

Complexité paramétrée (6)

Décomposition et largeur arborescente

Christophe PAUL
(CNRS - LIRMM)

Algorithmes pour les graphes de largeur arborescente bornée

ENSEMBLE INDÉPENDANT sur les arbres

Décomposition arborescente

ENSEMBLE INDÉPENDENT paramétré par $tw(G)$

Exercices (décompositions arborescentes, ENSEMBLE INDÉPENDANT)

CYCLE HAMILTONIEN paramétré par $tw(G)$

Calcul et approximation de la largeur arborescente

Théorème de Courcelle

Logique du second ordre monadique

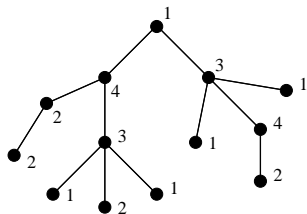
Méta-théorème

Réduction à largeur arborescente bornée

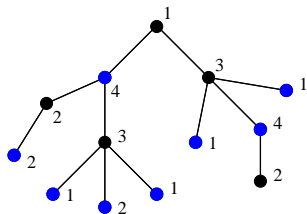
Obstructions

Sommet / arête inutile

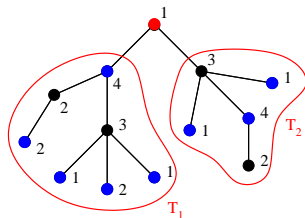
ENSEMBLE INDÉPENDANT (valué) sur les arbres



ENSEMBLE INDÉPENDANT (valué) sur les arbres



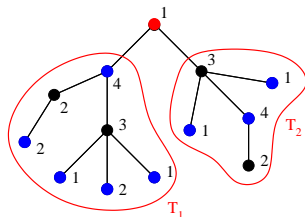
ENSEMBLE INDÉPENDANT (valué) sur les arbres



Observations

1. tout sommet d'un arbre est un séparateur
2. l'union d'ensembles indépendants de composantes connexes distinctes est un ensemble indépendant

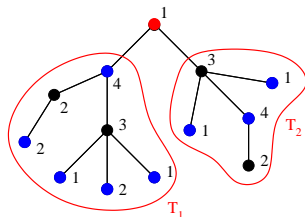
ENSEMBLE INDÉPENDANT (valué) sur les arbres



Soit x la racine de T , $x_1 \dots x_l$ ses fils et T_1, \dots, T_l les sous-arbres de $T - x$:

- ▶ $wIS(T, x)$ → ensemble indépendant max. contenant x
- ▶ $wIS(T, \bar{x})$ → ensemble indépendant max. ne contenant pas x

ENSEMBLE INDÉPENDANT (valué) sur les arbres

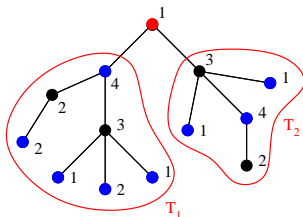


Soit x la racine de T , $x_1 \dots x_l$ ses fils et T_1, \dots, T_l les sous-arbres de $T - x$:

- ▶ $wIS(T, x)$ → ensemble indépendant max. contenant x
- ▶ $wIS(T, \bar{x})$ → ensemble indépendant max. ne contenant pas x

$$\left\{ \begin{array}{l} wIS(T, x) = \sum_{i \in [l]} wIS(T_i, \bar{x}_i) + \omega(x) \end{array} \right.$$

ENSEMBLE INDÉPENDANT (valué) sur les arbres

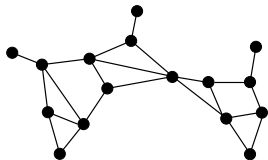


Soit x la racine de T , $x_1 \dots x_l$ ses fils et T_1, \dots, T_l les sous-arbres de $T - x$:

- ▶ $wIS(T, x)$ → ensemble indépendant max. contenant x
- ▶ $wIS(T, \bar{x})$ → ensemble indépendant max. ne contenant pas x

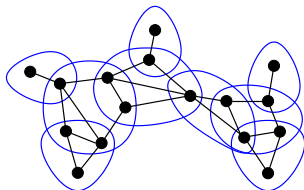
$$\begin{cases} wIS(T, x) &= \sum_{i \in [l]} wIS(T_i, \bar{x}_i) + \omega(x) \\ wIS(T, \bar{x}) &= \sum_{i \in [l]} \max\{wIS(T_i, x_i), wIS(T_i, \bar{x}_i)\} \end{cases}$$

Décomposition arborescente



Une **décomposition arborescente** d'un graphe $G = (V, E)$ est une paire $(T, \{X_t : t \in T\})$ avec T est un arbre et $\forall t \in T, X_t \subseteq V$, telle que

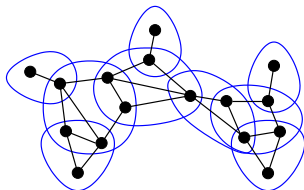
Décomposition arborescente



Une **décomposition arborescente** d'un graphe $G = (V, E)$ est une paire $(T, \{X_t : t \in T\})$ avec T est un arbre et $\forall t \in T, X_t \subseteq V$, telle que

- ▶ **[couverture des sommets]** $\forall x \in V, \exists t \in T$ tel que $x \in X_t$

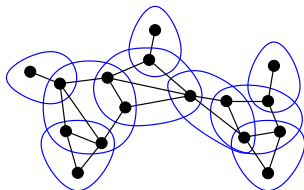
Décomposition arborescente



Une **décomposition arborescente** d'un graphe $G = (V, E)$ est une paire $(T, \{X_t : t \in T\})$ avec T est un arbre et $\forall t \in T, X_t \subseteq V$, telle que

- ▶ [couverture des sommets] $\forall x \in V, \exists t \in T$ tel que $x \in X_t$
- ▶ [couverture des arêtes] $\forall (x, y) \in E, \exists t \in T$ tel que $x, y \in X_t$

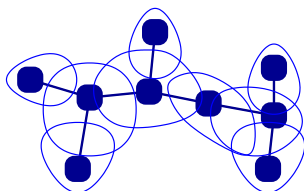
Décomposition arborescente



Une **décomposition arborescente** d'un graphe $G = (V, E)$ est une paire $(T, \{X_t : t \in T\})$ avec T est un arbre et $\forall t \in T, X_t \subseteq V$, telle que

- ▶ [couverture des sommets] $\forall x \in V, \exists t \in T$ tel que $x \in X_t$
- ▶ [couverture des arêtes] $\forall (x, y) \in E, \exists t \in T$ tel que $x, y \in X_t$
- ▶ [consistance] si $x \in V$ appartient à $X_{t_1} \cap X_{t_2}$, alors $\forall t \in T$ sur le chemin entre t_1 et t_2 dans $T, x \in X_t$.

Décomposition arborescente



Une **décomposition arborescente** d'un graphe $G = (V, E)$ est une paire $(T, \{X_t : t \in T\})$ avec T est un arbre et $\forall t \in T, X_t \subseteq V$, telle que

- ▶ [couverture des sommets] $\forall x \in V, \exists t \in T$ tel que $x \in X_t$
- ▶ [couverture des arêtes] $\forall (x, y) \in E, \exists t \in T$ tel que $x, y \in X_t$
- ▶ [consistance] si $x \in V$ appartient à $X_{t_1} \cap X_{t_2}$, alors $\forall t \in T$ sur le chemin entre t_1 et t_2 dans $T, x \in X_t$.

Décomposition arborescente

- ▶ La *largeur* d'une décomposition arborescente

$\mathcal{T}_G = (T, \{X_t : t \in T\})$ de G est

$$\text{width}(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$

Décomposition arborescente

- ▶ La *largeur* d'une décomposition arborescente

$\mathcal{T}_G = (T, \{X_t : t \in T\})$ de G est

$$\text{width}(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$

- ▶ La *largeur arborescente* d'un graphe G est

$$\text{tw}(G) = \min_{\mathcal{T}_G} \text{width}(\mathcal{T}_G)$$

Décomposition arborescente

- ▶ La **largeur** d'une décomposition arborescente

$\mathcal{T}_G = (T, \{X_t : t \in T\})$ de G est

$$\text{width}(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$

- ▶ La **largeur arborescente** d'un graphe G est

$$\text{tw}(G) = \min_{\mathcal{T}_G} \text{width}(\mathcal{T}_G)$$

Observation Soit $\mathcal{T}_G = (T, \{X_t : t \in T\})$ une décomposition arborescente de G

1. tout nœud X_t est un **séparateur** de G .

Décomposition arborescente

- ▶ La **largeur** d'une décomposition arborescente

$\mathcal{T}_G = (T, \{X_t : t \in T\})$ de G est

$$\text{width}(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$

- ▶ La **largeur arborescente** d'un graphe G est

$$\text{tw}(G) = \min_{\mathcal{T}_G} \text{width}(\mathcal{T}_G)$$

Observation Soit $\mathcal{T}_G = (T, \{X_t : t \in T\})$ une décomposition arborescente de G

1. tout nœud X_t est un **séparateur** de G .
2. si X_t et $X_{t'}$ sont deux nœuds tels que t et t' sont adjacents dans T , alors $X_t \cap X_{t'}$ est un **séparateur** de G .

Décomposition arborescente

- ▶ La **largeur** d'une décomposition arborescente

$\mathcal{T}_G = (T, \{X_t : t \in T\})$ de G est

$$\text{width}(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$

- ▶ La **largeur arborescente** d'un graphe G est

$$\text{tw}(G) = \min_{\mathcal{T}_G} \text{width}(\mathcal{T}_G)$$

Observation Soit $\mathcal{T}_G = (T, \{X_t : t \in T\})$ une décomposition arborescente de G

1. tout nœud X_t est un **séparateur** de G .
2. si X_t et $X_{t'}$ sont deux nœuds tels que t et t' sont adjacents dans T , alors $X_t \cap X_{t'}$ est un **séparateur** de G .

Notations : Si on enracine $\mathcal{T}_G = (T, \{X_t : t \in T\})$, alors :

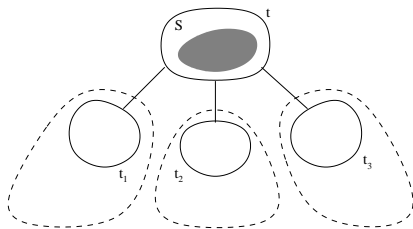
- ▶ V_t : l'ensemble de sommets présents dans les descendants de t
- ▶ G_t : le sous-graphe $G[V_t]$

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

$\forall S \subseteq X_t, IS(S, t) = \text{indép. max. de } G_t \text{ tq } X_t \cap IS(S, t) = S$

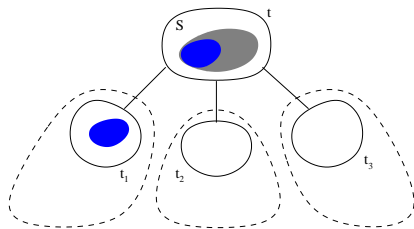
ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

$\forall S \subseteq X_t, IS(S, t) = \text{indép. max. de } G_t \text{ tq } X_t \cap IS(S, t) = S$



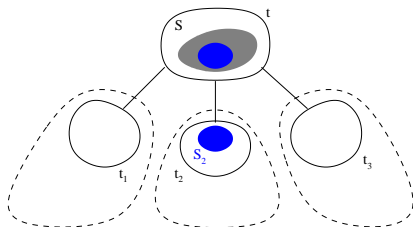
ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

$\forall S \subseteq X_t, IS(S, t) = \text{indép. max. de } G_t \text{ tq } X_t \cap IS(S, t) = S$



ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

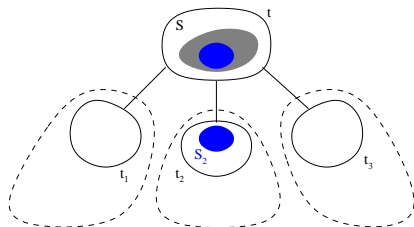
$\forall S \subseteq X_t, IS(S, t) = \text{indép. max. de } G_t \text{ tq } X_t \cap IS(S, t) = S$



Lemme : Si $S \subseteq X_t$ et $S_j = S \cap X_{t_j}$ alors $IS(S, t) \cap V_{t_j} = IS(S_j, t_j)$

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

$\forall S \subseteq X_t, IS(S, t) = \text{indép. max. de } G_t \text{ tq } X_t \cap IS(S, t) = S$

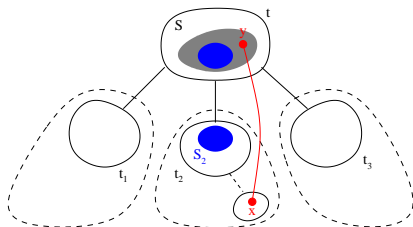


Lemme : Si $S \subseteq X_t$ et $S_j = S \cap X_{t_j}$ alors $IS(S, t) \cap V_{t_j} = IS(S_j, t_j)$

Hyp. : $IS(S, t) \cap V_{t_j}$ n'est pas max.

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

$\forall S \subseteq X_t, IS(S, t) = \text{indép. max. de } G_t \text{ tq } X_t \cap IS(S, t) = S$



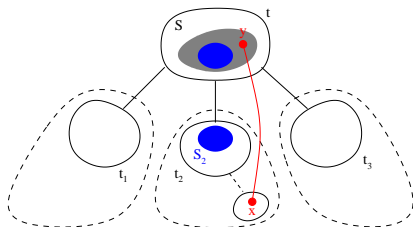
Lemme : Si $S \subseteq X_t$ et $S_j = S \cap X_{t_j}$ alors $IS(S, t) \cap V_{t_j} = IS(S_j, t_j)$

Hyp. : $IS(S, t) \cap V_{t_j}$ n'est pas max.

$\Rightarrow \exists y \in S \setminus S_j$ et $\exists x \in IS(S_j, t_j) \setminus X_{t_j}$ tels que $xy \in E$

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

$\forall S \subseteq X_t, IS(S, t) = \text{indép. max. de } G_t \text{ tq } X_t \cap IS(S, t) = S$



Lemme : Si $S \subseteq X_t$ et $S_j = S \cap X_{t_j}$ alors $IS(S, t) \cap V_{t_j} = IS(S_j, t_j)$

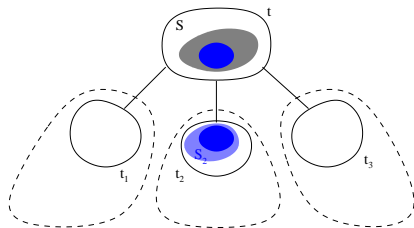
Hyp. : $IS(S, t) \cap V_{t_j}$ n'est pas max.

$\Rightarrow \exists y \in S \setminus S_j$ et $\exists x \in IS(S_j, t_j) \setminus X_{t_j}$ tels que $xy \in E$

► **contradiction :** X_{t_j} est un séparateur

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

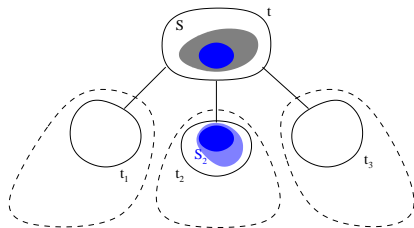
Idée de l'algo de programmation dynamique



Calcul de $IS(S, t)$ connaissant $IS(S_j^i, t_j)$, $\forall j \in [1]$, $\forall S_j^i \subseteq X_{t_j}$

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

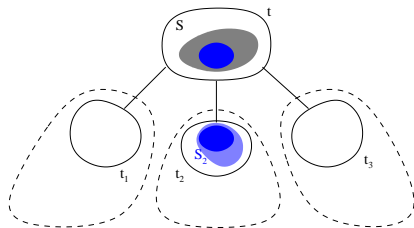
Idée de l'algo de programmation dynamique



Calcul de $IS(S, t)$ connaissant $IS(S_j^i, t_j)$, $\forall j \in [l]$, $\forall S_j^i \subseteq X_{t_j}$

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

Idee de l'algo de programmation dynamique

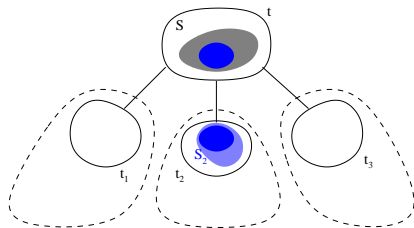


Calcul de $IS(S, t)$ connaissant $IS(S_j^i, t_j)$, $\forall j \in [l]$, $\forall S_j^i \subseteq X_{t_j}$

- vérifier que $S_j^i \cap X_t = S \cap X_{t_j} = S_j$ et $S_j \subseteq S_j^i$

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

Idee de l'algo de programmation dynamique

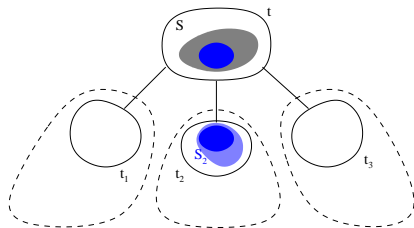


Calcul de $IS(S, t)$ connaissant $IS(S_j^i, t_j)$, $\forall j \in [l]$, $\forall S_j^i \subseteq X_{t_j}$

- ▶ vérifier que $S_j^i \cap X_t = S \cap X_{t_j} = S_j$ et $S_j \subseteq S_j^i$
- ▶ vérifier que S_j^i est un indépendant

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

Idee de l'algo de programmation dynamique



Calcul de $IS(S, t)$ connaissant $IS(S_j^i, t_j)$, $\forall j \in [l]$, $\forall S_j^i \subseteq X_{t_j}$

- ▶ vérifier que $S_j^i \cap X_t = S \cap X_{t_j} = S_j$ et $S_j \subseteq S_j^i$
- ▶ vérifier que S_j^i est un indépendant

$$IS(S, t) = \left\{ \sum_{i \in [l]} \max \begin{array}{l} |S| + \\ \{IS(S_j^i, t_j) - |S_j|\} : \\ S_j^i \cap X_t = S_j \text{ \& } S_j \subseteq S_j^i \text{ indépendant} \end{array} \right.$$

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

$$IS(S, t) = \left\{ \sum_{i \in [l]} \max \begin{array}{l} |S| + \\ \{IS(S_j^i, t_j) - |S_j| : \\ S_j^i \cap X_t = S_j \text{ \& } S_j \subseteq S_j^i \text{ indépendant} \} \end{array} \right\}$$

Analyse de complexité :

- ▶ calcul de $IS(S, t)$: $O(2^k \cdot k^2 \cdot l)$

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

$$IS(S, t) = \left\{ \sum_{i \in [l]} \max \begin{array}{l} |S| + \\ \{IS(S_j^i, t_j) - |S_j| : \\ S_j^i \cap X_t = S_j \text{ \& } S_j \subseteq S_j^i \text{ indépendant} \} \end{array} \right\}$$

Analyse de complexité :

- ▶ calcul de $IS(S, t)$: $O(2^k \cdot k^2 \cdot l)$
- ▶ calcul de $IS(S, t)$ pour tout $S \subseteq X_t$: $O(2^k \cdot 2^k \cdot k^2 \cdot l)$

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

$$IS(S, t) = \left\{ \sum_{i \in [l]} \max \begin{array}{l} |S| + \\ \{IS(S_j^i, t_j) - |S_j| : \\ S_j^i \cap X_t = S_j \text{ \& } S_j \subseteq S_j^i \text{ indépendant} \} \end{array} \right\}$$

Analyse de complexité :

- ▶ calcul de $IS(S, t)$: $O(2^k \cdot k^2 \cdot l)$
- ▶ calcul de $IS(S, t)$ pour tout $S \subseteq X_t$: $O(2^k \cdot 2^k \cdot k^2 \cdot l)$
- ▶ calcul de la solution : $O(4^k \cdot k^2 \cdot n)$

ENSEMBLE INDÉPENDANT (pondéré) paramétré $tw(G)$

$$IS(S, t) = \left\{ \sum_{i \in [l]} \max \begin{array}{l} |S| + \\ \{IS(S_j^i, t_j) - |S_j|\} : \\ S_j^i \cap X_t = S_j \text{ \& } S_j \subseteq S_j^i \text{ indépendant} \end{array} \right\}$$

Analyse de complexité :

- ▶ calcul de $IS(S, t)$: $O(2^k \cdot k^2 \cdot l)$
- ▶ calcul de $IS(S, t)$ pour tout $S \subseteq X_t$: $O(2^k \cdot 2^k \cdot k^2 \cdot l)$
- ▶ calcul de la solution : $O(4^k \cdot k^2 \cdot n)$

- ▶ il faut ajouter le temps de calcul d'une décomposition arborescente optimale!!!

Décomposition arborescente simple

Une décomposition arborescente enracinée $\mathcal{T}_G = (T, \{X_t : t \in T\})$ est **simple** si tout nœud t est d'un des quatre types suivant :

- ▶ **Feuille** : pas de fils et $|X_t| = 1$



Décomposition arborescente simple

Une décomposition arborescente enracinée $\mathcal{T}_G = (T, \{X_t : t \in T\})$ est **simple** si tout nœud t est d'un des quatre types suivant :

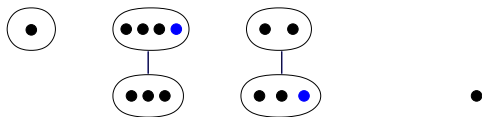
- ▶ **Feuille** : pas de fils et $|X_t| = 1$
- ▶ **Ajout** : un fils unique t' et $X_t = X_{t'} \cup \{v\}$ avec $v \notin X_{t'}$



Décomposition arborescente simple

Une décomposition arborescente enracinée $\mathcal{T}_G = (T, \{X_t : t \in T\})$ est **simple** si tout nœud t est d'un des quatre types suivant :

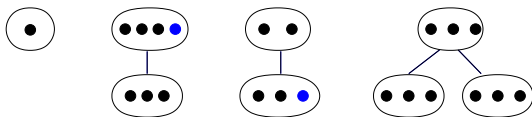
- ▶ **Feuille** : pas de fils et $|X_t| = 1$
- ▶ **Ajout** : un fils unique t' et $X_t = X_{t'} \cup \{v\}$ avec $v \notin X_{t'}$
- ▶ **Suppression** : un fils unique t' et $X_t = X_{t'} \setminus \{v\}$ avec $v \in X_{t'}$



Décomposition arborescente simple

Une décomposition arborescente enracinée $\mathcal{T}_G = (T, \{X_t : t \in T\})$ est **simple** si tout nœud t est d'un des quatre types suivant :

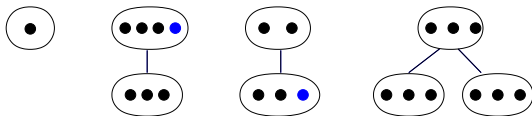
- ▶ **Feuille** : pas de fils et $|X_t| = 1$
- ▶ **Ajout** : un fils unique t' et $X_t = X_{t'} \cup \{v\}$ avec $v \notin X_{t'}$
- ▶ **Suppression** : un fils unique t' et $X_t = X_{t'} \setminus \{v\}$ avec $v \in X_{t'}$
- ▶ **Fusion** : deux fils t_1 et t_2 et $X_t = X_{t_1} \cup X_{t_2}$.



Décomposition arborescente simple

Une décomposition arborescente enracinée $\mathcal{T}_G = (T, \{X_t : t \in T\})$ est **simple** si tout nœud t est d'un des quatre types suivant :

- ▶ **Feuille** : pas de fils et $|X_t| = 1$
- ▶ **Ajout** : un fils unique t' et $X_t = X_{t'} \cup \{v\}$ avec $v \notin X_{t'}$
- ▶ **Suppression** : un fils unique t' et $X_t = X_{t'} \setminus \{v\}$ avec $v \in X_{t'}$
- ▶ **Fusion** : deux fils t_1 et t_2 et $X_t = X_{t_1} \cup X_{t_2}$.



Observation :

Une décomposition arborescente \mathcal{T} de largeur k avec c nœuds peut être transformée en temps $O(kn)$ en une décomposition arborescente simple de largeur k avec $k^2 \cdot c$ nœuds.

Algorithme amélioré pour ENSEMBLE INDÉPENDANT

Calcul de $IS(S, t)$ pour tout $S \subseteq X_t$

- ▶ t est une **feuille** : trivial

Algorithme amélioré pour ENSEMBLE INDÉPENDANT

Calcul de $IS(S, t)$ pour tout $S \subseteq X_t$

- ▶ t est une **feuille** : trivial
- ▶ t est un nœud **ajout** : $X_t = X_{t'} \cup \{v\}$

$$IS(S, t) = \begin{cases} IS(S, t') & \text{si } v \notin S \\ IS(S \setminus \{v\}, t') + \omega(v) & \text{si } v \in S \text{ et } S \text{ indépendant} \\ -\infty & \text{sinon} \end{cases}$$

Algorithme amélioré pour ENSEMBLE INDÉPENDANT

Calcul de $IS(S, t)$ pour tout $S \subseteq X_t$

- ▶ t est une **feuille** : trivial
- ▶ t est un nœud **ajout** : $X_t = X_{t'} \cup \{v\}$

$$IS(S, t) = \begin{cases} IS(S, t') & \text{si } v \notin S \\ IS(S \setminus \{v\}, t') + \omega(v) & \text{si } v \in S \text{ et } S \text{ indépendant} \\ -\infty & \text{sinon} \end{cases}$$

- ▶ t est un nœud **suppression** : $X_t = X_{t'} \setminus \{v\}$

$$IS(S, t) = \max\{IS(S, t'), IS(S \cup \{v\}, t')\}$$

Algorithme amélioré pour ENSEMBLE INDÉPENDANT

Calcul de $IS(S, t)$ pour tout $S \subseteq X_t$

- ▶ t est une **feuille** : trivial
- ▶ t est un nœud **ajout** : $X_t = X_{t'} \cup \{v\}$

$$IS(S, t) = \begin{cases} IS(S, t') & \text{si } v \notin S \\ IS(S \setminus \{v\}, t') + \omega(v) & \text{si } v \in S \text{ et } S \text{ indépendant} \\ -\infty & \text{sinon} \end{cases}$$

- ▶ t est un nœud **suppression** : $X_t = X_{t'} \setminus \{v\}$

$$IS(S, t) = \max\{IS(S, t'), IS(S \cup \{v\}, t')\}$$

- ▶ t est un nœud **fusion** : $X_t = X_{t_1} = X_{t_2}$

$$IS(S, t) = IS(S, t_1) + IS(S, t_2) - \omega(S)$$

Algorithme amélioré pour ENSEMBLE INDÉPENDANT

Calcul de $IS(S, t)$ pour tout $S \subseteq X_t$

- ▶ t est une **feuille** : trivial
- ▶ t est un nœud **ajout** : $X_t = X_{t'} \cup \{v\}$

$$IS(S, t) = \begin{cases} IS(S, t') & \text{si } v \notin S \\ IS(S \setminus \{v\}, t') + \omega(v) & \text{si } v \in S \text{ et } S \text{ indépendant} \\ -\infty & \text{sinon} \end{cases}$$

- ▶ t est un nœud **suppression** : $X_t = X_{t'} \setminus \{v\}$

$$IS(S, t) = \max\{IS(S, t'), IS(S \cup \{v\}, t')\}$$

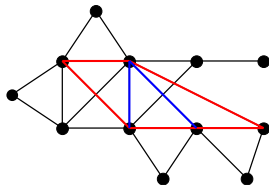
- ▶ t est un nœud **fusion** : $X_t = X_{t_1} = X_{t_2}$

$$IS(S, t) = IS(S, t_1) + IS(S, t_2) - \omega(S)$$

Complexité : $O(2^k \cdot n)$

ENSEMBLE INDÉPENDANT et graphes triangulés

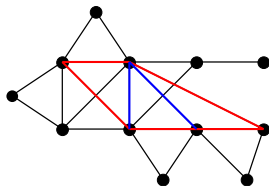
Un graphe $G = (V, E)$ est **triangulé** si tout cycle de longueur au moins 4 possède une corde.



ENSEMBLE INDÉPENDANT et graphes triangulés

Un graphe $G = (V, E)$ est **triangulé** si tout cycle de longueur au moins 4 possède une corde.

Théorème [Dirac] : Si un graphe triangulé n'est pas une clique, alors il contient un séparateur complet.



Exercice : Montrer que

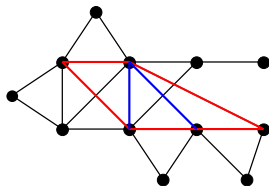
1. Tout graphe triangulé G possède une décomposition arborescente dont les nœuds sont les cliques maximales de G .

\rightsquigarrow **Théorème** : Soit S un séparateur complet d'un graphe G .
 G est triangulé ssi $\forall C$, composante connexe de $G - S$,
 $G[C \cup S]$ est triangulé.

ENSEMBLE INDÉPENDANT et graphes triangulés

Un graphe $G = (V, E)$ est **triangulé** si tout cycle de longueur au moins 4 possède une corde.

Théorème [Dirac] : Si un graphe triangulé n'est pas une clique, alors il contient un séparateur complet.



Exercice : Montrer que

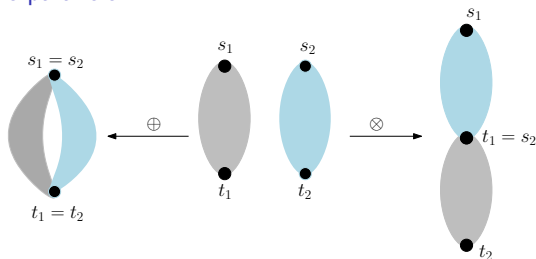
1. Tout graphe triangulé G possède une décomposition arborescente dont les nœuds sont les cliques maximales de G .

\rightsquigarrow **Théorème** : Soit S un séparateur complet d'un graphe G .
 G est triangulé ssi $\forall C$, composante connexe de $G - S$,
 $G[C \cup S]$ est triangulé.

2. ENSEMBLE INDÉPENDANT est polynomial sur les graphes triangulés

ENSEMBLE INDÉPENDANT et graphes série-parallèles

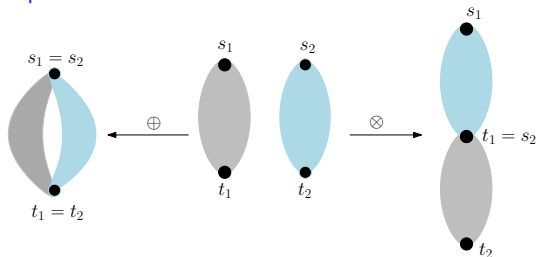
Un graphe G est **série-parallèle** s'il existe deux sommets s et t tel que G est (s, t) -série-parallèle.



$\rightsquigarrow G = (V = \{s, t\}, E = \{(s, t)\})$ est (s, t) -série-parallèle.

ENSEMBLE INDÉPENDANT et graphes série-parallèles

Un graphe G est **série-parallèle** s'il existe deux sommets s et t tel que G est (s, t) -série-parallèle.

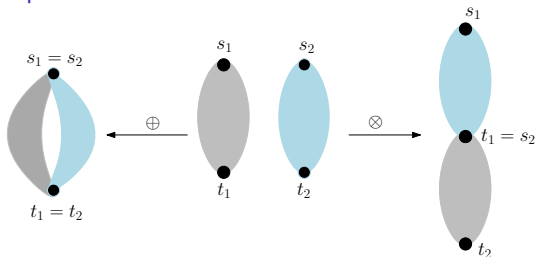


Exercice :

1. Soient G_1 et G_2 des graphes avec une seule arête. Dessiner le graphe $((G_1 \oplus G_2) \otimes G_1) \oplus ((G_1 \otimes (G_1 \oplus (G_1 \otimes G_2))) \otimes G_1)$.
2. Décrire un algorithme de calcul d'une décomposition arborescente d'un graphe série-parallèle à partir de son expression régulière

ENSEMBLE INDÉPENDANT et graphes série-parallèles

Un graphe G est **série-parallèle** s'il existe deux sommets s et t tel que G est (s, t) -série-parallèle.



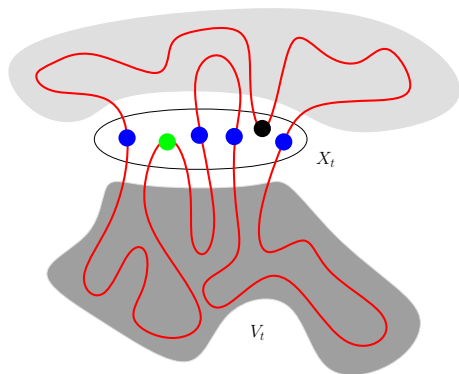
Exercice :

1. Soient G_1 et G_2 des graphes avec une seule arête. Dessiner le graphe $((G_1 \oplus G_2) \otimes G_1) \oplus ((G_1 \otimes (G_1 \oplus (G_1 \otimes G_2))) \otimes G_1)$.
2. Décrire un algorithme de calcul d'une décomposition arborescente d'un graphe série-parallèle à partir de son expression régulière
3. En déduire que pour tout graphe G série-parallèle, $\mathbf{tw}(G) \leq 2$
4. Quel est la complexité de ENSEMBLE INDÉPENDANT sur les graphes séries-parallèles ?

CYCLE HAMILTONIEN paramétré par $tw(G)$

Soit \mathcal{C} un cycle hamiltonien.

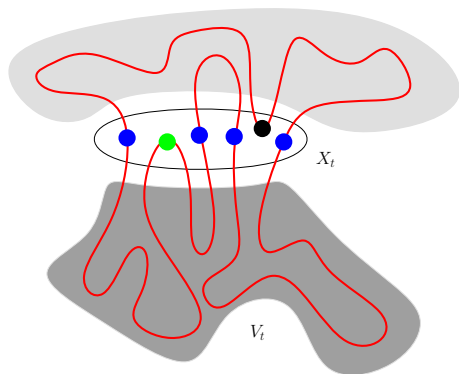
- ▶ $\mathcal{C} \cap G[V_t]$ est une collection de chemins



CYCLE HAMILTONIEN paramétré par $tw(G)$

Soit \mathcal{C} un cycle hamiltonien.

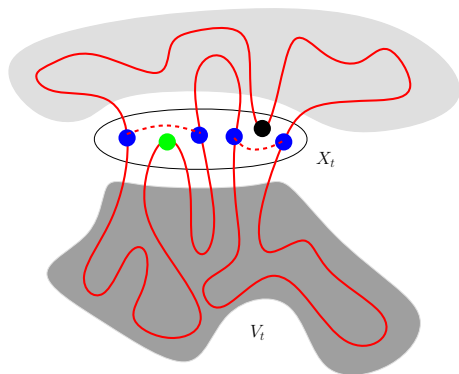
- ▶ $\mathcal{C} \cap G[V_t]$ est une collection de chemins
- ▶ Les sommets de X_t sont :
 - ▶ isolés : X_t^0
 - ▶ des extrémités : X_t^1
 - ▶ des sommets internes : X_t^2



CYCLE HAMILTONIEN paramétré par $tw(G)$

Soit \mathcal{C} un cycle hamiltonien.

- ▶ $\mathcal{C} \cap G[V_t]$ est une collection de chemins
- ▶ Les sommets de X_t sont :
 - ▶ isolés : X_t^0
 - ▶ des extrémités : X_t^1
 - ▶ des sommets internes : X_t^2



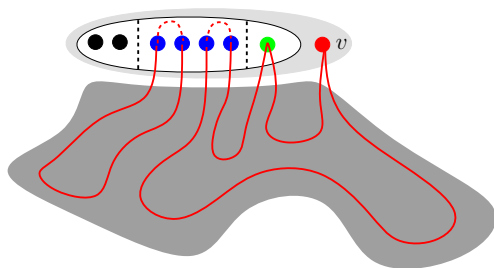
Pour chaque nœud t de la décomposition arborescente, il faut savoir si

$$(X_t^0, X_t^1, X_t^2, M)$$

où M est un couplage sur X_t^1 , est une solution partielle.

Nœud Suppression

Soit t un nœud suppression et t' son fils tel que $X_t = X_{t'} \setminus \{v\}$

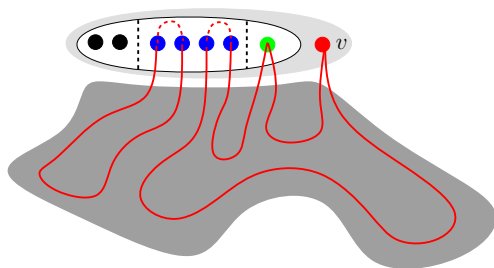


Obs. : X_t est un séparateur \Rightarrow

$\forall v \in V_t \setminus X_t, v$ est interne dans toute solution partielle

Nœud Suppression

Soit t un nœud suppression et t' son fils tel que $X_t = X_{t'} \setminus \{v\}$



Obs. : X_t est un séparateur \Rightarrow

$\forall v \in V_t \setminus X_t$, v est interne dans toute solution partielle

$(X_{t'}^0, X_{t'}^1, X_{t'}^2 \setminus \{v\}, M)$ est une solution partielle pour t

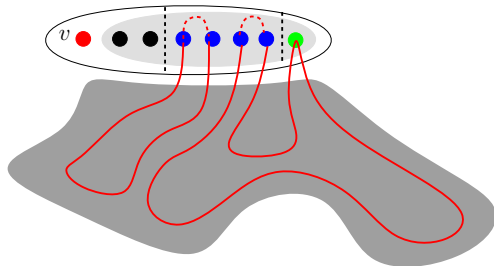
\Leftrightarrow

$(X_{t'}^0, X_{t'}^1, X_{t'}^2, M)$ est une solution partielle pour t'

Nœud Ajout

Soit t un nœud ajout et t' son fils tel que $X_t = X_{t'} \cup \{v\}$

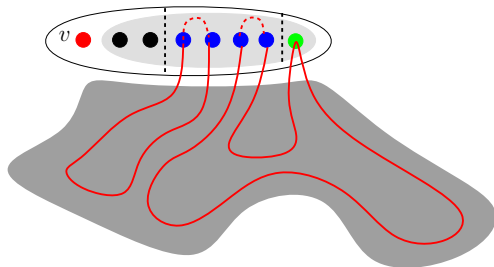
► Hyp. : $v \in X_t^0$



Nœud Ajout

Soit t un nœud ajout et t' son fils tel que $X_t = X_{t'} \cup \{v\}$

► Hyp. : $v \in X_t^0$



$(X_{t'}^0 \cup \{v\}, X_{t'}^1, X_{t'}^2, M)$ est une solution partielle pour t

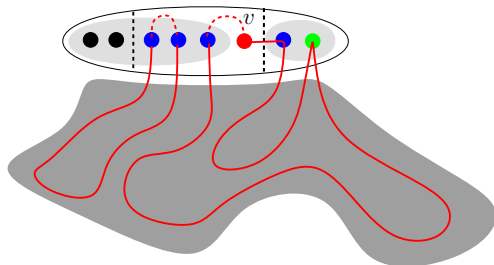
\Leftrightarrow

$(X_{t'}^0, X_{t'}^1, X_{t'}^2, M)$ est une solution partielle pour t'

Nœud Ajout

Soit t un nœud ajout et t' son fils tel que $X_t = X_{t'} \cup \{v\}$

- ▶ Hyp. : $v \in X_t^1$



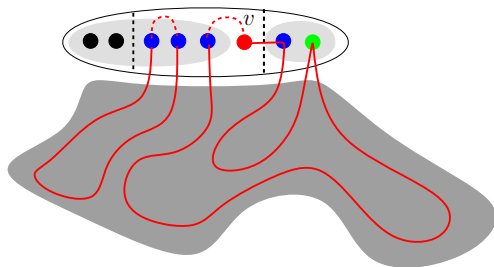
Obs. : $X_{t'}$ est un séparateur $\Rightarrow N(v) \subset X_t$

- ▶ un sommet $u \in X_{t'}^1$ devient interne $\Rightarrow u \in X_t^2$

Nœud Ajout

Soit t un nœud ajout et t' son fils tel que $X_t = X_{t'} \cup \{v\}$

- ▶ Hyp. : $v \in X_t^1$



Obs. : $X_{t'}$ est un séparateur $\Rightarrow N(v) \subset X_t$

- ▶ un sommet $u \in X_{t'}^1$ devient interne $\Rightarrow u \in X_t^2$

$(X_{t'}^0, X_{t'}^1 \cup \{v\} \setminus \{u\}, X_{t'}^2 \cup \{u\}, M')$ est une solution partielle pour t

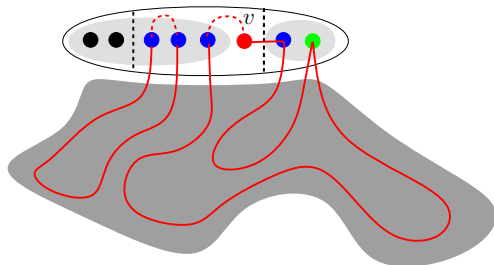
\Leftrightarrow

$(X_{t'}^0, X_{t'}^1, X_{t'}^2, M)$ est une solution partielle pour t'

Nœud Ajout

Soit t un nœud ajout et t' son fils tel que $X_t = X_{t'} \cup \{v\}$

- ▶ Hyp. : $v \in X_t^1$



Obs. : $X_{t'}$ est un séparateur $\Rightarrow N(v) \subset X_t$

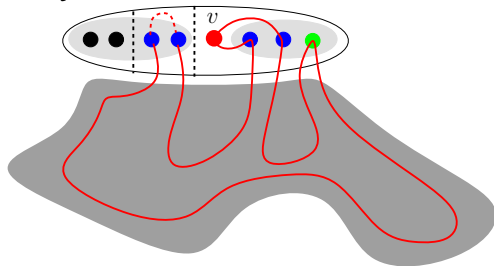
- ▶ un sommet $u \in X_{t'}^1$ devient interne $\Rightarrow u \in X_t^2$
- ▶ ou un sommet $w \in X_{t'}^0$ devient extrémité d'un chemin

$\Rightarrow w \in X_t^1$

Nœud Ajout

Soit t un nœud ajout et t' son fils tel que $X_t = X_{t'} \cup \{v\}$

- ▶ Hyp. : $v \in X_t^2$



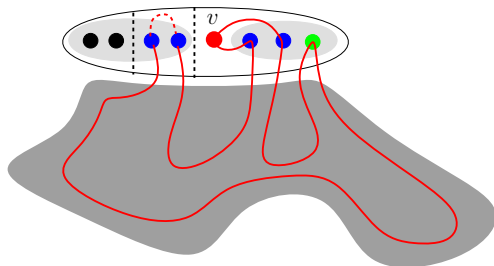
Obs. : $X_{t'}$ est un séparateur $\Rightarrow N(v) \subset X_t$

- ▶ deux sommets $u, u' \in X_{t'}^1$ deviennent internes $\Rightarrow u, u' \in X_t^2$

Nœud Ajout

Soit t un nœud ajout et t' son fils tel que $X_t = X_{t'} \cup \{v\}$

- ▶ Hyp. : $v \in X_t^2$



Obs. : $X_{t'}$ est un séparateur $\Rightarrow N(v) \subset X_t$

- ▶ deux sommets $u, u' \in X_{t'}^1$ deviennent internes $\Rightarrow u, u' \in X_t^2$

$(X_{t'}^0, X_{t'}^1 \setminus \{u, u'\}, X_{t'}^2 \cup \{v, u, u'\}, M')$ est une solution partielle pour t

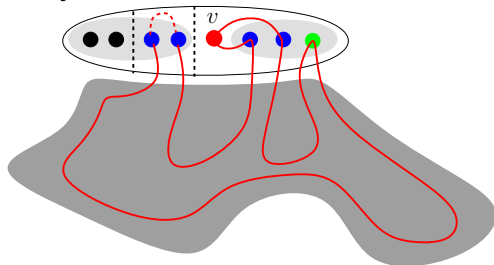
\Leftrightarrow

$(X_{t'}^0, X_{t'}^1, X_{t'}^2, M)$ est une solution partielle pour t'

Nœud Ajout

Soit t un nœud ajout et t' son fils tel que $X_t = X_{t'} \cup \{v\}$

- ▶ Hyp. : $v \in X_t^2$

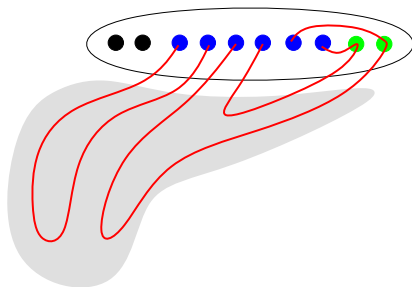


Obs. : $X_{t'}$ est un séparateur $\Rightarrow N(v) \subset X_t$

- ▶ deux sommets $u, u' \in X_{t'}^1$ deviennent internes $\Rightarrow u, u' \in X_t^2$
- ▶ deux sommets $w, w' \in X_{t'}^0$ deviennent extrémités
 $\Rightarrow w, w' \in X_t^1$
- ▶ $w \in X_{t'}^0$ devient extrémité et $v \in X_{t'}^1$ devient interne
 $\Rightarrow w \in X_t^1, v \in X_t^2$

Nœud Fusion

Soit t un nœud fusion et t_1, t_2 ses fils tels que $X_t = X_{t_1} = X_{t_2}$

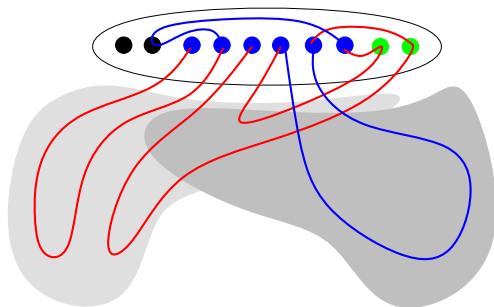


Obs. : pour être compatible, les solutions partielles doivent vérifier

- ▶ $X_{t_1}^2 \subseteq X_{t_2}^0$ et $X_{t_1}^1 \subseteq X_{t_2}^1 \cup X_{t_2}^0$
- ▶ $X_{t_2}^2 \subseteq X_{t_1}^0$ et $X_{t_2}^1 \subseteq X_{t_1}^1 \cup X_{t_1}^0$
- ▶ L'union des couplages M_1 et M_2 ne crée pas de cycle

Nœud Fusion

Soit t un nœud fusion et t_1, t_2 ses fils tels que $X_t = X_{t_1} = X_{t_2}$



Obs. : pour être compatible, les solutions partielles doivent vérifier

- ▶ $X_{t_1}^2 \subseteq X_{t_2}^0$ et $X_{t_1}^1 \subseteq X_{t_2}^1 \cup X_{t_2}^0$
- ▶ $X_{t_2}^2 \subseteq X_{t_1}^0$ et $X_{t_2}^1 \subseteq X_{t_1}^1 \cup X_{t_1}^0$
- ▶ L'union des couplages M_1 et M_2 ne crée pas de cycle

CYCLE HAMILTONIEN paramétré par $tw(G)$

Théorème

Etant donnée une décomposition arborescente de largeur ω ,
tw-CYCLE HAMILTONIEN peut-être résolu en temps

$$\omega^{O(\omega)} \cdot n$$

CYCLE HAMILTONIEN paramétré par $tw(G)$

Théorème

Etant donnée une décomposition arborescente de largeur ω ,
tw-CYCLE HAMILTONIEN peut-être résolu en temps

$$\omega^{O(\omega)} \cdot n$$

- ▶ nombre de sous problèmes pour chaque nœud : $3^\omega \cdot \omega!$

CYCLE HAMILTONIEN paramétré par $tw(G)$

Théorème

Etant donnée une décomposition arborescente de largeur ω ,
tw-CYCLE HAMILTONIEN peut-être résolu en temps

$$\omega^{O(\omega)} \cdot n$$

- ▶ nombre de sous problèmes pour chaque nœud : $3^\omega \cdot \omega!$
- ▶ nombre de nœuds dans une décomposition simple : $\omega \cdot n$

CYCLE HAMILTONIEN paramétré par $tw(G)$

Théorème

Etant donnée une décomposition arborescente de largeur ω ,
tw-CYCLE HAMILTONIEN peut-être résolu en temps

$$\omega^{O(\omega)} \cdot n$$

- ▶ nombre de sous problèmes pour chaque nœud : $3^\omega \cdot \omega!$
- ▶ nombre de nœuds dans une décomposition simple : $\omega \cdot n$

Exercice : donner un algorithme de programmation dynamique pour les problèmes

1. **tw-COLORATION**
2. **tw-FEEDBACK VERTEX SET**

Algorithmes pour les graphes de largeur arborescente bornée

ENSEMBLE INDÉPENDANT sur les arbres

Décomposition arborescente

ENSEMBLE INDÉPENDENT paramétré par $tw(G)$

Exercices (décompositions arborescentes, ENSEMBLE INDÉPENDANT)

CYCLE HAMILTONIEN paramétré par $tw(G)$

Calcul et approximation de la largeur arborescente

Théorème de Courcelle

Logique du second ordre monadique

Méta-théorème

Réduction à largeur arborescente bornée

Obstructions

Sommet / arête inutile

Théorème de Bodlaender

Théorème [Arnborg, Corneil, Proskurowski]

Etant donné un graphe G et un entier k , décider si $tw(G) \leq k$ est un problème NP-difficile.

Théorème [Bodlaender]

Etant donné un graphe G et un entier k fixé, il existe un algorithme de complexité $O(2^{k^3} \cdot n)$ qui décide si $tw(G) \leq k$

Théorème de Bodlaender

Théorème [Arnborg, Corneil, Proskurowski]

Etant donné un graphe G et un entier k , décider si $tw(G) \leq k$ est un problème NP-difficile.

Théorème [Bodlaender]

Etant donné un graphe G et un entier k fixé, il existe un algorithme de complexité $O(2^{k^3} \cdot n)$ qui décide si $tw(G) \leq k$

Théorème [Diestel et al.]

Il existe un algorithme de complexité $O(3^{3k} \cdot k \cdot n)$ qui calcule une décomposition arborescente \mathcal{T} tel que $width(\mathcal{T}) \leq 4k + 1$ si $tw(G) \leq k$

Théorème de Bodlaender

Théorème [Arnborg, Corneil, Proskurowski]

Etant donné un graphe G et un entier k , décider si $tw(G) \leq k$ est un problème NP-difficile.

Théorème [Bodlaender]

Etant donné un graphe G et un entier k fixé, il existe un algorithme de complexité $O(2^{k^3} \cdot n)$ qui décide si $tw(G) \leq k$

Théorème [Diestel et al.]

Il existe un algorithme de complexité $O(3^{3k} \cdot k \cdot n)$ qui calcule une décomposition arborescente \mathcal{T} tel que $width(\mathcal{T}) \leq 4k + 1$ si $tw(G) \leq k$

Problème ouvert : existence d'un noyau polynomial

Algorithme de Diestel et al.

Quelques définitions

- ▶ Y et Z , $|Y| = |Z|$ sont **séparables** si V contient un ensemble S tel que $|S| < |Y|$ et $G - S$ ne contient pas de chemin entre $Y \setminus S$ et $Z \setminus S$.

Algorithme de Diestel et al.

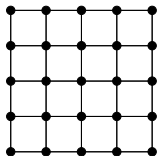
Quelques définitions

- ▶ Y et Z , $|Y| = |Z|$ sont **séparables** si V contient un ensemble S tel que $|S| < |Y|$ et $G - S$ ne contient pas de chemin entre $Y \setminus S$ et $Z \setminus S$.
- ▶ Un ensemble X est **k -lié** si $|X| \geq k$ et $\forall Y, Z \subseteq X, |Y| = |Z| \leq k$, Y and Z sont **non-séparables**

Algorithme de Diestel et al.

Quelques définitions

- ▶ Y et Z , $|Y| = |Z|$ sont **séparables** si V contient un ensemble S tel que $|S| < |Y|$ et $G - S$ ne contient pas de chemin entre $Y \setminus S$ et $Z \setminus S$.
- ▶ Un ensemble X est **k -lié** si $|X| \geq k$ et $\forall Y, Z \subseteq X, |Y| = |Z| \leq k$, Y and Z sont **non-séparables**

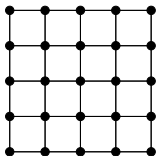


La grille $k \times k$ est k -liée

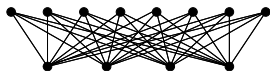
Algorithme de Diestel et al.

Quelques définitions

- ▶ Y et Z , $|Y| = |Z|$ sont **séparables** si V contient un ensemble S tel que $|S| < |Y|$ et $G - S$ ne contient pas de chemin entre $Y \setminus S$ et $Z \setminus S$.
- ▶ Un ensemble X est **k -lié** si $|X| \geq k$ et $\forall Y, Z \subseteq X, |Y| = |Z| \leq k$, Y and Z sont **non-séparables**



La grille $k \times k$ est k -liée



Dans le biparti complet $K_{2k,k}$:

A est $2k$ -lié

B est k -lié

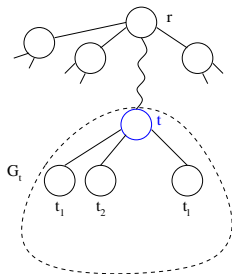
Algorithme de Diestel et al.

Lemme : Si G contient un ensemble $(k + 1)$ -lié X tq $|X| \geq 3k$,
alors $tw(G) \geq k$

Algorithme de Diestel et al.

Lemme : Si G contient un ensemble $(k + 1)$ -lié X tq $|X| \geq 3k$,
alors $tw(G) \geq k$

Soit \mathcal{T}_G une décomposition arborescente tq $width(\mathcal{T}_G) < k$

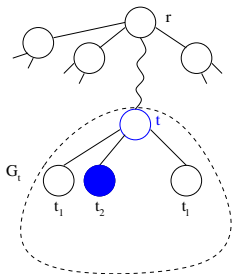


Soit t le nœud le plus "bas" tq $|V_t \cap X| \geq 2k$

Algorithme de Diestel et al.

Lemme : Si G contient un ensemble $(k + 1)$ -lié X tq $|X| \geq 3k$,
alors $tw(G) \geq k$

Soit \mathcal{T}_G une décomposition arborescente tq $width(\mathcal{T}_G) < k$



Soit t le nœud le plus "bas" tq $|V_t \cap X| \geq 2k$

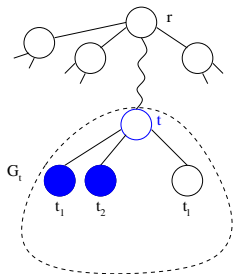
Si $\exists i \in [l]$ tq $|V_{t_i} \cap X| \geq k$, alors
 $Y \subseteq V_{t_i} \cap X$, $|Y| = k$ et $Z \subset V \setminus V_{t_i}$, $|Z| = k$

Mais $S = X_{t_i} \cap X_t$ sépare Y de Z et
 $|S| \leq k - 1$.

Algorithme de Diestel et al.

Lemme : Si G contient un ensemble $(k + 1)$ -lié X tq $|X| \geq 3k$,
alors $tw(G) \geq k$

Soit \mathcal{T}_G une décomposition arborescente tq $width(\mathcal{T}_G) < k$



Soit t le nœud le plus "bas" tq $|V_t \cap X| \geq 2k$

Sinon $W = V_{t_1} \cup \dots \cup V_{t_i}$ avec $|W \cap X| \geq k$
et $|(W \setminus V_{t_j}) \cap X| < k$ pour $1 \leq j \leq i$

$Y \subseteq W \cap X$, $|Y| = k$ et $Z \subseteq V \setminus W$, $|Z| = k$

Mais $S = X_t$ sépare Y de Z et $|S| \leq k - 1$.

Algorithme de Diestel et al.

Lemme : Etant donné un ensemble de sommets X d'un graphe G , on peut décider si X est k -lié (pour $k \leq |X|$) en temps $O(f(k).n^{O(1)})$.

Algorithme de Diestel et al.

Lemme : Etant donné un ensemble de sommets X d'un graphe G , on peut décider si X est k -lié (pour $k \leq |X|$) en temps $O(f(k).n^{O(1)})$.

- ▶ pour toute paire de sous-ensembles Y et Z de X telle que $|Y| = |Z| \leq k$, on teste si Y et Z sont séparables (algorithme de flôts)

Algorithme de Diestel et al.

Lemme : Etant donné un ensemble de sommets X d'un graphe G , on peut décider si X est k -lié (pour $k \leq |X|$) en temps $O(f(k).n^{O(1)})$.

- ▶ pour toute paire de sous-ensembles Y et Z de X telle que $|Y| = |Z| \leq k$, on teste si Y et Z sont séparables (algorithme de flôts)
- ▶ complexité : $O(4^k.n^{O(1)})$

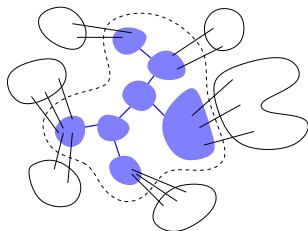
Algorithme de Diestel et al.

Lemme : Etant donné un ensemble de sommets X d'un graphe G , on peut décider si X est k -lié (pour $k \leq |X|$) en temps $O(f(k).n^{O(1)})$.

- ▶ pour toute paire de sous-ensembles Y et Z de X telle que $|Y| = |Z| \leq k$, on teste si Y et Z sont séparables (algorithme de flôts)
- ▶ complexité : $O(4^k.n^{O(1)})$

Remarque : Si X n'est pas k -lié, on trouve deux sous-ensembles séparables.

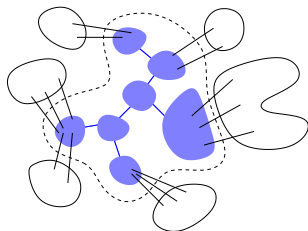
Algorithme de Diestel et al.



Principe

- ▶ On ajoute les sommets dans U de manière gloutonne

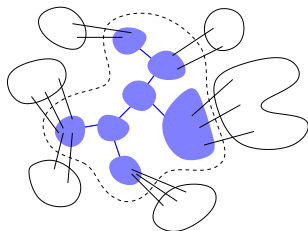
Algorithme de Diestel et al.



Principe

- ▶ On ajoute les sommets dans U de manière gloutonne
- ▶ On maintient une décomposition arborescente \mathcal{T}_U tq $width(\mathcal{T}_U) \leq 4k$

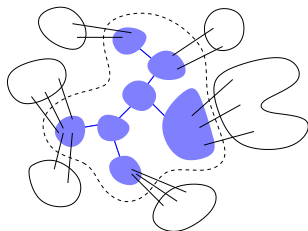
Algorithme de Diestel et al.



Principe

- ▶ On ajoute les sommets dans U de manière gloutonne
- ▶ On maintient une décomposition arborescente \mathcal{T}_U tq $width(\mathcal{T}_U) \leq 4k$
- ▶ **Invariant** : Chaque composante connexe C de $G - U$ possède au plus $3k$ voisins dans U , et il existe un nœud t tel que V_t les contient tous.

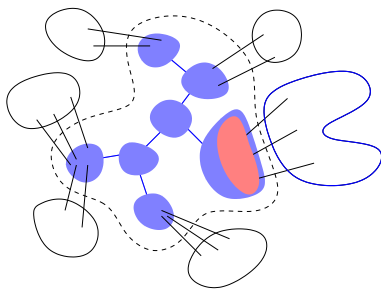
Algorithme de Diestel et al.



Principe

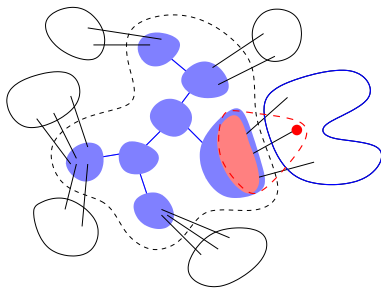
- ▶ On ajoute les sommets dans U de manière gloutonne
- ▶ On maintient une décomposition arborescente \mathcal{T}_U tq $width(\mathcal{T}_U) \leq 4k$
- ▶ **Invariant** : Chaque composante connexe C de $G - U$ possède au plus $3k$ voisins dans U , et il existe un nœud t tel que V_t les contient tous.
- ▶ Initialement on choisit V_1 de taille $3k$.

Algorithme de Diestel et al.



Soit X les voisins d'une composante C et t le nœud tq $X \subseteq X_t$.

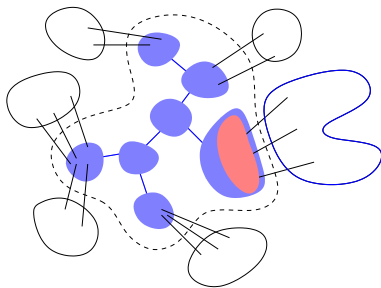
Algorithme de Diestel et al.



Soit X les voisins d'une composante C et t le nœud tq $X \subseteq X_t$.

- ▶ Si $|X| < 3k$: on ajoute un nœud t' voisin de t tel que $X_{t'} = \{x\} \cup X$ avec $x \in C$ un voisin de X_t

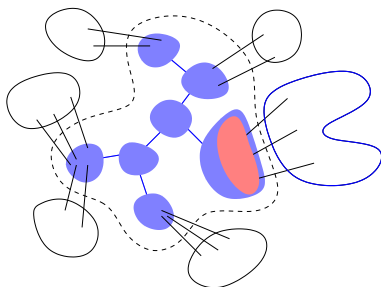
Algorithme de Diestel et al.



Soit X les voisins d'une composante C et t le nœud tq $X \subseteq X_t$.

- ▶ Si $|X| = 3k$: on teste si X est $(k + 1)$ -lié

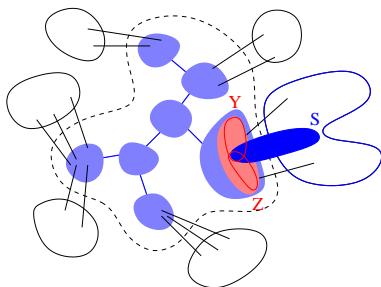
Algorithme de Diestel et al.



Soit X les voisins d'une composante C et t le nœud tq $X \subseteq X_t$.

- ▶ Si $|X| = 3k$: on teste si X est $(k + 1)$ -lié
 1. Si X est $(k + 1)$ -lié alors $tw(G) \geq k + 1$

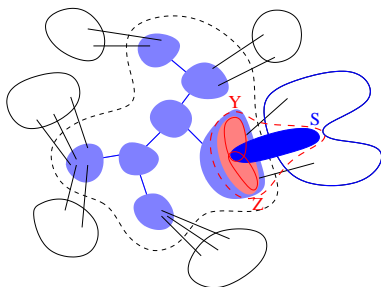
Algorithme de Diestel et al.



Soit X les voisins d'une composante C et t le nœud tq $X \subseteq X_t$.

- ▶ Si $|X| = 3k$: on teste si X est $(k + 1)$ -lié
 1. Si X est $(k + 1)$ -lié alors $tw(G) \geq k + 1$
 2. Sinon on trouve Y , Z et S tq $|S| < |Y| = |Z| \leq k + 1$ et S sépare Y et Z

Algorithme de Diestel et al.

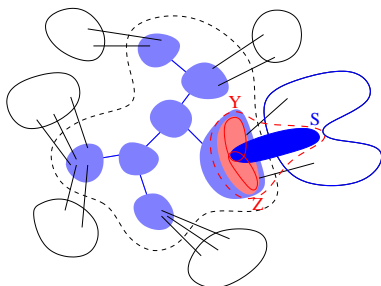


Soit X les voisins d'une composante C et t le nœud tq $X \subseteq X_t$.

- ▶ Si $|X| = 3k$: on teste si X est $(k + 1)$ -lié
 1. Si X est $(k + 1)$ -lié alors $tw(G) \geq k + 1$
 2. Sinon on trouve Y , Z et S tq $|S| < |Y| = |Z| \leq k + 1$ et S sépare Y et Z

On crée un nœud t' voisin de t tel que $X_{t'} = (S \cap C) \cup X$

Algorithme de Diestel et al.



Soit X les voisins d'une composante C et t le nœud tq $X \subseteq X_t$.

- ▶ Si $|X| = 3k$: on teste si X est $(k + 1)$ -lié
 1. Si X est $(k + 1)$ -lié alors $tw(G) \geq k + 1$
 2. Sinon on trouve Y , Z et S tq $|S| < |Y| = |Z| \leq k + 1$ et S sépare Y et Z

On crée un nœud t' voisin de t tel que $X_{t'} = (S \cap C) \cup X$

Obs : les voisins de chaque nouvelle composante $C' \subseteq C$ sont dans $(X \setminus Z) \cup (S \cap C)$ ou dans $(X \setminus Y) \cup (S \cap C) \Rightarrow$ au plus $3k$ voisins

Algorithmes pour les graphes de largeur arborescente bornée

ENSEMBLE INDÉPENDANT sur les arbres

Décomposition arborescente

ENSEMBLE INDÉPENDENT paramétré par $tw(G)$

Exercices (décompositions arborescentes, ENSEMBLE INDÉPENDANT)

CYCLE HAMILTONIEN paramétré par $tw(G)$

Calcul et approximation de la largeur arborescente

Théorème de Courcelle

Logique du second ordre monadique

Méta-théorème

Réduction à largeur arborescente bornée

Obstructions

Sommet / arête inutile

Logique du second ordre monadique sur les graphes

On représente un graphe $G = (V, E)$ à l'aide de la structure $\mathcal{G} = (U, \textit{Vertex}, \textit{Edge}, I)$ où

- ▶ $U = V \cup E$ est l'univers
- ▶ *Vertex* et *Edge* sont des relations **unaires** permettant de distinguer les sommets et les arêtes
- ▶ $I = \{(v, e) \mid v \in V, e \in E, v \in e\}$ est la **relation d'incidence**.

Logique du second ordre monadique sur les graphes

On représente un graphe $G = (V, E)$ à l'aide de la structure $\mathcal{G} = (U, \text{Vertex}, \text{Edge}, I)$ où

- ▶ $U = V \cup E$ est l'univers
- ▶ *Vertex* et *Edge* sont des relations **unaires** permettant de distinguer les sommets et les arêtes
- ▶ $I = \{(v, e) \mid v \in V, e \in E, v \in e\}$ est la **relation d'incidence**.

Une formule MSOL est construite à partir

- ▶ des connecteurs logiques $\vee, \wedge, \Rightarrow, \neg, =, \neq$
- ▶ prédicats *adj*(u, v) et *inc*(e, v)
- ▶ des quantificateurs \exists, \forall sur des variables de sommets / arêtes ou d'ensembles de sommets / arêtes
- ▶ des relations \in, \subseteq sur les ensemble de sommets / arêtes

Logique du second ordre monadique sur les graphes

Exemple : G est un graphe connexe

- ▶ pour toute **bipartition** de V , il existe une **arête transverse**

Logique du second ordre monadique sur les graphes

Exemple : G est un graphe connexe

► pour toute **bipartition** de V , il existe une **arête transverse**

$$\forall V_1, V_2, [\forall v \in V, (v \in V_1 \vee v \in V_2) \wedge (v \in V_1 \Rightarrow v \notin V_2) \wedge (v \in V_2 \Rightarrow v \notin V_1)] \\ \wedge \exists v_1 \in V_1, \exists v_2 \in V_2, \exists e \in E, inc(v_1, e) \wedge inc(v_2, e)$$

Logique du second ordre monadique sur les graphes

Exemple : G est un graphe connexe

- ▶ pour toute **bipartition** de V , il existe une **arête transverse**

$$\forall V_1, V_2, [\forall v \in V, (v \in V_1 \vee v \in V_2) \wedge (v \in V_1 \Rightarrow v \notin V_2) \wedge (v \in V_2 \Rightarrow v \notin V_1)] \\ \wedge \exists v_1 \in V_1, \exists v_2 \in V_2, \exists e \in E, inc(v_1, e) \wedge inc(v_2, e)$$

Exercice : Exprimer qu'un graphe G

- ▶ possède un vertex cover, un ensemble indépendant, un ensemble dominant de taille k ...
- ▶ est k -colorable
- ▶ possède un cycle hamiltonien

Théorème [Courcelle]

Toute propriété de graphe exprimable en MSOL peut être testée en temps $O(f(k).n)$ sur les graphes de largeur arborescente au plus k (pour k fixé).

Théorème [Courcelle]

Toute propriété de graphe exprimable en MSOL peut être testée en temps $O(f(k).n)$ sur les graphes de largeur arborescente au plus k (pour k fixé).

Corollaire : VERTEX COVER, ENSEMBLE INDÉPENDANT, DOMINATING SET paramétrés par la largeur arborescente sont des problèmes FPT

Théorème [Courcelle]

Toute propriété de graphe exprimable en MSOL peut être testée en temps $O(f(k).n)$ sur les graphes de largeur arborescente au plus k (pour k fixé).

Corollaire : VERTEX COVER, ENSEMBLE INDÉPENDANT, DOMINATING SET paramétrés par la largeur arborescente sont des problèmes FPT

Remarque : la fonction $f(k)$ dépend de la structure de la formule

Théorème [Courcelle]

Toute propriété de graphe exprimable en MSOL peut être testée en temps $O(f(k).n)$ sur les graphes de largeur arborescente au plus k (pour k fixé).

Corollaire : VERTEX COVER, ENSEMBLE INDÉPENDANT, DOMINATING SET paramétrés par la largeur arborescente sont des problèmes FPT

Remarque : la fonction $f(k)$ dépend de la structure de la formule

Utilisation du théorème de Courcelle

1. Montrer que le problème est exprimable en MSOL
2. Montrer que si $tw(G)$ est trop grande (par rapport au paramètre k), alors
 - ▶ l'instance est négative (G contient une obstruction)
 - ▶ ou l'instance peut-être réduite (on diminue $tw(G)$)

Algorithmes pour les graphes de largeur arborescente bornée

ENSEMBLE INDÉPENDANT sur les arbres

Décomposition arborescente

ENSEMBLE INDÉPENDENT paramétré par $tw(G)$

Exercices (décompositions arborescentes, ENSEMBLE INDÉPENDANT)

CYCLE HAMILTONIEN paramétré par $tw(G)$

Calcul et approximation de la largeur arborescente

Théorème de Courcelle

Logique du second ordre monadique

Méta-théorème

Réduction à largeur arborescente bornée

Obstructions

Sommet / arête inutile

Dominating set dans les graphes planaires (bidim)

