d'équivalence polynomiale

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

▶ On pose $m = \max_{i \in [t]} |x_i|$ et $t \leq (|\Sigma| + 1)^m$ (sinon certaines instances sont des copies que l'on supprime)

▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_i \# 1^{k_i}) \in \tilde{Q}$

▶ La complexité de l'algorithme est polynomiale en

$$\sum_{i \in [t]} |x_i|$$

▶ Il reste à prouver que la taille de l'instance z de \tilde{Q} est polynomiale en $n = \max_{i \in [t]} |x_i \# 1^{k_i}|$

▶ r est polynomiale en m

\mathcal{R} est une relation d'équivalence polynomiale

▶ $\forall i \in [r], k_i'$ est polynomiale en $m + \log t$

\mathcal{A} est une composition croisée

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

▶ On pose $m = \max_{i \in [t]} |x_i|$ et $t \leq (|\Sigma| + 1)^m$ (sinon certaines instances sont des copies que l'on supprime)

▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_1 \# 1^{k_1}) \in \tilde{Q}$

▶ La complexité de l'algorithme est polynomiale en

$$\sum_{i \in [t]} |x_i|$$

▶ Il reste à prouver que la taille de l'instance z de \tilde{Q} est polynomiale en $n = \max_{i \in [t]} |x_i \# 1^{k_i}|$

▶ r est polynomiale en m

\mathcal{R} est une relation d'équivalence polynomiale

▶ $\forall i \in [r], k_i'$ est polynomiale en $m + \log t$

\mathcal{A} est une composition croisée

▶ $\forall i \in [r], |z_i|$ et k_i'' est polynomiale en $m + \log t$

\mathcal{K} est une kernelization

L'algorithme décrit est un algorithme de OU-distillation pour \tilde{Q} .

▶ On pose $m = \max_{i \in [t]} |x_i|$ et $t \leq (|\Sigma| + 1)^m$ (sinon certaines instances sont des copies que l'on supprime)

▶ $\Psi(\bigvee_{i \in [r]} (\Phi(z_i \# 1^{k_i''}))) \in \tilde{Q} \iff \exists i \in [r], (x_1 \# 1^{k_1}) \in \tilde{Q}$

▶ La complexité de l'algorithme est polynomiale en

$$\sum_{i \in [t]} |x_i|$$

▶ Il reste à prouver que la taille de l'instance z de \tilde{Q} est polynomiale en $n = \max_{i \in [t]} |x_i \# 1^{k_i}|$

▶ r est polynomiale en m

\mathcal{R} est une relation d'équivalence polynomiale

▶ $\forall i \in [r], k_i'$ est polynomiale en $m + \log t$

\mathcal{A} est une composition croisée

▶ $\forall i \in [r], |z_i|$ et k_i'' est polynomiale en $m + \log t$

\mathcal{K} est une kernelization

▶ $|z|$ est polynomiale en $r \times (m + \log t)$ donc polynomiale en m

Corollaire :

Soit \mathcal{A} une composition croisée d'un problème NP-complet $L \subseteq \Sigma^*$ vers un problème paramétré (Q, κ) tel que $\tilde{Q} \in \text{NP-complet}$.
Sauf si $PH = \Sigma_p^3$, (Q, κ) n'admet pas de noyau polynomial.

Corollaire :

Soit \mathcal{A} une composition croisée d'un problème NP-complet $L \subseteq \Sigma^*$ vers un problème paramétré (Q, κ) tel que $\tilde{Q} \in \text{NP-complet}$.
Sauf si $PH = \Sigma_p^3$, (Q, κ) n'admet pas de noyau polynomial.

Observations :

1. Si le problème paramétré (P, κ) est OU-composable, alors il existe une composition croisée de \tilde{P} vers (P, κ) .
 - ▶ la relation d'équivalence polynomiale regroupe les instances "valides" $x\#1^k$ selon la valeur du paramètre k

Corollaire :

Soit \mathcal{A} une composition croisée d'un problème NP-complet $L \subseteq \Sigma^*$ vers un problème paramétré (Q, κ) tel que $\tilde{Q} \in \text{NP-complet}$.
Sauf si $PH = \Sigma_p^3$, (Q, κ) n'admet pas de noyau polynomial.

Observations :

1. Si le problème paramétré (P, κ) est OU-composable, alors il existe une composition croisée de \tilde{P} vers (P, κ) .
 - ▶ la relation d'équivalence polynomiale regroupe les instances "valides" $x\#1^k$ selon la valeur du paramètre k
2. Soient (P, κ_P) et (Q, κ_Q) deux problèmes paramétrés tels que (P, κ_P) est OU-composable et $(P, \kappa_P) \leq_{TPP} (Q, \kappa_Q)$, alors il existe une composition croisée de \tilde{P} vers (Q, κ_Q) .

DOMINATING SET

- ▶ Un graphe $G = (V, E)$ contient-il un **ensemble dominant** $S \subseteq V$ de taille au plus k :

$$\forall x \in V, x \notin S \Rightarrow \exists y \in S, (x, y) \in E$$

DOMINATING SET

- ▶ Un graphe $G = (V, E)$ contient-il un **ensemble dominant** $S \subseteq V$ de taille au plus k :

$$\forall x \in V, x \notin S \Rightarrow \exists y \in S, (x, y) \in E$$

Résultats connus

- ▶ $W[2]$ -complet si paramétré par la **taille de la solution**
 \Rightarrow pas de noyau exponentiel !

DOMINATING SET

- ▶ Un graphe $G = (V, E)$ contient-il un **ensemble dominant** $S \subseteq V$ de taille au plus k :

$$\forall x \in V, x \notin S \Rightarrow \exists y \in S, (x, y) \in E$$

Résultats connus

- ▶ $W[2]$ -complet si paramétré par la **taille de la solution**
 \Rightarrow pas de noyau exponentiel !
- ▶ FPT et OU-composable si paramétré par $treewidth(G)$
 \Rightarrow pas de noyau polynomial !

DOMINATING SET

- ▶ Un graphe $G = (V, E)$ contient-il un **ensemble dominant** $S \subseteq V$ de taille au plus k :

$$\forall x \in V, x \notin S \Rightarrow \exists y \in S, (x, y) \in E$$

Résultats connus

- ▶ $W[2]$ -complet si paramétré par la **taille de la solution**
 \Rightarrow pas de noyau exponentiel !
- ▶ FPT et OU-composable si paramétré par $treewidth(G)$
 \Rightarrow pas de noyau polynomial !
- ▶ FPT si paramétré par la taille d'un **vertex cover**

DOMINATING SET

- ▶ Un graphe $G = (V, E)$ contient-il un **ensemble dominant** $S \subseteq V$ de taille au plus k :

$$\forall x \in V, x \notin S \Rightarrow \exists y \in S, (x, y) \in E$$

Résultats connus

- ▶ $W[2]$ -complet si paramétré par la **taille de la solution**
 \Rightarrow pas de noyau exponentiel !
- ▶ FPT et OU-composable si paramétré par $treewidth(G)$
 \Rightarrow pas de noyau polynomial !
- ▶ FPT si paramétré par la taille d'un **vertex cover**

MAIS

$$vertex\ cover(G) \geq treewidth(G)$$

$\exists?$ un noyau polynomial pour la paramétrisation par vertex cover

DOMINATING SET

Théorème [Wratigan]

Il existe une **composition croisée** de VERTEX COVER vers DOMINATING SET paramétré par vertex cover.

DOMINATING SET

Théorème [Wratigan]

Il existe une **composition croisée** de VERTEX COVER vers DOMINATING SET paramétré par vertex cover.

Soit $(G_1 \# 1^{k_1}, \dots, G_t \# 1^{k_t})$ une série d'instances de VERTEX COVER

- ▶ les deux problèmes sont NP-Complets
- ▶ $G_i \# 1^{k_i} \mathcal{R} G_j \# 1^{k_j}$ si $|V(G_i)| = |V(G_j)| = n$ et $k_i = k_j = k$

DOMINATING SET

Théorème [Wratigan]

Il existe une **composition croisée** de VERTEX COVER vers DOMINATING SET paramétré par vertex cover.

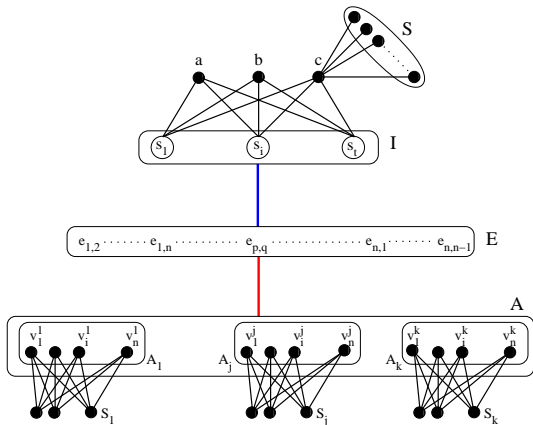
Soit $(G_1 \# 1^{k_1}, \dots, G_t \# 1^{k_t})$ une série d'instances de VERTEX COVER

- ▶ les deux problèmes sont NP-Complets
- ▶ $G_i \# 1^{k_i} \mathcal{R} G_j \# 1^{k_j}$ si $|V(G_i)| = |V(G_j)| = n$ et $k_i = k_j = k$
- ▶ On construit une instance (G, k_G, Z) de DOMINATING SET paramétré par vertex cover avec Z un vertex cover de G tel que

G admet un ensemble dominant de taille k_G

ssi

$\exists j \in [t]$ tel que G_j possède un vertex cover de taille k_j

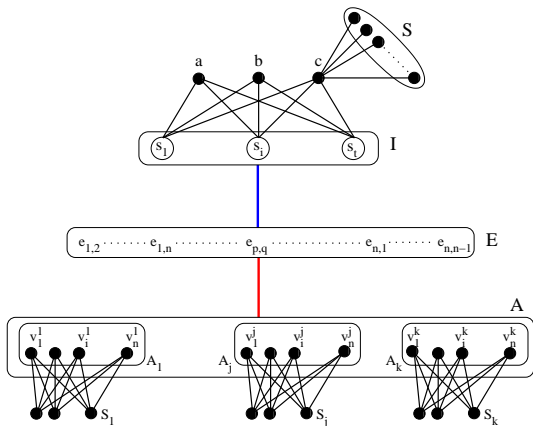


▶ $|S| = k + 3$

▶ $\forall j \in [k], |S_j| = k + 3$

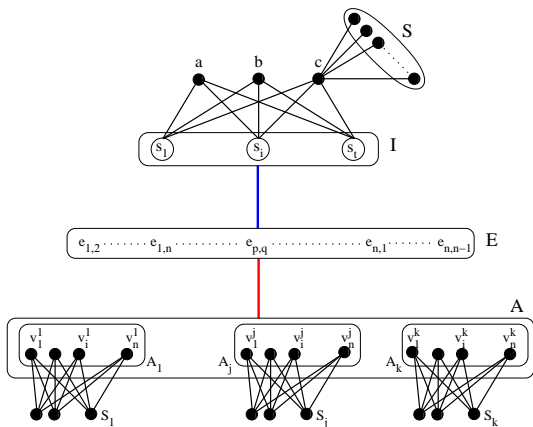
$(s_i, e_{p,q}) \in E(G) \Leftrightarrow (p, q) \notin E(G_i)$

$(v_i^j, e_{p,q}) \in E(G) \Leftrightarrow i = p \vee i = q$



- ▶ $|S| = k + 3$ $(s_i, e_{p,q}) \in E(G) \Leftrightarrow (p, q) \notin E(G_i)$
- ▶ $\forall j \in [k], |S_j| = k + 3$ $(v_i^j, e_{p,q}) \in E(G) \Leftrightarrow i = p \vee i = q$
- ▶ $\forall j \in [k], A_j$ est une clique et S_j un indépendant

G_i possède un vertex cover $S = \{u_1, \dots, u_k\}$ de taille k
 $\Rightarrow \{c, s_i\} \cup \{v_{u_i}^j : i \in [l]\}$ est un $(k + 2)$ -dominant de G



- ▶ $|S| = k + 3$ $(s_i, e_{p,q}) \in E(G) \Leftrightarrow (p, q) \notin E(G_i)$
- ▶ $\forall j \in [k], |S_j| = k + 3$ $(v_i^j, e_{p,q}) \in E(G) \Leftrightarrow i = p \vee i = q$
- ▶ $\forall j \in [k], A_j$ est une clique et S_j un indépendant

G possède un $(k + 2)$ -dominant

$\Rightarrow \exists i \in [k]$ tel que G_i possède un vertex cover de taille k

Noyaux exponentiels

EDGE CLIQUE-COVER

LONGEST PATH

Algorithmes de distillation et de composition

Conjecture de OU-distillation

Non-existence de noyau polynomial

Exemples

Transformations paramétrées polynomiales

Définition et exemple

Réductions non-triviales

Compositions croisées

Définition et théorèmes

Dominating set

Autres techniques

ET-composition

Bornes inférieures polynomiales

ET-composition

Un **algorithme de ET-distillation** \mathcal{A} pour un problème de décision Q (i.e. non paramétré) sur l'alphabet Σ est un algorithme qui :

- ▶ reçoit une séquence (x_1, \dots, x_t) , avec $x_i \in \Sigma^*$, $\forall i \in [t]$;

ET-composition

Un **algorithme de ET-distillation** \mathcal{A} pour un problème de décision Q (i.e. non paramétré) sur l'alphabet Σ est un algorithme qui :

- ▶ reçoit une séquence (x_1, \dots, x_t) , avec $x_i \in \Sigma^*$, $\forall i \in [t]$;
- ▶ possède une complexité **polynomiale** en $\sum_{i=1}^t |x_i|$,

ET-composition

Un **algorithme de ET-distillation** \mathcal{A} pour un problème de décision Q (i.e. non paramétré) sur l'alphabet Σ est un algorithme qui :

- ▶ reçoit une séquence (x_1, \dots, x_t) , avec $x_i \in \Sigma^*$, $\forall i \in [t]$;
- ▶ possède une complexité **polynomiale** en $\sum_{i=1}^t |x_i|$,
- ▶ retourne $y \in \Sigma^*$ tel que
 1. $y \in Q \Leftrightarrow \forall i \in [t], x_i \in Q$;
 2. $|y|$ est **polynomial** en $\max_{i \in [t]} |x_i|$.

ET-composition

Un **algorithme de ET-distillation** \mathcal{A} pour un problème de décision Q (i.e. non paramétré) sur l'alphabet Σ est un algorithme qui :

- ▶ reçoit une séquence (x_1, \dots, x_t) , avec $x_i \in \Sigma^*$, $\forall i \in [t]$;
- ▶ possède une complexité **polynomiale** en $\sum_{i=1}^t |x_i|$,
- ▶ retourne $y \in \Sigma^*$ tel que
 1. $y \in Q \Leftrightarrow \forall i \in [t], x_i \in Q$;
 2. $|y|$ est **polynomial** en $\max_{i \in [t]} |x_i|$.

Conjecture [Bodlaender, Downey, Fellows, Hermelin] Aucun problème NP-complet n'admet un algorithme de ET-distillation

ET-composition

Un algorithme de ET-composition pour un problème paramétré $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$ est un algorithme \mathcal{A} qui

- ▶ reçoit une séquence $((x_1, k), \dots, (x_t, k))$, avec $(x_i, k) \in \Sigma^* \times \mathbb{N}$ $\forall i \in [t]$;
- ▶ a une complexité polynomiale en $\sum_{i=1}^t |x_i| + k$,
- ▶ retourne $(y, k') \in \Sigma^* \times \mathbb{N}$ tel que
 1. $(y, k') \in (Q, \kappa) \Leftrightarrow \forall i \in [t], (x_i, k) \in (Q, \kappa)$;
 2. k' est polynomial en k .

ET-composition

Un algorithme de ET-composition pour un problème paramétré $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$ est un algorithme \mathcal{A} qui

- ▶ reçoit une séquence $((x_1, k), \dots, (x_t, k))$, avec $(x_i, k) \in \Sigma^* \times \mathbb{N} \forall i \in [t]$;
- ▶ a une complexité polynomiale en $\sum_{i=1}^t |x_i| + k$,
- ▶ retourne $(y, k') \in \Sigma^* \times \mathbb{N}$ tel que
 1. $(y, k') \in (Q, \kappa) \Leftrightarrow \forall i \in [t], (x_i, k) \in (Q, \kappa)$;
 2. k' est polynomial en k .

Théorème [Drucker'12]

Soit $(Q, \kappa) \in \Sigma^* \times \mathbb{N}$ un problème paramétré ET-composable tel que le problème $\tilde{Q} \in \Sigma^*$ (non-paramétré) est NP-Complet.

Si (Q, κ) admet un noyau polynomial, alors $\text{coNP} \subseteq \text{NP/poly}$.

ET-composition

Un algorithme de ET-composition pour un problème paramétré $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$ est un algorithme \mathcal{A} qui

- ▶ reçoit une séquence $((x_1, k), \dots, (x_t, k))$, avec $(x_i, k) \in \Sigma^* \times \mathbb{N} \forall i \in [t]$;
- ▶ a une complexité polynomiale en $\sum_{i=1}^t |x_i| + k$,
- ▶ retourne $(y, k') \in \Sigma^* \times \mathbb{N}$ tel que
 1. $(y, k') \in (Q, \kappa) \Leftrightarrow \forall i \in [t], (x_i, k) \in (Q, \kappa)$;
 2. k' est polynomial en k .

Théorème [Drucker'12]

Soit $(Q, \kappa) \in \Sigma^* \times \mathbb{N}$ un problème paramétré ET-composable tel que le problème $\tilde{Q} \in \Sigma^*$ (non-paramétré) est NP-Complet.

Si (Q, κ) admet un noyau polynomial, alors $\text{coNP} \subseteq \text{NP/poly}$.

Conséquence : des problèmes paramétrés tels que TREEWIDTH, PATHWIDTH n'admettent donc pas de noyaux polynomiaux sous des hypothèses standards de complexité (prendre l'union disjointe).

Bornes inférieures polynomiales

Théorème [Dell, van Melkebeek'2010]

Soient $L \subseteq \Sigma^*$ un problème NP-difficile, $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$ un problème paramétré et $d > 0$. S'il existe un algorithme \mathcal{A} qui

- ▶ reçoit une séquence (x_1, \dots, x_t) tel que $\max_{i \in [t]} |x_i| \leq s$
- ▶ a une complexité polynomiale en $\sum_{i=1}^t |x_i|$
- ▶ retourne une instance $(x, k) \in \Sigma^* \times \mathbb{N}$ tel que
 1. $(x, k) \in (Q, \kappa) \Leftrightarrow \exists i \in [t], x_i \in L$
 2. $k \leq t^{\frac{1}{d} + o(1)} \cdot \text{poly}(s)$

alors (Q, κ) ne possède pas de noyau de taille $O(k^{d-\epsilon})$ pour tout $\epsilon > 0$, à moins que $\text{coNP} \subseteq \text{NP}/\text{poly}$.

Bornes inférieures polynomiales

Théorème [Dell, van Melkebeek'2010]

Soient $L \subseteq \Sigma^*$ un problème NP-difficile, $(Q, \kappa) \subseteq \Sigma^* \times \mathbb{N}$ un problème paramétré et $d > 0$. S'il existe un algorithme \mathcal{A} qui

- ▶ reçoit une séquence (x_1, \dots, x_t) tel que $\max_{i \in [t]} |x_i| \leq s$
- ▶ a une complexité polynomiale en $\sum_{i=1}^t |x_i|$
- ▶ retourne une instance $(x, k) \in \Sigma^* \times \mathbb{N}$ tel que
 1. $(x, k) \in (Q, \kappa) \Leftrightarrow \exists i \in [t], x_i \in L$
 2. $k \leq t^{\frac{1}{d} + o(1)} \cdot \text{poly}(s)$

alors (Q, κ) ne possède pas de noyau de taille $O(k^{d-\epsilon})$ pour tout $\epsilon > 0$, à moins que $\text{coNP} \subseteq \text{NP}/\text{poly}$.

Théorème

Pour tout $\epsilon > 0$, le problème p-VERTEX COVER ne possède pas de noyau de taille $O(k^{2-\epsilon})$, à moins que $\text{coNP} \subseteq \text{NP}/\text{poly}$.

Bornes inférieures polynomiales

Réduction depuis **MULTICOLORED BICLIQUE** :

- ▶ étant donné un graphe biparti $B = (X, Y, E)$ tel que $X = X_1 \uplus \dots \uplus X_k$ et $Y = Y_1 \uplus \dots \uplus Y_k$ pour $k > 0$;
- ▶ décider si B contient une biclique $K_{k,k}$ colorée, *i.e.* intersectant chaque X_i, Y_i pour $i \in [k]$.

Bornes inférieures polynomiales

Réduction depuis **MULTICOLORED BICLIQUE** :

- ▶ étant donné un graphe biparti $B = (X, Y, E)$ tel que $X = X_1 \uplus \dots \uplus X_k$ et $Y = Y_1 \uplus \dots \uplus Y_k$ pour $k > 0$;
- ▶ décider si B contient une biclique $K_{k,k}$ colorée, *i.e.* intersectant chaque X_i, Y_i pour $i \in [k]$.

Hypothèses : Soit $(B_{1,1}, \dots, B_{i,j}, \dots, B_{\sqrt{t}, \sqrt{t}})$ une séquences de t graphes bipartis tels que

- ▶ $\forall i \in [t], B_i$ est partitionné en k groupes chacun de taille n .

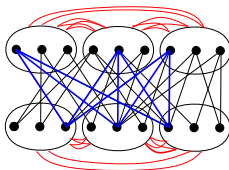
Bornes inférieures polynomiales

Réduction depuis **MULTICOLORED BICLIQUE** :

- ▶ étant donné un graphe biparti $B = (X, Y, E)$ tel que $X = X_1 \uplus \dots \uplus X_k$ et $Y = Y_1 \uplus \dots \uplus Y_k$ pour $k > 0$;
- ▶ décider si B contient une biclique $K_{k,k}$ colorée, *i.e.* intersectant chaque X_i, Y_i pour $i \in [k]$.

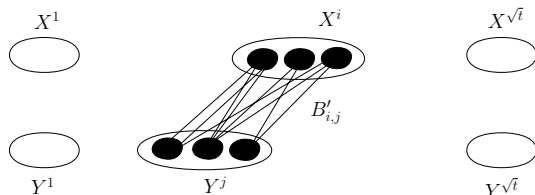
Hypothèses : Soit $(B_{1,1}, \dots, B_{i,j}, \dots, B_{\sqrt{t}, \sqrt{t}})$ une séquences de t graphes bipartis tels que

- ▶ $\forall i \in [t], B_i$ est partitionné en k groupes chacun de taille n .



Observation : $B'_{i,j}$ contient un $2k$ -clique ssi $B_{i,j}$ contient une biclique colorée $K_{k,k}$.

Bornes inférieures polynomiales

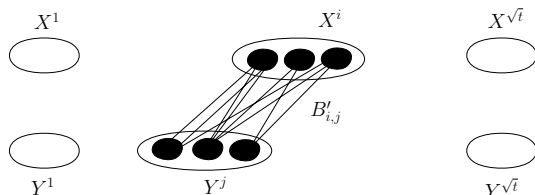


On construit G tel que $G[X^i \cup Y^j]$ est isomorphe à $B'_{i,j}$. Notons que le nombre de sommets de G est

$$N = 2\sqrt{t} \cdot kn = t^{\frac{1}{2}} \cdot \text{poly}(s)$$

avec s la taille maximum des $B_{i,j}$.

Bornes inférieures polynomiales



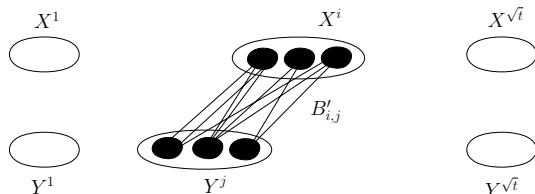
On construit G tel que $G[X^i \cup Y^j]$ est isomorphe à $B'_{i,j}$. Notons que le nombre de sommets de G est

$$N = 2\sqrt{t} \cdot kn = t^{\frac{1}{2}} \cdot \text{poly}(s)$$

avec s la taille maximum des $B_{i,j}$.

- ▶ G possède une $2k$ -clique ssi
- ▶ $\exists i, j \in [\sqrt{t}]$ tel que $B'_{i,j}$ possède une $2k$ -clique ssi
- ▶ $\exists i, j \in [\sqrt{t}]$ tel que $B_{i,j}$ contient un biparti complet coloré $K_{k,k}$

Bornes inférieures polynomiales



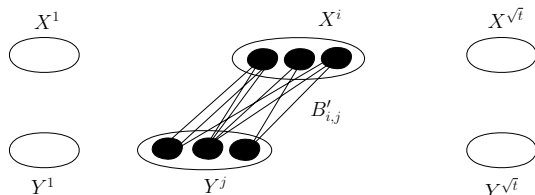
On construit G tel que $G[X^i \cup Y^j]$ est isomorphe à $B'_{i,j}$. Notons que le nombre de sommets de G est

$$N = 2\sqrt{t} \cdot kn = t^{\frac{1}{2}} \cdot \text{poly}(s)$$

avec s la taille maximum des $B_{i,j}$.

- ▶ \overline{G} possède un vertex cover de taille $N - 2k$ ssi
- ▶ $\exists i, j \in [\sqrt{t}]$ tel que $B'_{i,j}$ possède une $2k$ -clique ssi
- ▶ $\exists i, j \in [\sqrt{t}]$ tel que $B_{i,j}$ contient un biparti complet coloré $K_{k,k}$

Bornes inférieures polynomiales



On construit G tel que $G[X^i \cup Y^j]$ est isomorphe à $B'_{i,j}$. Notons que le nombre de sommets de G est

$$N = 2\sqrt{t} \cdot kn = t^{\frac{1}{2}} \cdot \text{poly}(s)$$

avec s la taille maximum des $B_{i,j}$.

Donc $(B_{1,1}, \dots, B_{\sqrt{t}, \sqrt{t}})$, instances de MULTICOLORED BICLIQUE, se réduisent en temps polynomial en une instance $(G, N - 2k)$ de p -VERTEX COVER avec $N - 2k \leq t^{\frac{1}{2}} \cdot \text{poly}(s)$