Complexité et algorithmes paramétrés (3)

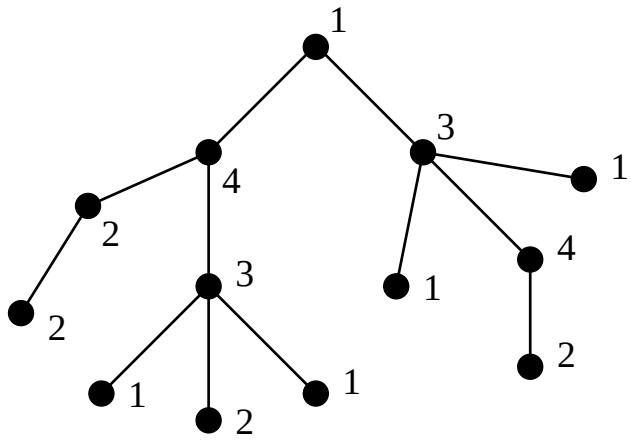
-

Décomposition arborescente et programmation dynamique

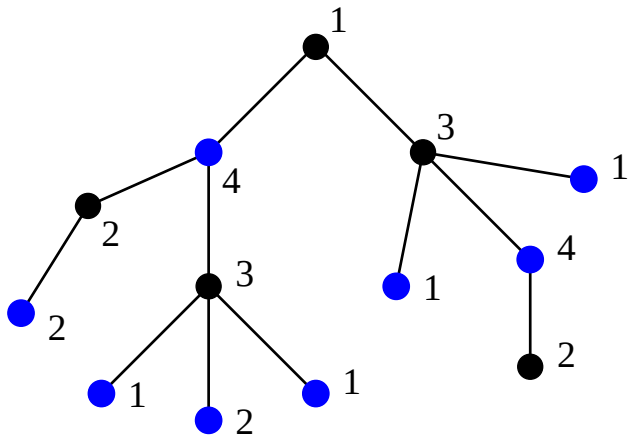
Christophe PAUL
CNRS - LIRMM

EJC Informatique Mathématique du GDR IM
Perpignan, avril 2013

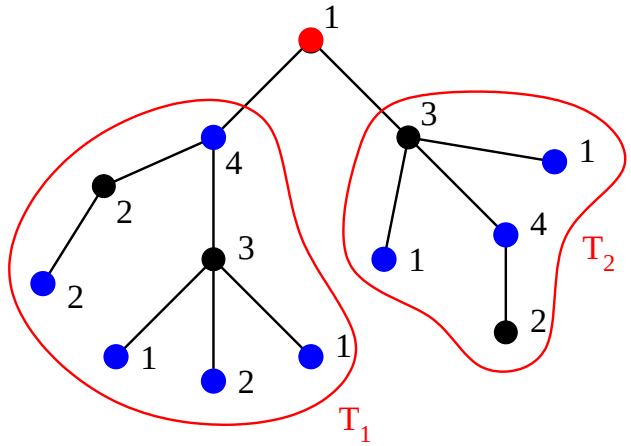
Ensemble Indépendant (valué) sur les arbres



Ensemble Indépendant (valué) sur les arbres



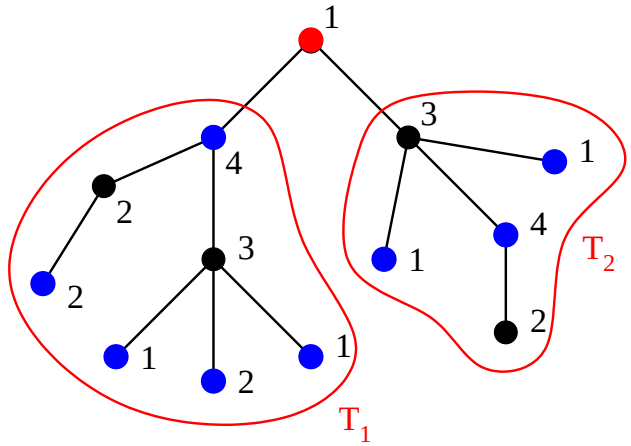
Ensemble Indépendant (valué) sur les arbres



Observations

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

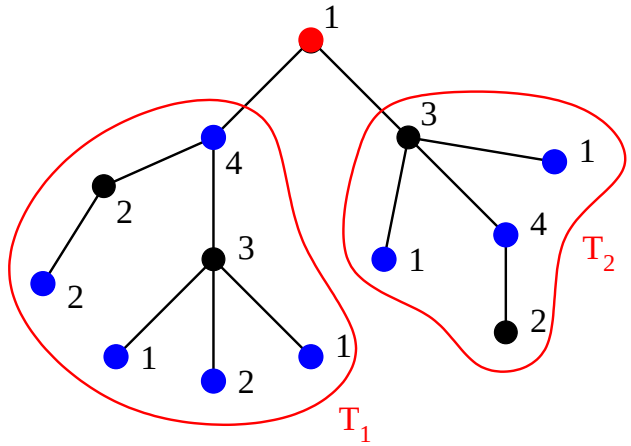
Ensemble Indépendant (valué) sur les arbres



Soit x la racine de T et $x_1 \dots x_l$ ses fils :

- $wIS(T, x) \rightarrow$ ensemble indépendant max. contenant x
- $wIS(T, \overline{x}) \rightarrow$ ensemble indépendant max. ne contenant pas x

Ensemble Indépendant (valué) sur les arbres

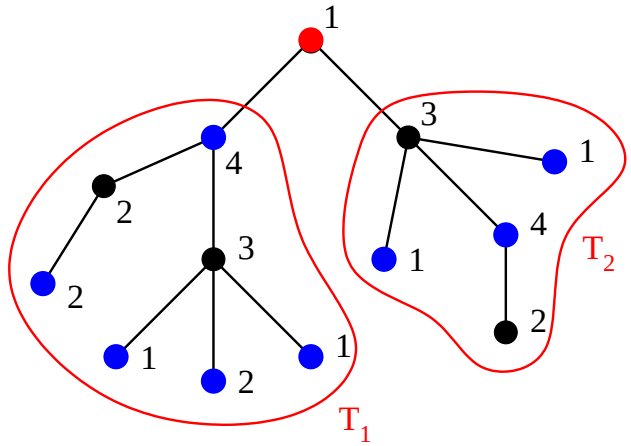


Soit x la racine de T et $x_1 \dots x_l$ ses fils :

- $wIS(T, x) \rightarrow$ ensemble indépendant max. contenant x
- $wIS(T, \overline{x}) \rightarrow$ ensemble indépendant max. ne contenant pas x

$$\begin{cases} wIS(T, x) & = & \sum_{i \in [l]} wIS(T, \overline{x_i}) \\ & & \end{cases}$$

Ensemble Indépendant (valué) sur les arbres

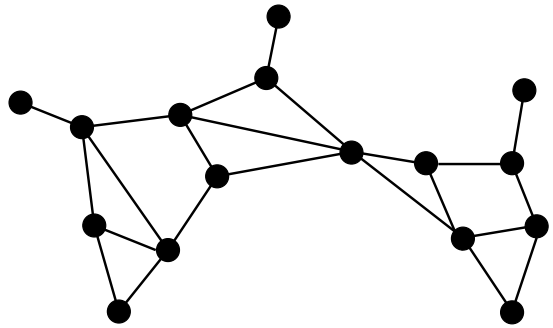


Soit x la racine de T et $x_1 \dots x_l$ ses fils :

- $wIS(T, x) \rightarrow$ ensemble indépendant max. contenant x
- $wIS(T, \overline{x}) \rightarrow$ ensemble indépendant max. ne contenant pas x

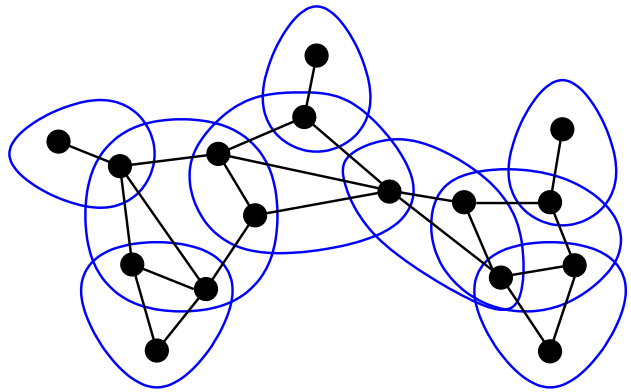
$$\begin{cases} wIS(T, x) & = & \sum_{i \in [l]} wIS(T, \overline{x_i}) \\ wIS(T, \overline{x}) & = & \sum_{i \in [l]} \max\{wIS(T, x_i), wIS(T, \overline{x_i})\} \end{cases}$$

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 $\forall t \in T$, $V_t \subseteq V$, telle que

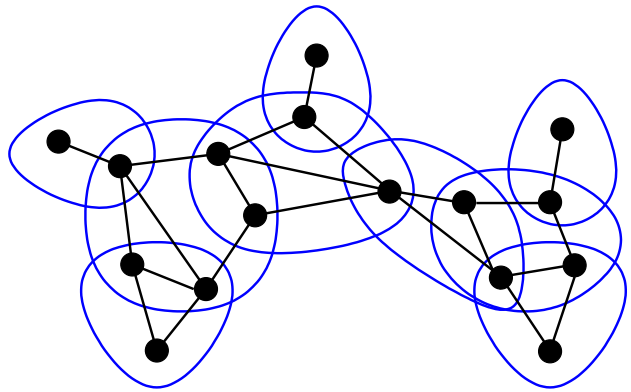
Décomposition arborescente



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

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

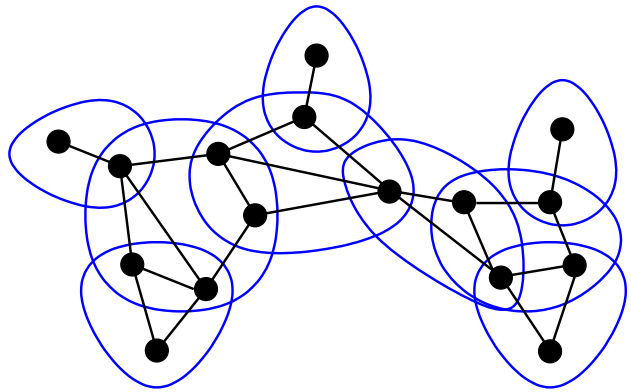
Décomposition arborescente



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

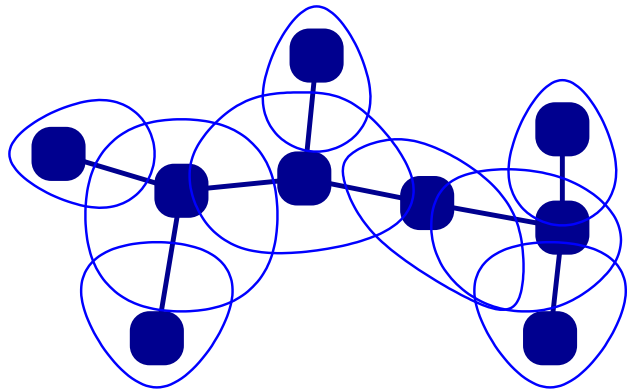
Décomposition arborescente



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

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

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

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

Décomposition arborescente

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

- La largeur de $\mathcal{T}_G = (T, \{X_t : t \in T\})$ est
$$width(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$
- La largeur arborescente d'un graphe G est
$$tw(G) = \min_{\mathcal{T}_G} width(\mathcal{T}_G)$$

Décomposition arborescente

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

- La largeur de $\mathcal{T}_G = (T, \{X_t : t \in T\})$ est
$$width(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$
- La largeur arborescente d'un graphe G est
$$tw(G) = \min_{\mathcal{T}_G} width(\mathcal{T}_G)$$

Observation :

1. tout nœud X_t de $\mathcal{T}_G$ est un séparateur de G.

Décomposition arborescente

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

- La largeur de $\mathcal{T}_G = (T, \{X_t : t \in T\})$ est
$$width(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$
- La largeur arborescente d'un graphe G est
$$tw(G) = \min_{\mathcal{T}_G} width(\mathcal{T}_G)$$

Observation :

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

Décomposition arborescente

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

- La largeur de $\mathcal{T}_G = (T, \{X_t : t \in T\})$ est
$$width(\mathcal{T}_G) = \max_{t \in T} |X_t| - 1$$
- La largeur arborescente d'un graphe G est
$$tw(G) = \min_{\mathcal{T}_G} width(\mathcal{T}_G)$$

Observation :

1. tout nœud X_t de $\mathcal{T}_G$ est un séparateur de G.
2. si X_t et $X_{t'}$ sont deux nœuds de $\mathcal{T}_G$ 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 :

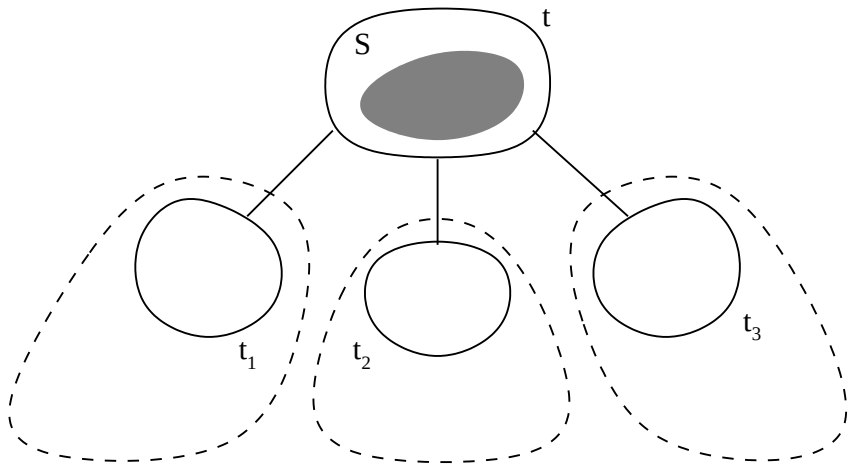
- V_t : l'ensemble de sommets présents dans les descendants de t
- G_t : le sous-graphe $G[V_t]$

Ensemble Indépendant (pondéré) paramétré $tw(G)$

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

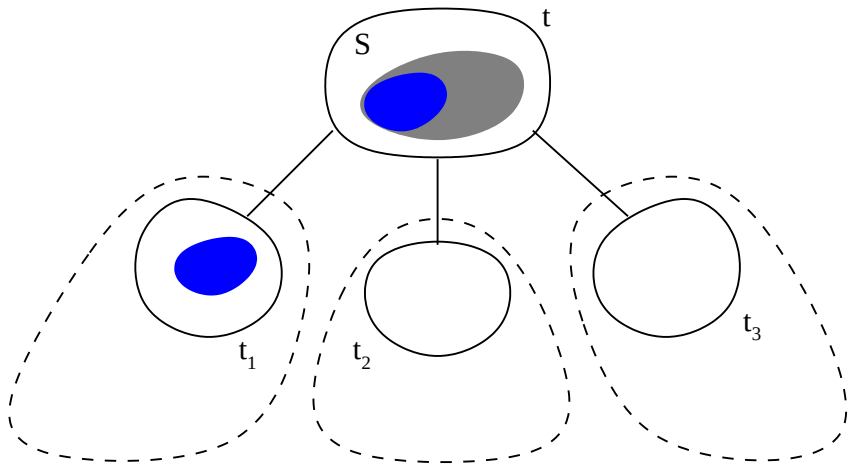
Ensemble Indépendant (pondéré) paramétré $tw(G)$

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



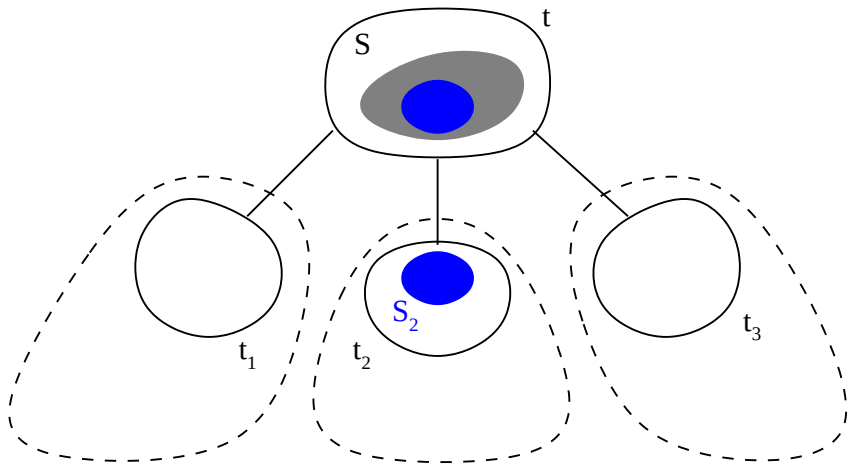
Ensemble Indépendant (pondéré) paramétré $tw(G)$

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



Ensemble Indépendant (pondéré) paramétré $tw(G)$

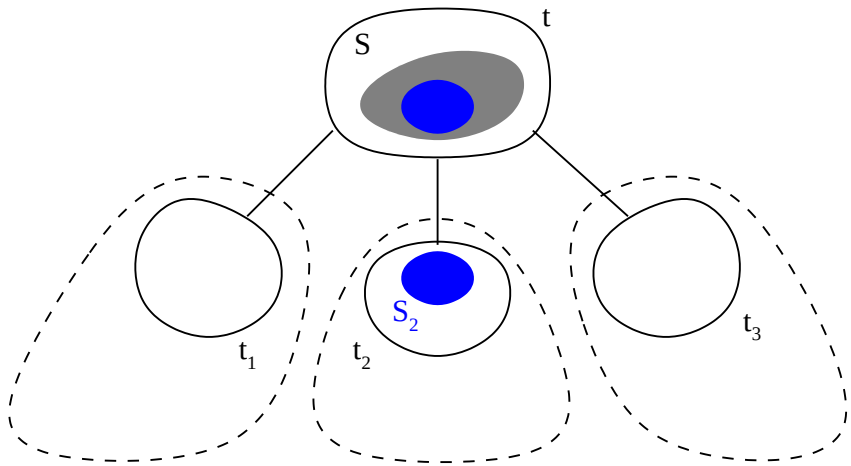
$\forall S \subseteq X_t$, $IS(S,t)$ = indép. max. de G_t tq $X_t \cap IS(S,t) = S$



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 $tw(G)$

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

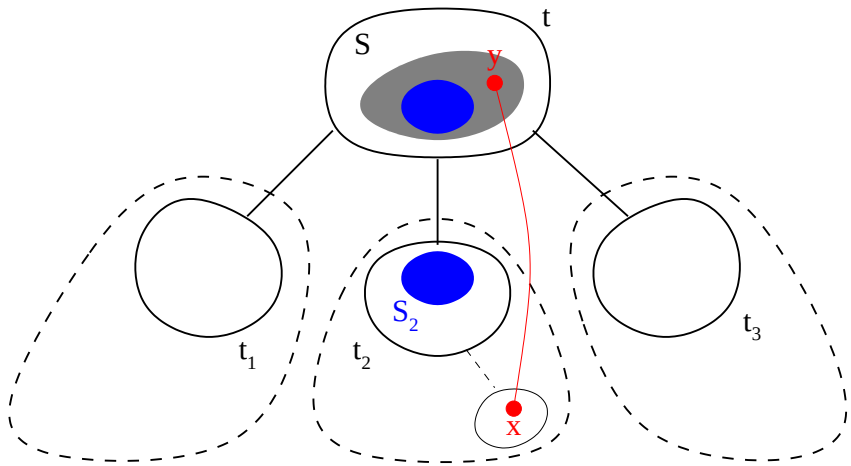


Lemme : Si $S \subseteq X_t$ et $S_j = S \cap X_{t_j}$ alors $IS(S,t) \cap V_{t_j} = IS(S_j, t_j)$

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

Ensemble Indépendant (pondéré) paramétré $tw(G)$

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



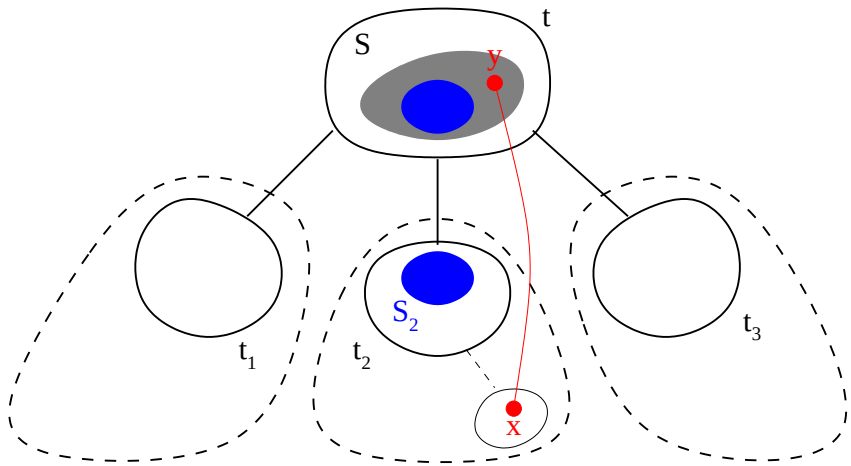
Lemme : Si $S \subseteq X_t$ et $S_j = S \cap X_{t_j}$ alors $IS(S,t) \cap V_{t_j} = IS(S_j, t_j)$

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

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

Ensemble Indépendant (pondéré) paramétré $tw(G)$

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



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

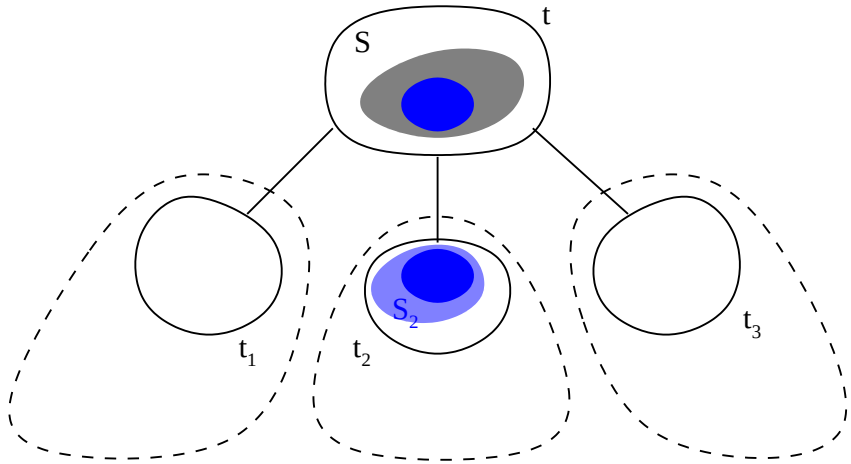
Hyp. : $IS(S,t) \cap V_{t_j}$ n'est pas un indépendant max. de G_{t_j}

$\Rightarrow$ $\exists y \in S \setminus S_j$ 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 $tw(G)$

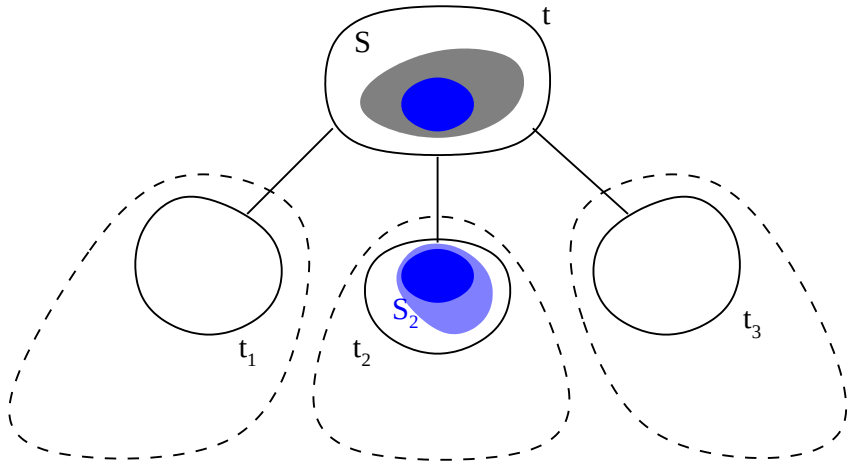
Idée de l'algorithme de programmation dynamique



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 $tw(G)$

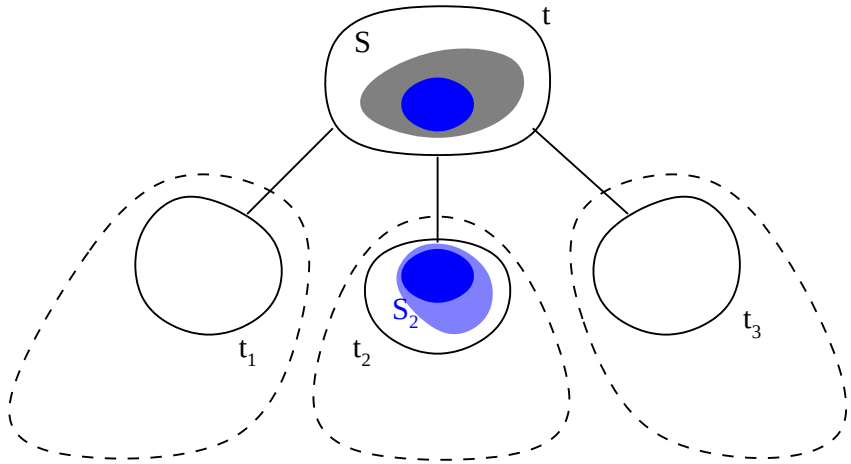
Idée de l'algorithme de programmation dynamique



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 $tw(G)$

Idée de l'algorithme de programmation dynamique

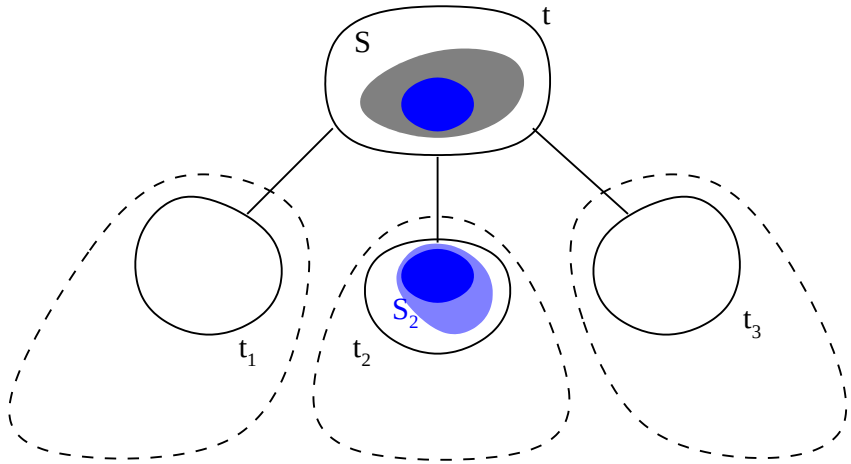


Calcul de $IS(S, t)$ connaissant $IS(S_j^i, t_j)$, $\forall j \in [l]$, $\forall S_j^i \subseteq X_{t_j}$

- 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 $tw(G)$

Idée de l'algorithme de programmation dynamique

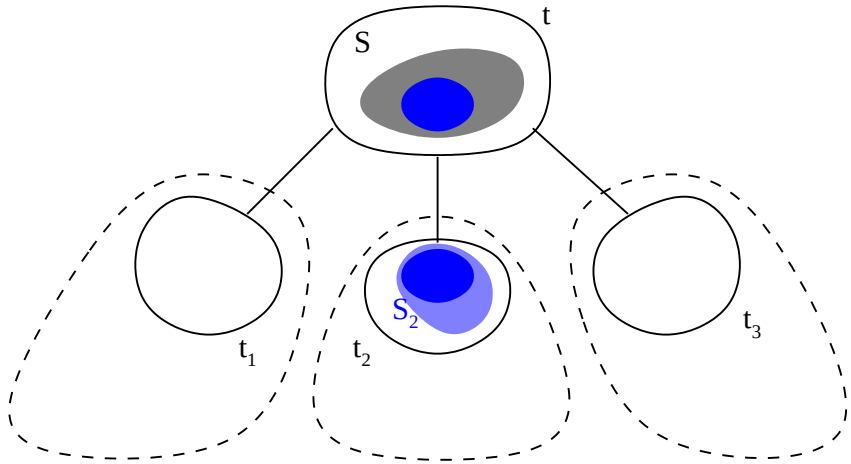


Calcul de $IS(S, t)$ connaissant $IS(S_j^i, t_j)$, $\forall j \in [l]$, $\forall S_j^i \subseteq X_{t_j}$

- 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 indépendent

Ensemble Indépendant (pondéré) paramétré $tw(G)$

Idée de l'algorithme 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 indépendent

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

Ensemble Indépendant (pondéré) paramétré $tw(G)$

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

Analyse de complexité :

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

Ensemble Indépendant (pondéré) paramétré $tw(G)$

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

Analyse de complexité :

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

Ensemble Indépendant (pondéré) paramétré $tw(G)$

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

Analyse de complexité :

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

Ensemble Indépendant (pondéré) paramétré $tw(G)$

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

Analyse de complexité :

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

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

Théorème de Bodlaender

Théoreme [Arnborg, Corneil, Proskurowski]
Etant donnés une graphe G et un entier k, décider si $tw(G) \leqslant k$ est un problème NP-difficile.

Théorème [Bodlaender]
Etant donnés une graphe G et un entier k fixé, il existe un algorithme de complexité $O(2^{k^3}.n)$ qui décide si $tw(G) \leqslant k$

Théorème de Bodlaender

Théoreme [Arnborg, Corneil, Proskurowski]
Etant donnés une graphe G et un entier k, décider si $tw(G) \leqslant k$ est un problème NP-difficile.

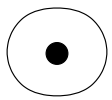
Théorème [Bodlaender]
Etant donnés une graphe G et un entier k fixé, il existe un algorithme de complexité $O(2^{k^3}.n)$ qui décide si $tw(G) \leqslant k$

Théorème [Diestel et al.]
Il existe un algorithme de complexité $O(3^{3k}.k.n)$ qui calcule une décomposition arborescente $\mathcal{T}$ tel que $width(\mathcal{T}) \leqslant 4k+1$ si $tw(G) \leqslant k$

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

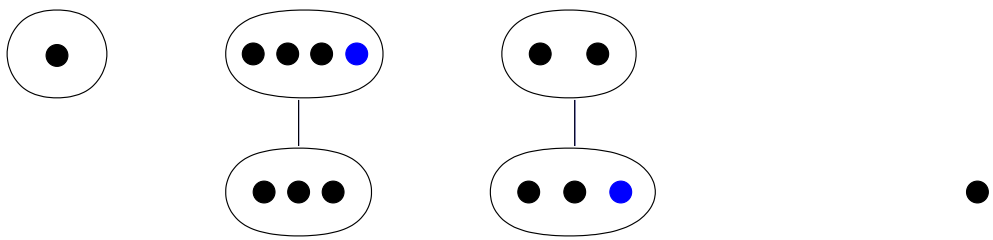
Une décomposition arborescente enracinée $\mathcal{T}_G = (T, \{X_t : t \in T\})$ est simple si tout nœud t est d'un des quatre types suivant :

- 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 :

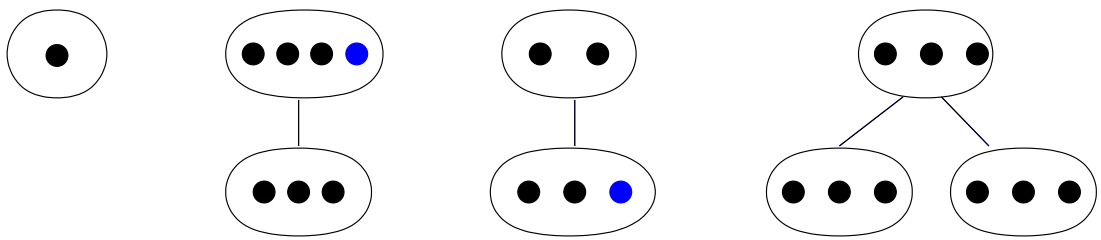
- Feuille : pas de fils et $|X_t| = 1$
- Ajout : un fils unique t' et $X_t = X_{t'} \cup \{v\}$ avec $v \notin X_{t'}$
- Suppression : un fils unique t' et $X_t = X_{t'} \setminus \{v\}$ avec $v \in X_{t'}$



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 :

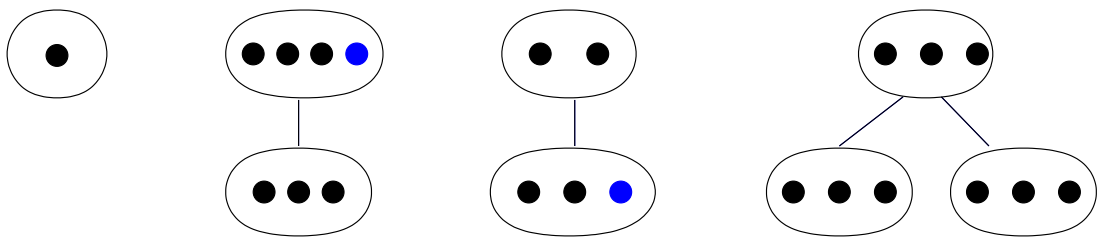
- Feuille : pas de fils et $|X_t| = 1$
- Ajout : un fils unique t' et $X_t = X_{t'} \cup \{v\}$ avec $v \notin X_{t'}$
- Suppression : un fils unique t' et $X_t = X_{t'} \setminus \{v\}$ avec $v \in X_{t'}$
- Fusion : deux fils t_1 et t_2 et $X_t = X_{t_1} = X_{t_2}$.



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 $X_t = X_{t_1} = X_{t_2}$.



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 $k^2.c$ 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 : $X_t = X_{t'} \cup \{v\}$

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

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 : $X_t = X_{t'} \cup \{v\}$

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

- 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 : $X_t = X_{t'} \cup \{v\}$

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

- 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 : $X_t = X_{t'} \cup \{v\}$

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

- 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 : $X_t = X_{t_1} = X_{t_2}$

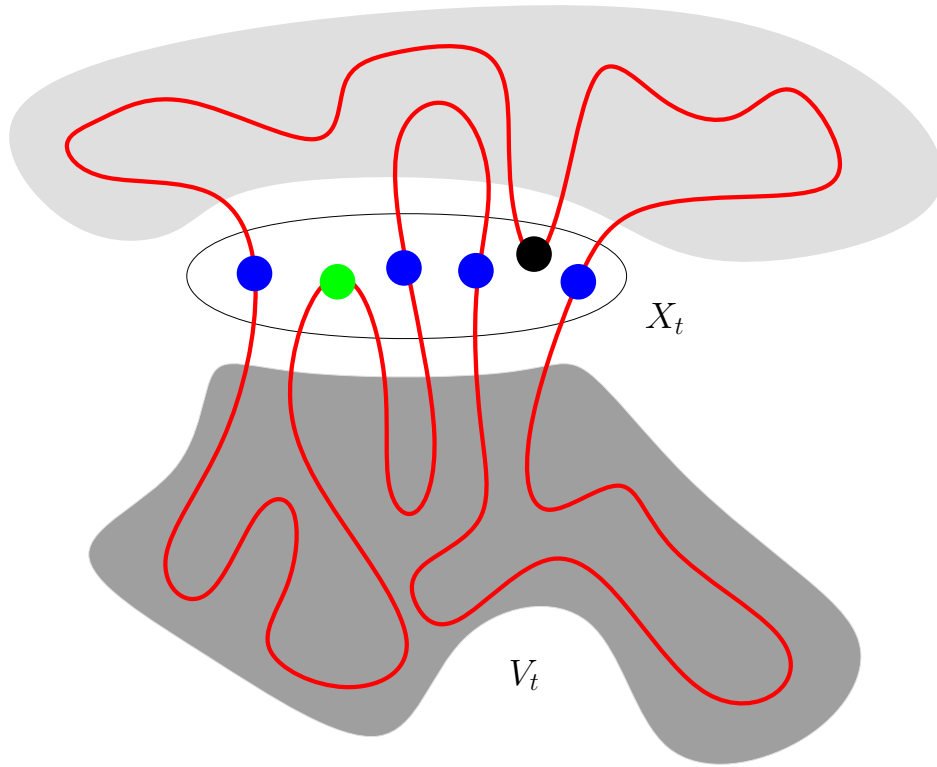
$$IS(S,t) = IS(S,t_1) + IS(S,t_2) - \omega(S)$$

Complexité : $O(2^k.n)$

Cycle Hamiltonien paramétré par $tw(G)$

Soit $\mathcal{C}$ un cycle hamiltonien.

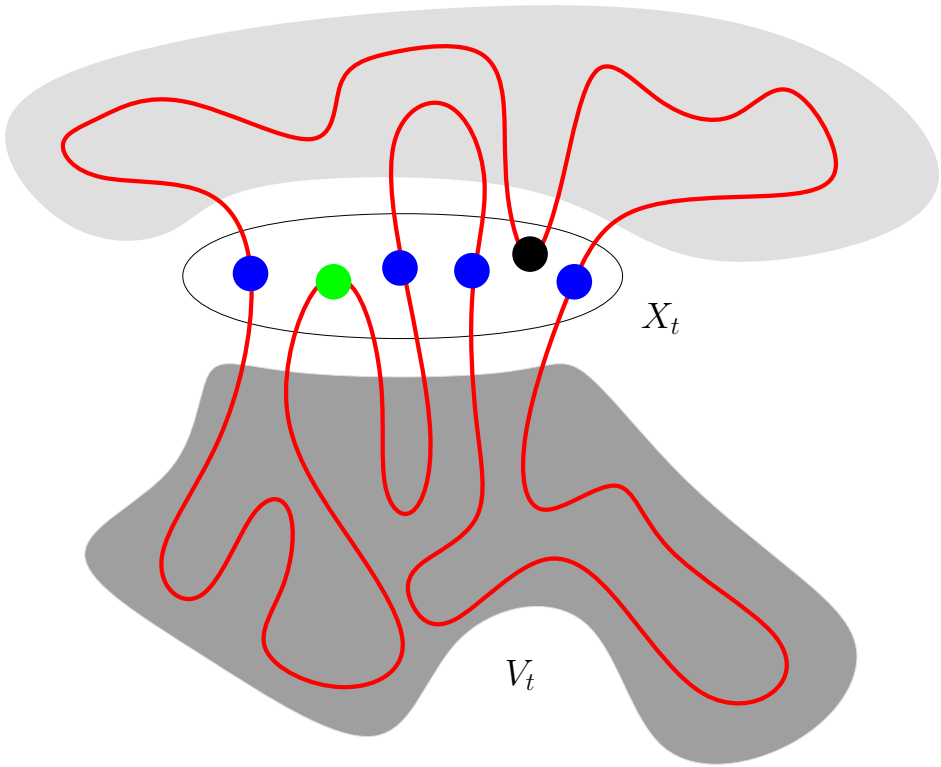
- $\mathcal{C} \cap G[V_t]$ est une collection de chemins



Cycle Hamiltonien paramétré par $tw(G)$

Soit $\mathcal{C}$ un cycle hamiltonien.

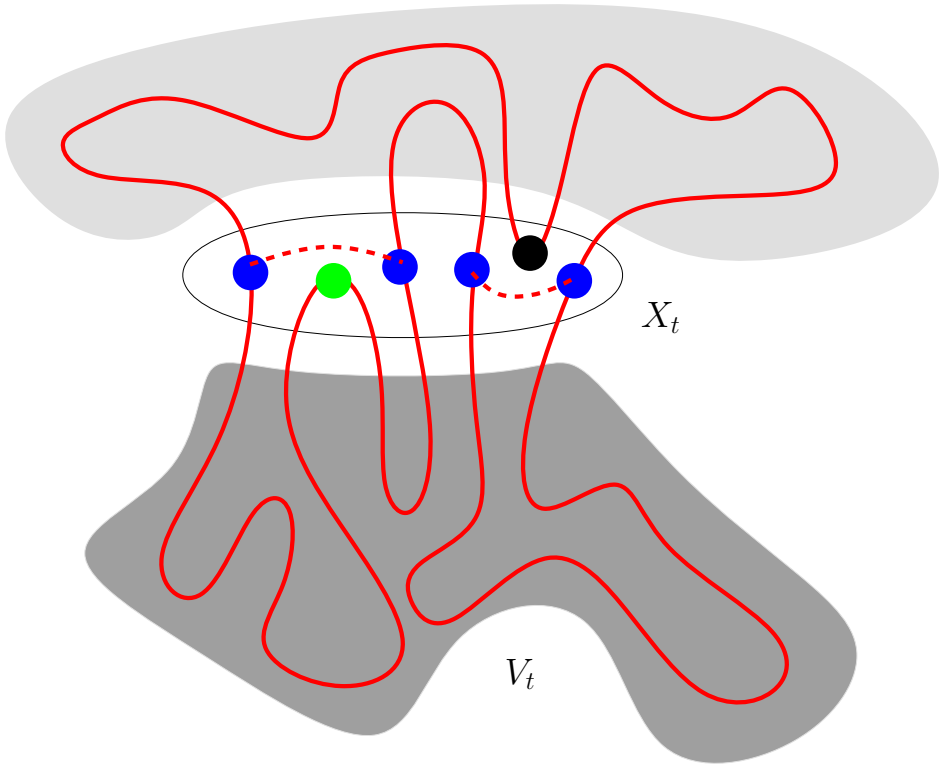
- $\mathcal{C} \cap G[V_t]$ est une collection de chemins
- Les sommets de X_t sont :
  - isolés : X_t^0
  - des extrémités : X_t^1
  - des sommets internes : X_t^2



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

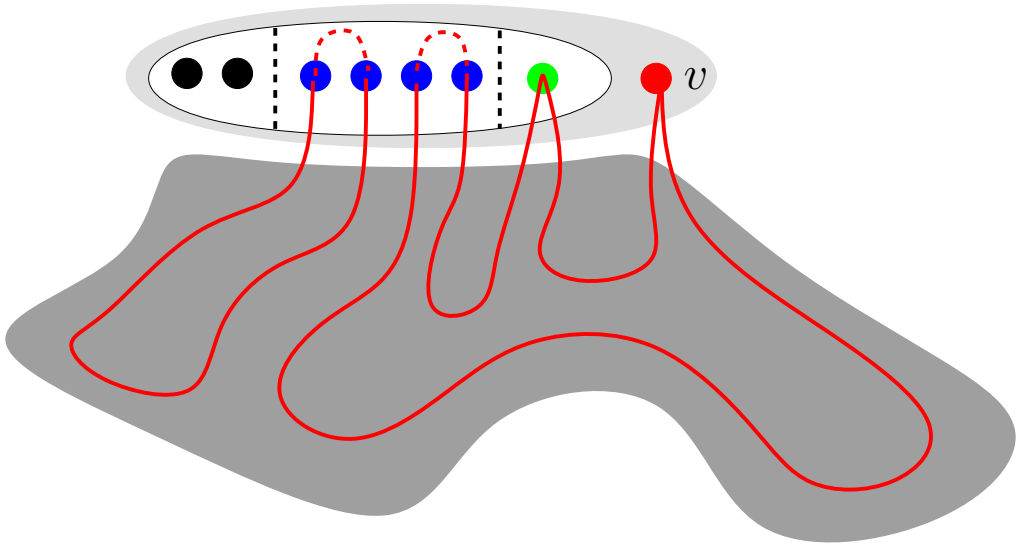
$$(X_t^0, X_t^1, X_t^2, M)$$

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

Preuve Résultat

Nœud Suppression

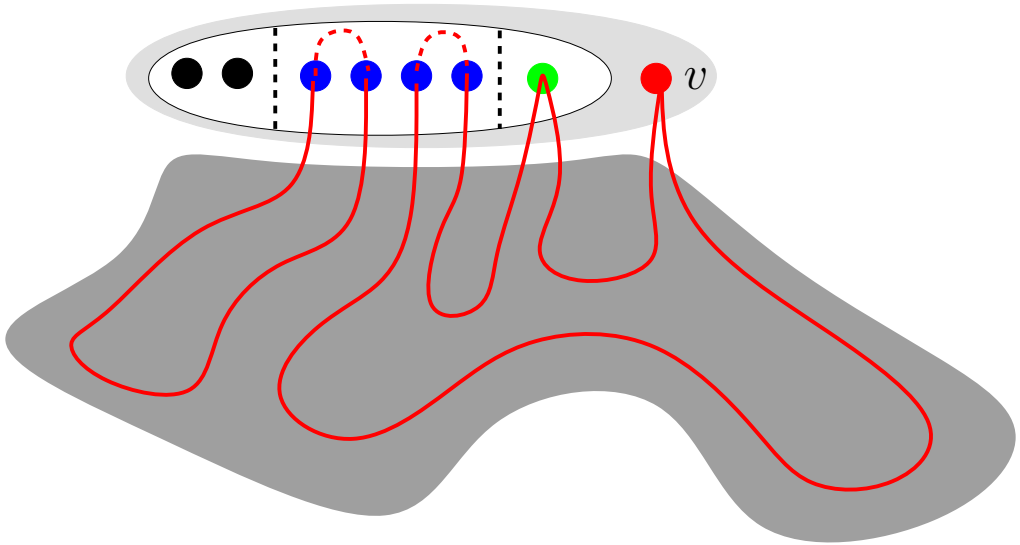
Soit t un nœud suppression et t' son fils tel que $X_t = X_{t'} \setminus \{v\}$



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

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

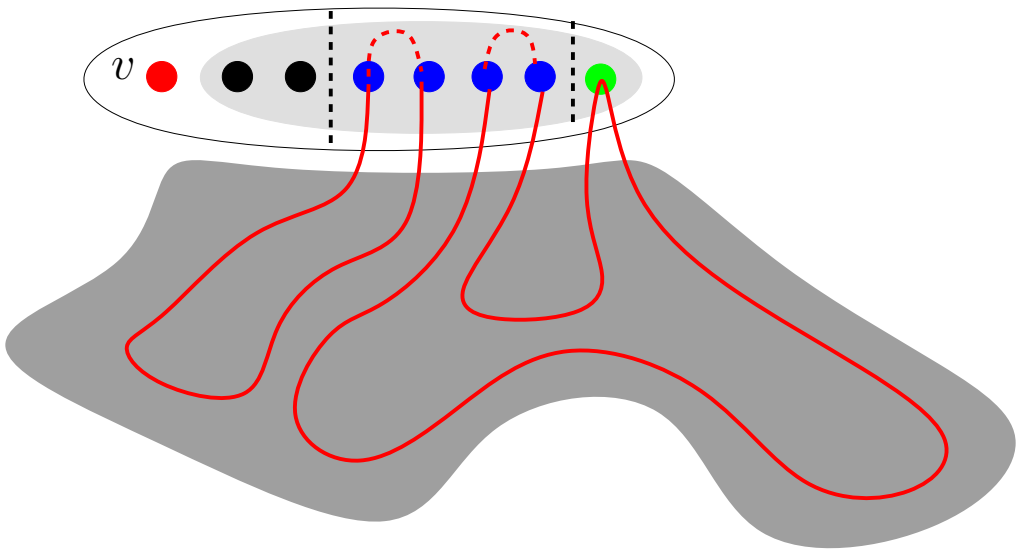
$\Leftrightarrow$

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

Nœud Ajout

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

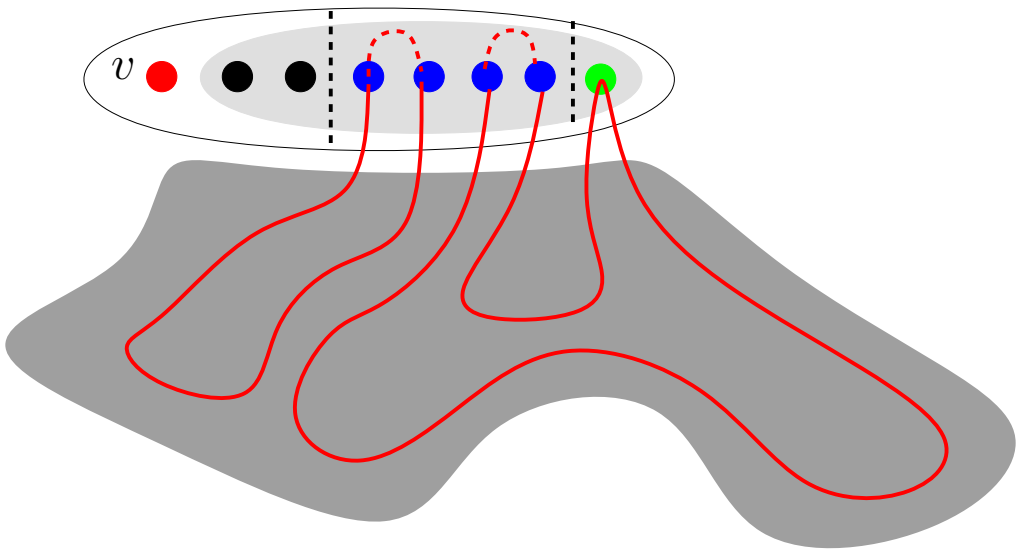
- Hyp. : $v \in X_t^0$



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

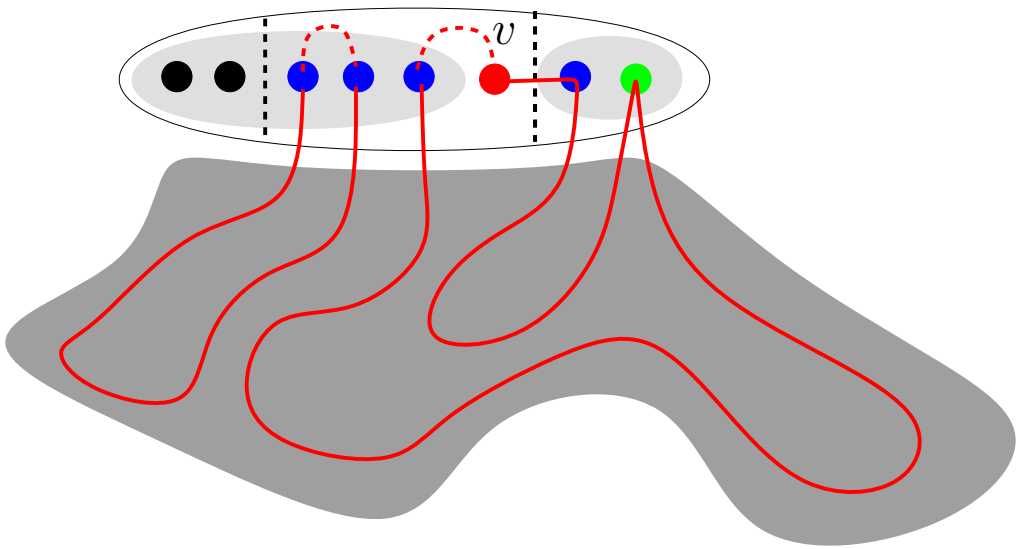
$\Leftrightarrow$

$(X_{t'}^0, X_{t'}^1, X_{t'}^2, M)$ 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. : $v \in X_t^1$

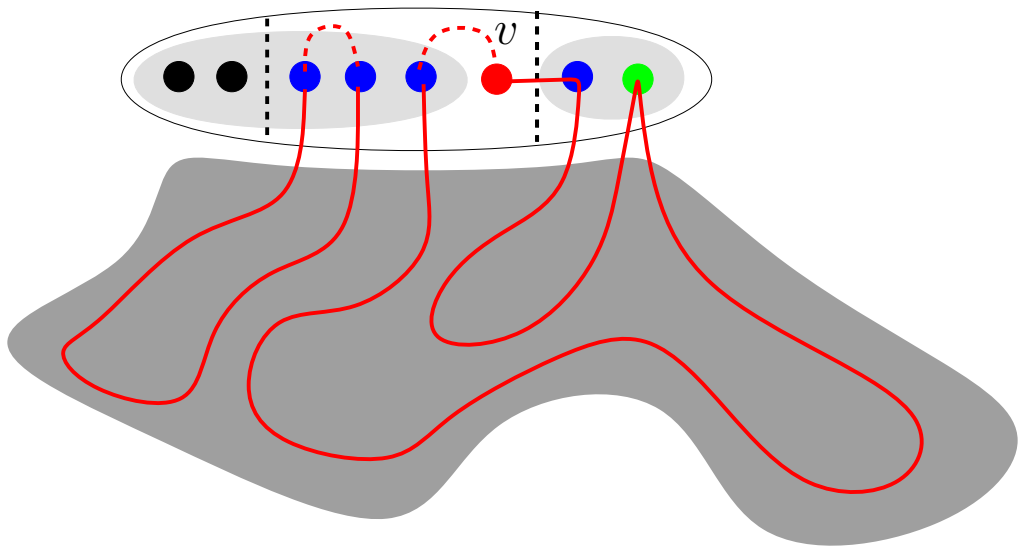


Obs. : $X_{t'}$ est un séparateur $\Rightarrow$ $N(v) \subset X_t$
donc un sommet u de $X_{t'}^1$ devient interne $\longrightarrow u \in X_t^2$

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 $\Rightarrow$ $N(v) \subset X_t$
donc un sommet u de $X_{t'}^1$ devient interne $\longrightarrow u \in X_t^2$

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

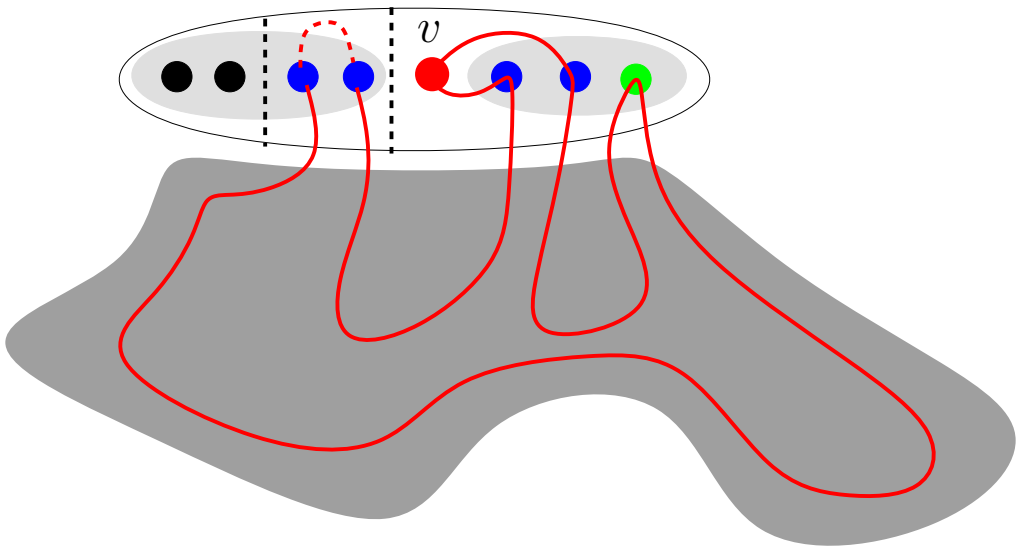
$\Leftrightarrow$

$(X_{t'}^0, X_{t'}^1, X_{t'}^2, M)$ 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. : $v \in X_t^2$

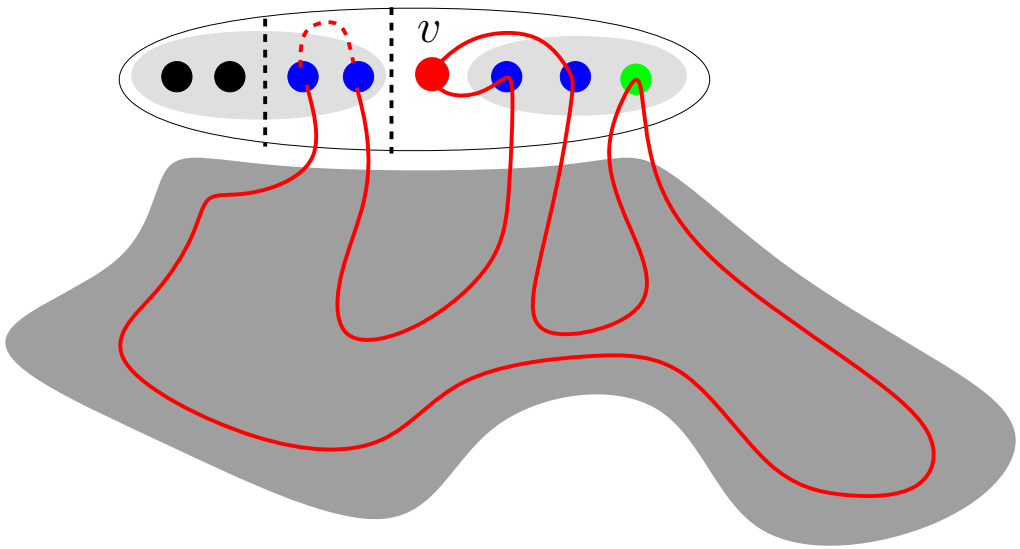


Obs. : $X_{t'}$ est un séparateur $\Rightarrow$ $N(v) \subset X_t$
donc deux sommets u, u' de $X_{t'}^1$ deviennent internes $\longrightarrow u, u' \in X_t^2$

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 $\Rightarrow$ $N(v) \subset X_t$
donc deux sommets u, u' de $X_{t'}^1$ deviennent internes $\longrightarrow u, u' \in X_t^2$

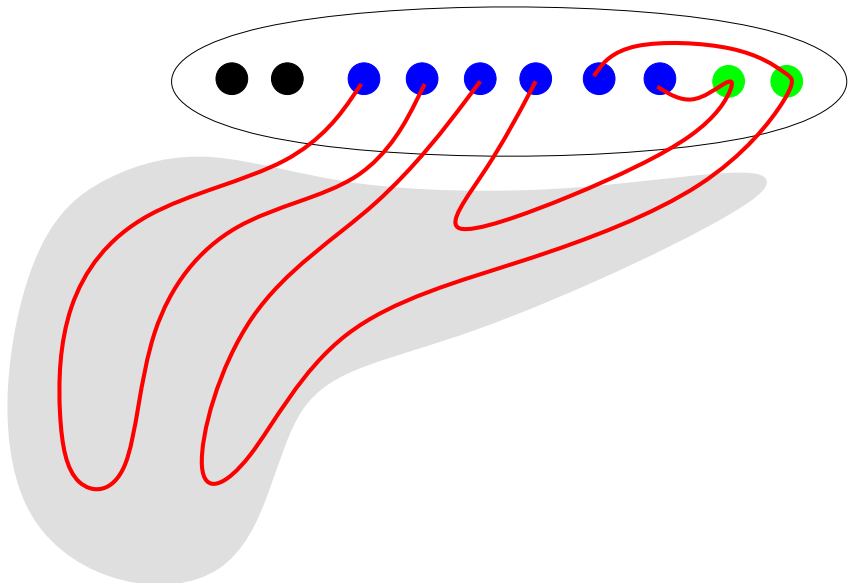
$(X_{t'}^0, X_{t'}^1 \setminus \{u, u'\}, X_{t'}^2 \cup \{v, u, u'\}, M')$ est une solution partielle pour t

$\Leftrightarrow$

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

Nœud Fusion

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

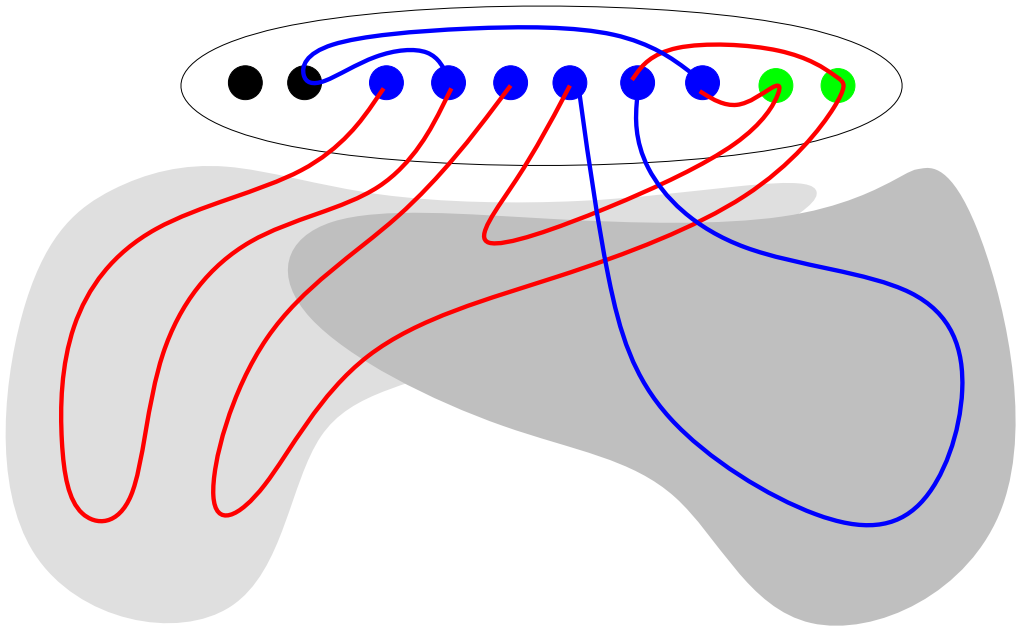


Obs. : pour être compatible, les solutions partielles doivent vérifier

- $X^2_{t_1} \subseteq X^0_{t_2}$ et $X^1_{t_1} \subseteq X^1_{t_2} \cup X^0_{t_2}$
- $X^2_{t_2} \subseteq X^0_{t_1}$ et $X^1_{t_2} \subseteq X^1_{t_1} \cup X^0_{t_1}$
- 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

- $X^2_{t_1} \subseteq X^0_{t_2}$ et $X^1_{t_1} \subseteq X^1_{t_2} \cup X^0_{t_2}$
- $X^2_{t_2} \subseteq X^0_{t_1}$ et $X^1_{t_2} \subseteq X^1_{t_1} \cup X^0_{t_1}$
- 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

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

- 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

Logique du second ordre monadique sur les graphes

On représente un graphe $G = (V, E)$ à l'aide de la structure $\mathcal{G} = (U, \mathit{Vertex}, \mathit{Edge}, I)$ 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 $\mathcal{G} = (U, \mathit{Vertex}, \mathit{Edge}, I)$ 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 $\mathit{adj}(u, v)$ et $\mathit{inc}(e, v)$
- 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 : Peut-on exprimer en MSOL qu'un graphe G

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

Largeur arborescente et logique du second ordre monadique

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).

Largeur arborescente et logique du second ordre monadique

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

Largeur arborescente et logique du second ordre monadique

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

Largeur arborescente et logique du second ordre monadique

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

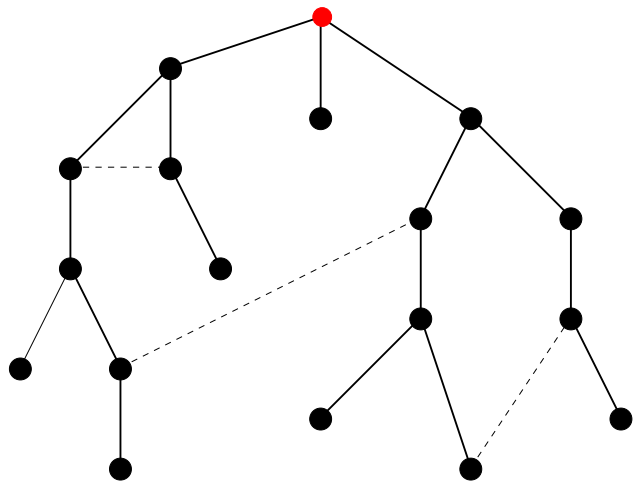
Utilisation du théorème de Courcelle

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 $tw(G)$)

Application du théorème de Courcelle

Exercice : Résoudre k-Longest-Path à l'aide du théorème de Courcelle

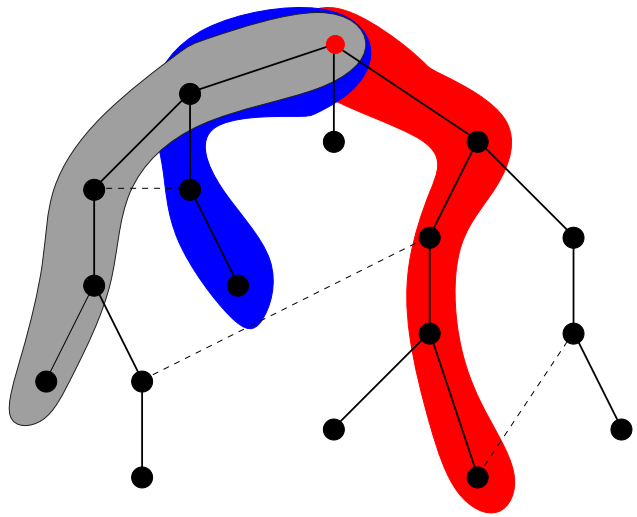
1. Calculer un arbre en profondeur T depuis un sommet quelconque x
2. Si T est de profondeur au moins k, il existe un chemin de longueur k depuis x



Application du théorème de Courcelle

Exercice : Résoudre k-Longest-Path à l'aide du théorème de Courcelle

1. Calculer un arbre en profondeur T depuis un sommet quelconque x
2. Si T est de profondeur au moins k, il existe un chemin de longueur k depuis x



3. Sinon $\mathbf{tw}(G) \leqslant k$, on peut utiliser le théorème de Courcelle (ou une programmation dynamique)

Un dernier exercice

- Est-ce que Vertex Cover paramétré par $\mathbf{tw}(G)$ admet un noyau polynomial ?

Merci