

Encerclement connexe dans les graphes

Nicolas Nisse

LRI, Orsay, Université Paris-Sud.

séminaire, INRIA Sophia Antipolis, 19 decembre 2006

Plan

- 1 Introduction
- 2 Stratégie d'encerclement
- 3 Connexité

Encerclement dans les graphes

But

Dans un réseau

- contaminé par un gaz toxique, un virus...,
- une équipe d'**agents** mobiles doit **nettoyer** le graphe.

Trouver une **stratégie** qui nettoie le graphe
en utilisant le moins de ressources possible.

Encerclement dans les graphes

But (Alternative)

Dans un réseau

- envahi par un **fugitif** invisible, omniscient et arbitrairement rapide,
- une équipe d'**agents** mobiles doit **capturer** le fugitif.

Trouver une **stratégie** qui capture le fugitif
en utilisant le moins d'agents possible.

Motivations

Applications

- sécurité dans les réseaux de type internet
- maintenance de réseaux de pipelines
- opération de secours dans des souterrains

Aspects fondamentaux

L'encerclement est en étroite relation avec des paramètres statiques des graphes.

- largeur arborescente
- largeur linéaire

Plan

- 1 Introduction
- 2 Stratégie d'encerclement**
 - Définitions et exemples
 - Complexité et monotonie
 - Décompositions de graphe
- 3 Connexité

Stratégies d' Encerclement, Parson. [GTC,78] Variante de Kirousis et Papadimitriou. [TCS,86]

Séquence de deux opérations élémentaires,...

- 1 Placer un agent sur un sommet du graphe ;
- 2 Supprimer un agent d'un sommet du graphe.

...qui doit aboutir à la capture du fugitif

Le fugitif est capturé lorsqu'il occupe (ou croise) un sommet occupé par un agent. Une arête est dite **nettoyée**, lorsque ses extrémités sont occupées par des agents.

Il faut minimiser le nombre d'agents.

Soit $s(G)$ le nombre minimum d'agents nécessaires pour capturer un fugitif invisible dans le graphe G .

Stratégies d' Encerclement, Parson. [GTC,78] Variante de Kirousis et Papadimitriou. [TCS,86]

Séquence de **deux** opérations élémentaires,...

- 1 **Placer** un agent sur un sommet du graphe ;
- 2 **Supprimer** un agent d'un sommet du graphe.

...qui doit aboutir à la capture du fugitif

Le fugitif est capturé lorsqu'il occupe (ou croise) un sommet occupé par un agent. Une arête est dite **nettoyée**, lorsque ses extrémités sont occupées par des agents.

Il faut minimiser le nombre d'agents.

Soit $s(G)$ le nombre minimum d'agents nécessaires pour capturer un fugitif invisible dans le graphe G .

Stratégies d' Encerclement, Parson. [GTC,78] Variante de Kirousis et Papadimitriou. [TCS,86]

Séquence de **deux** opérations élémentaires,...

- 1 **Placer** un agent sur un sommet du graphe ;
- 2 **Supprimer** un agent d'un sommet du graphe.

... qui doit aboutir à la capture du fugitif

Le fugitif est capturé lorsqu'il occupe (ou croise) un sommet occupé par un agent. Une arête est dite **nettoyée**, lorsque ses extrémités sont occupées par des agents.

Il faut minimiser le nombre d'agents.

Soit $s(G)$ le nombre minimum d'agents nécessaires pour capturer un fugitif invisible dans le graphe G .

Stratégies d' Encerclement, Parson. [GTC,78] Variante de Kirousis et Papadimitriou. [TCS,86]

Séquence de **deux** opérations élémentaires,...

- 1 **Placer** un agent sur un sommet du graphe ;
- 2 **Supprimer** un agent d'un sommet du graphe.

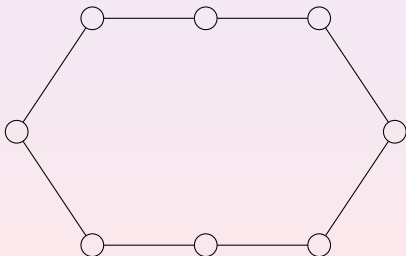
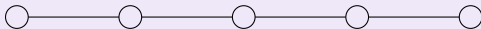
... qui doit aboutir à la capture du fugitif

Le fugitif est capturé lorsqu'il occupe (ou croise) un sommet occupé par un agent. Une arête est dite **nettoyée**, lorsque ses extrémités sont occupées par des agents.

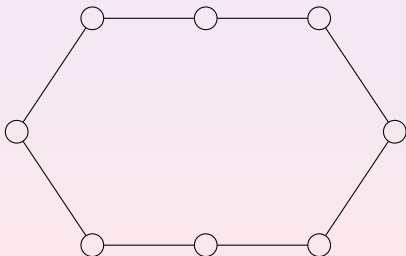
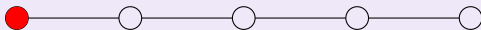
Il faut minimiser le nombre d'agents.

Soit $s(G)$ le nombre minimum d'agents nécessaires pour capturer un fugitif invisible dans le graphe G .

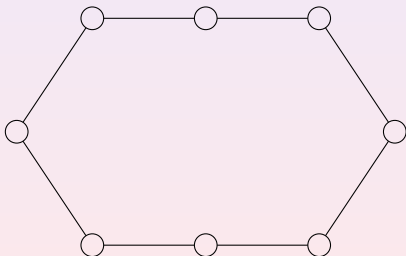
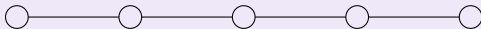
Exemples simples : le chemin et l'anneau



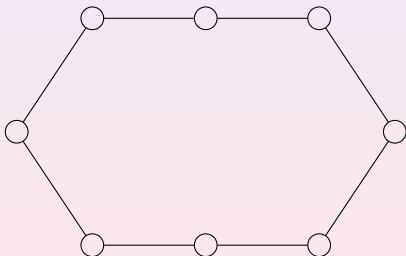
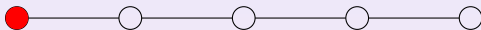
Exemples simples : le chemin et l'anneau



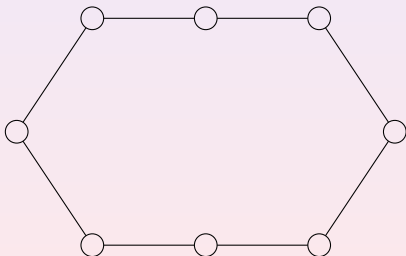
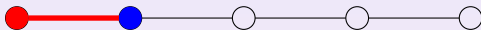
Exemples simples : le chemin et l'anneau



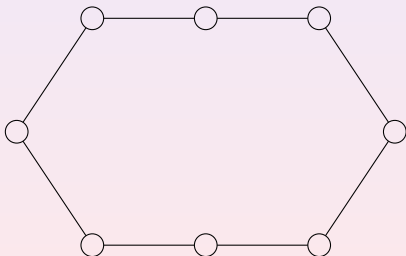
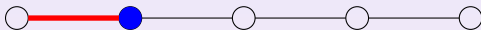
Exemples simples : le chemin et l'anneau



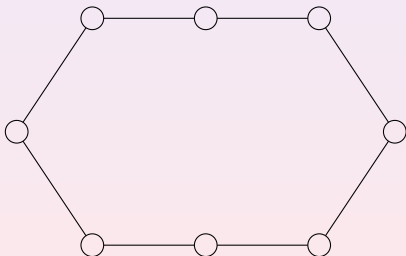
Exemples simples : le chemin et l'anneau



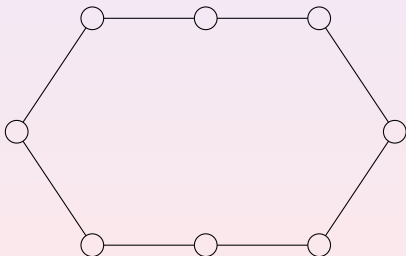
Exemples simples : le chemin et l'anneau



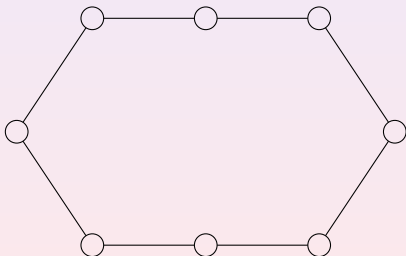
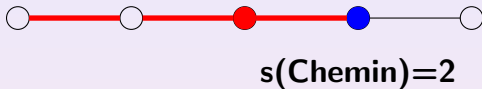
Exemples simples : le chemin et l'anneau



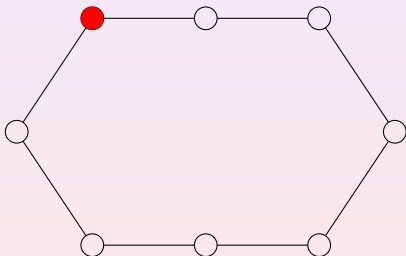
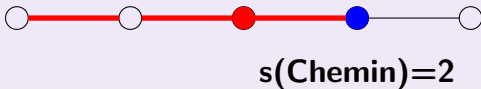
Exemples simples : le chemin et l'anneau



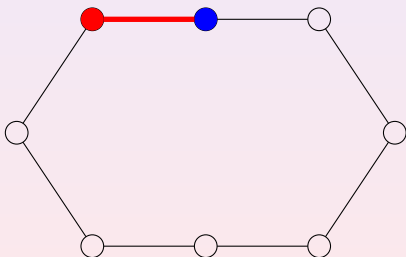
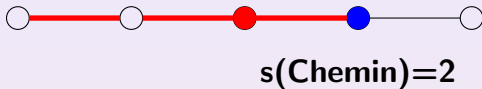
Exemples simples : le chemin et l'anneau



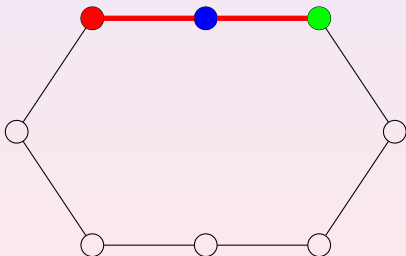
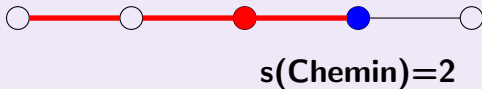
Exemples simples : le chemin et l'anneau



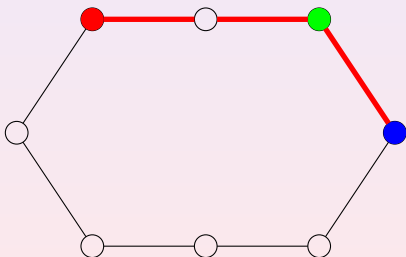
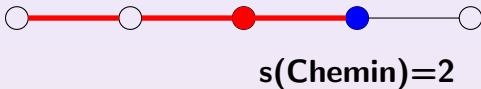
Exemples simples : le chemin et l'anneau



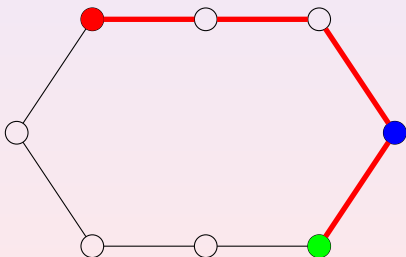
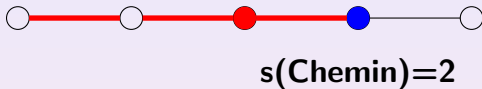
Exemples simples : le chemin et l'anneau



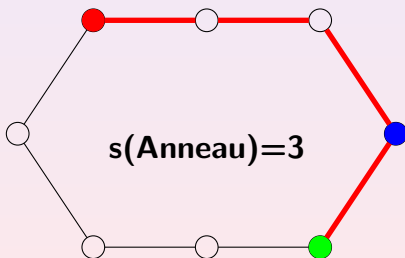
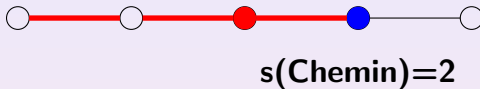
Exemples simples : le chemin et l'anneau



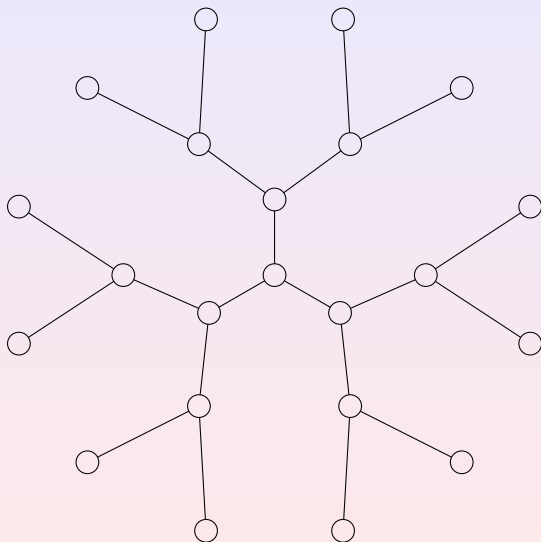
Exemples simples : le chemin et l'anneau



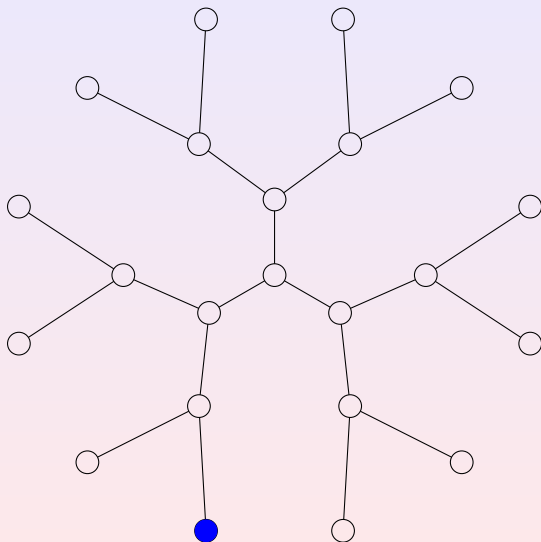
Exemples simples : le chemin et l'anneau



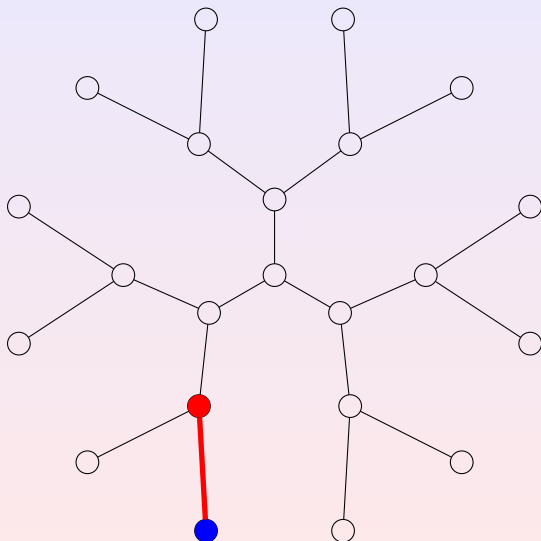
Encerclement dans un arbre



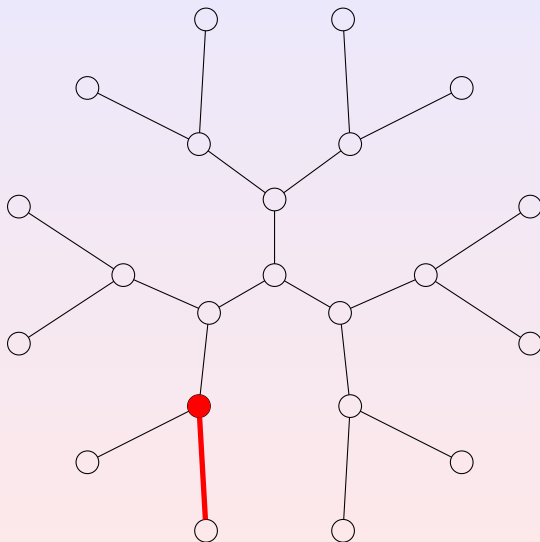
Encerclement dans un arbre



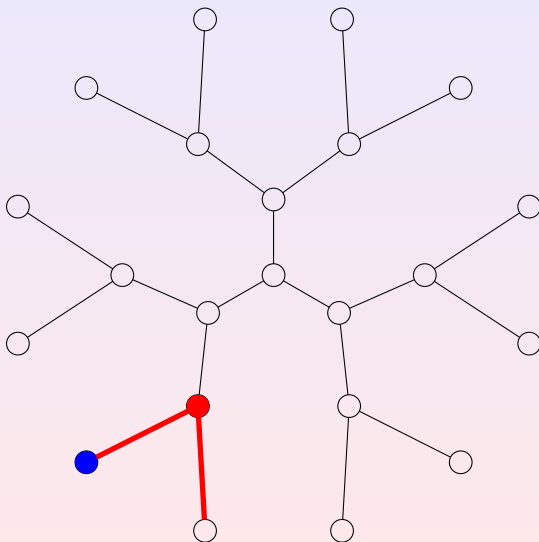
Encerclement dans un arbre



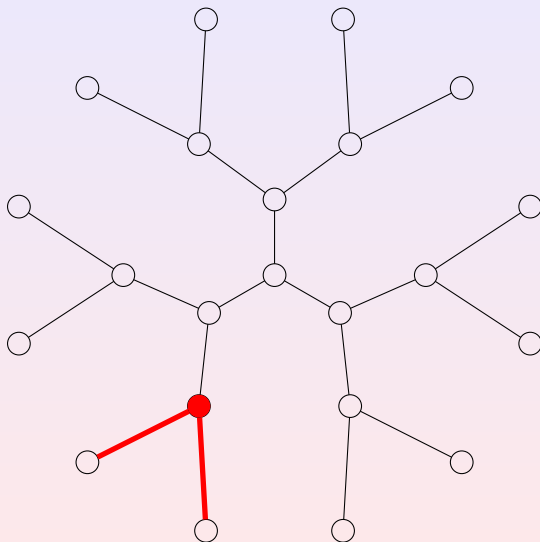
Encerclement dans un arbre



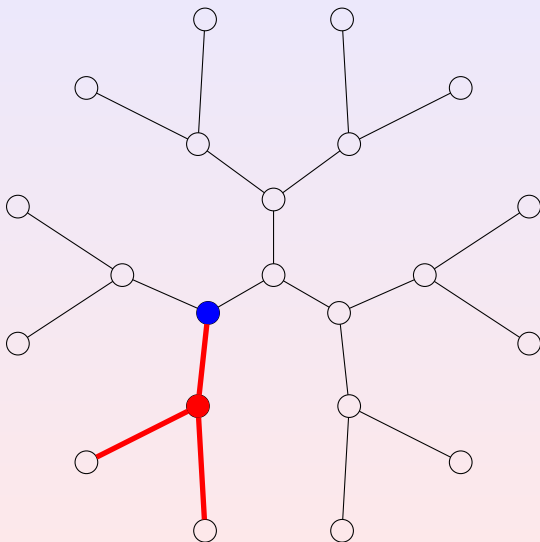
Encerclement dans un arbre



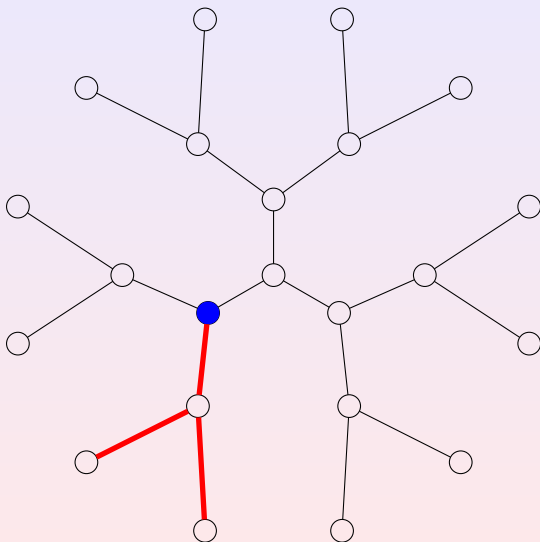
Encerclement dans un arbre



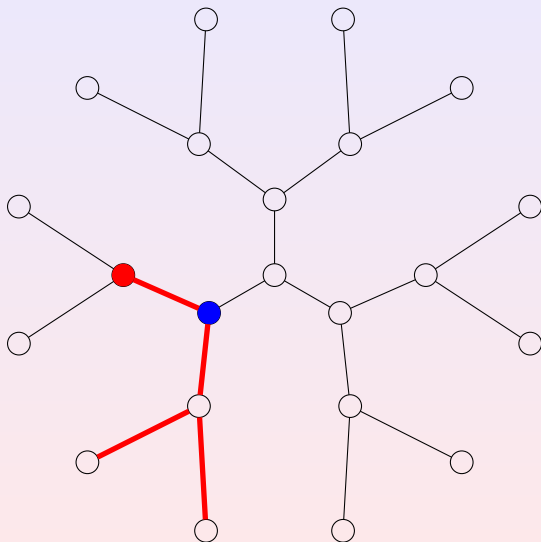
Encerclement dans un arbre



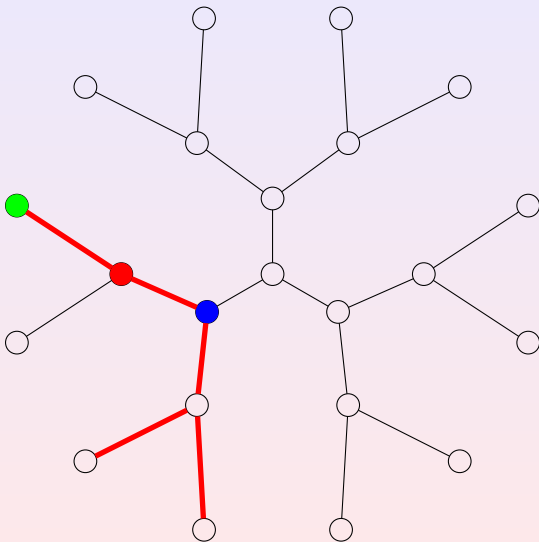
Encerclement dans un arbre



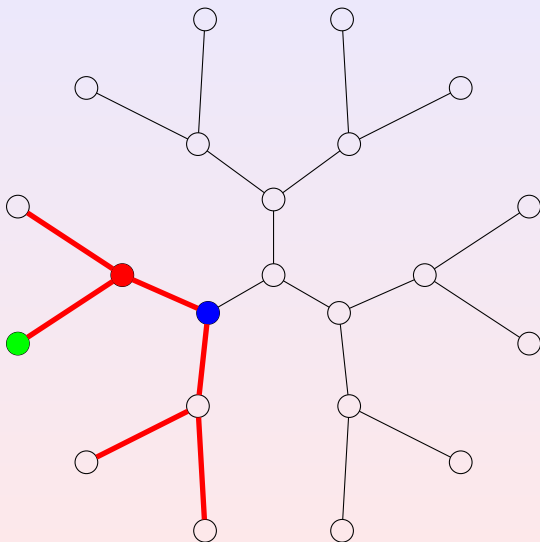
Encerclement dans un arbre



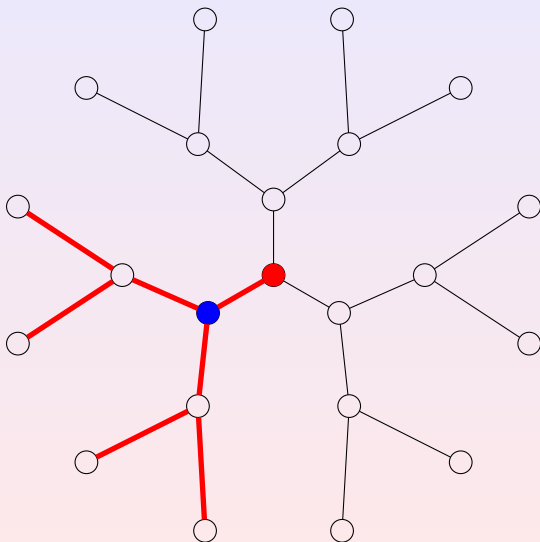
Encerclement dans un arbre



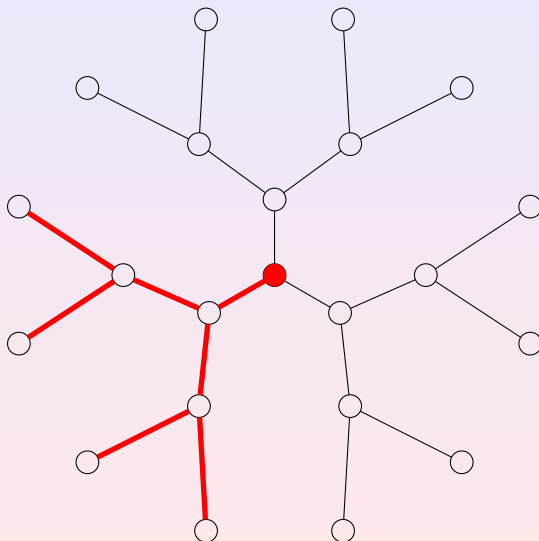
Encerclement dans un arbre



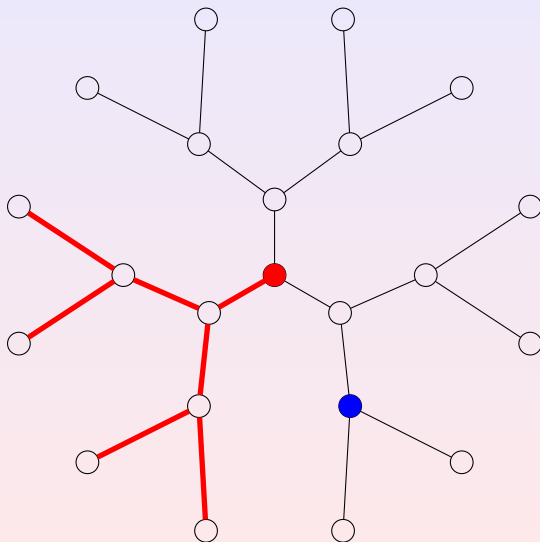
Encerclement dans un arbre



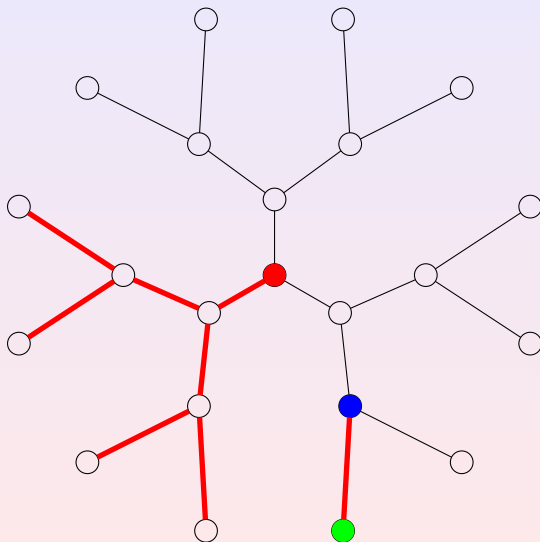
Encerclement dans un arbre



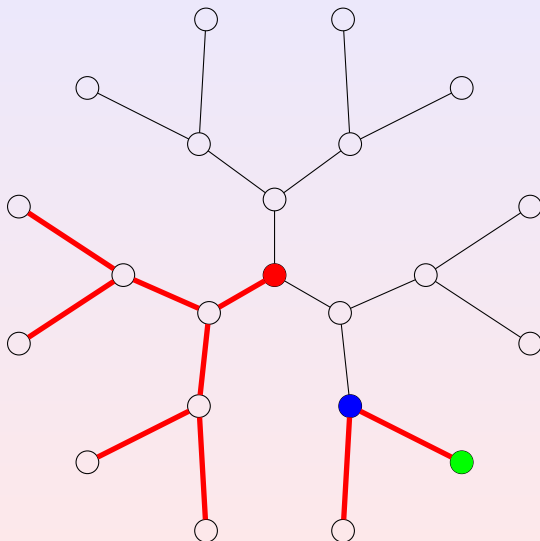
Encerclement dans un arbre



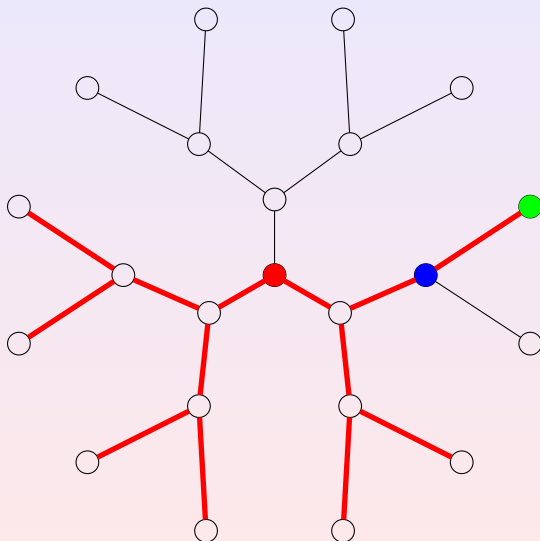
Encerclement dans un arbre



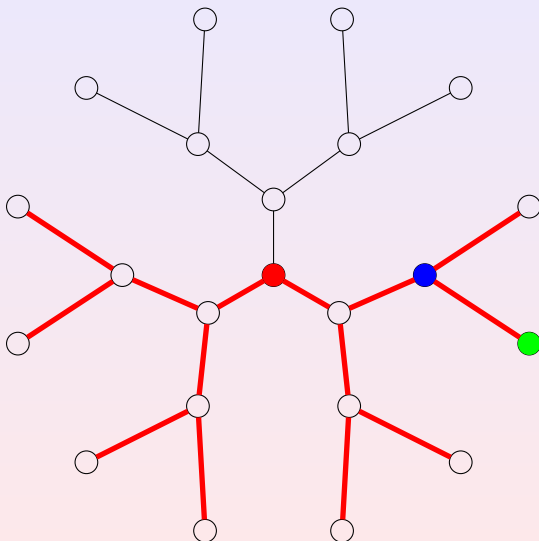
Encerclement dans un arbre



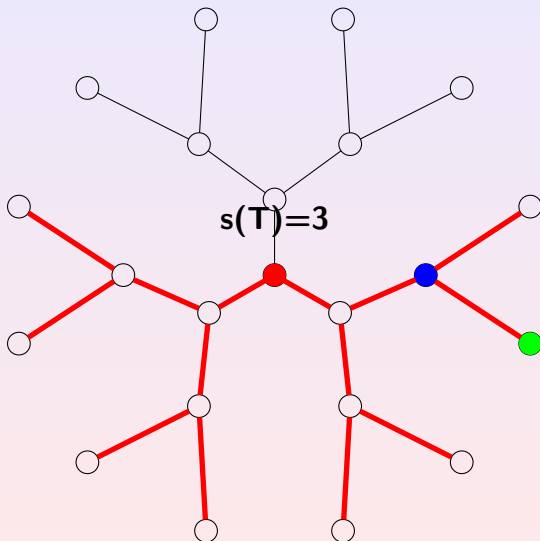
Encerclement dans un arbre



Encerclement dans un arbre



Encerclement dans un arbre



Encerclement visible

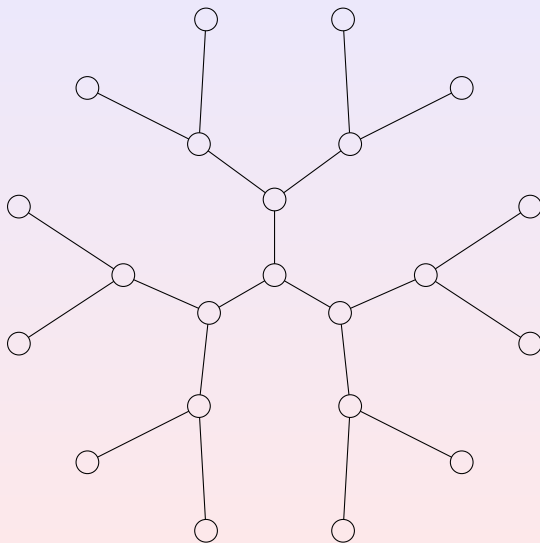
Visibilité du fugitif

Le fugitif est **visible** si, à chaque étape, les agents connaissent sa position (en fait la composante connexe où il se trouve). La stratégie peut donc être orientée d'après la position du fugitif.

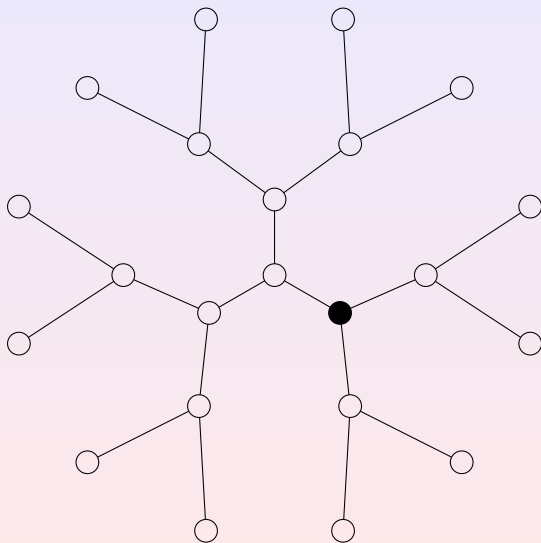
Paramètre associé

Soit $vs(G)$ l'encerclement d'un fugitif visible dans le graphe G . De manière évidente, $vs(G) \leq s(G)$.

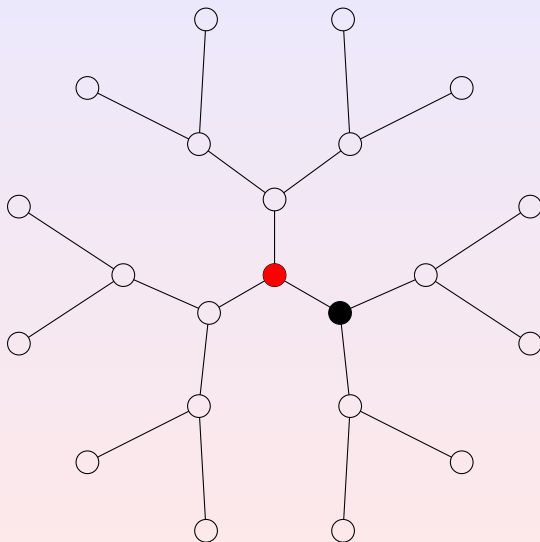
Encerclement d'un fugitif visible dans un arbre



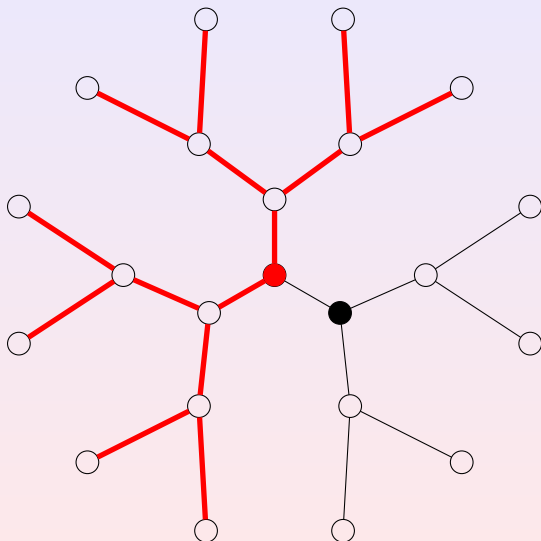
Encerclement d'un fugitif visible dans un arbre



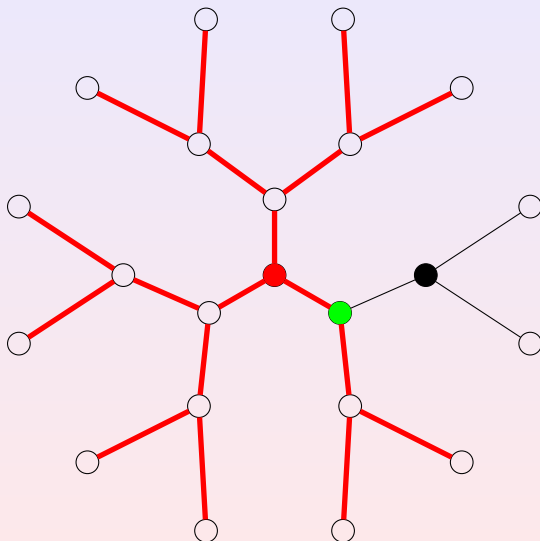
Encerclement d'un fugitif visible dans un arbre



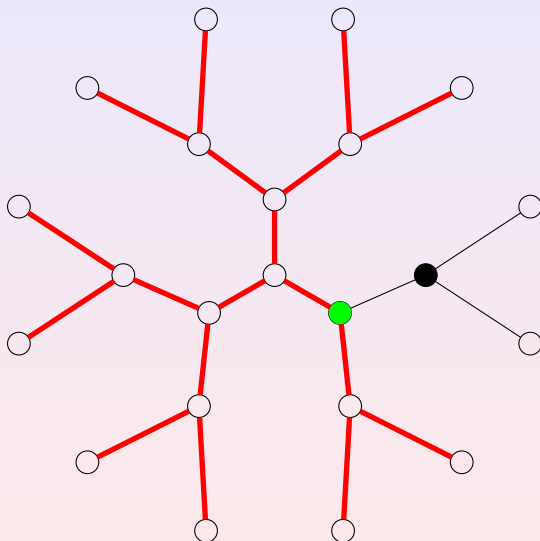
Encerclement d'un fugitif visible dans un arbre



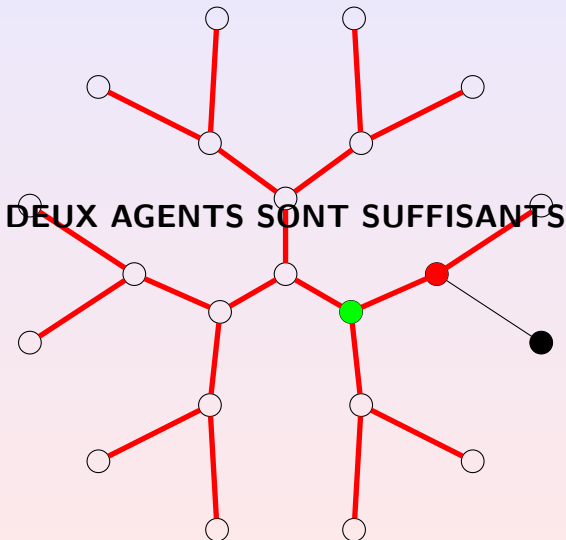
Encerclement d'un fugitif visible dans un arbre



Encerclement d'un fugitif visible dans un arbre



Encerclement d'un fugitif visible dans un arbre



Complexité du problème

Déterminer $s(G)$ est **NP-difficile**

- **Megiddo et al**, J.of ACM, 1988
The complexity of searching a graph.

Déterminer $vs(G)$ est **NP-difficile**

- **Seymour et Thomas**, J. of Comb. Th., 1993.
Graph searching and a min-max theorem for tree-width

Remarque :

Ils sont linéaires dans le cas des arbres.

Question :

Ces problèmes sont-ils NP-complet ?

Complexité du problème

Déterminer $s(G)$ est **NP-difficile**

- **Megiddo et al**, J.of ACM, 1988
The complexity of searching a graph.

Déterminer $vs(G)$ est **NP-difficile**

- **Seymour et Thomas**, J. of Comb. Th., 1993.
Graph searching and a min-max theorem for tree-width

Remarque :

Ils sont linéaires dans le cas des arbres.

Question :

Ces problèmes sont-ils NP-complet ?

Complexité du problème

Déterminer $s(G)$ est **NP-difficile**

- **Megiddo et al**, J.of ACM, 1988
The complexity of searching a graph.

Déterminer $vs(G)$ est **NP-difficile**

- **Seymour et Thomas**, J. of Comb. Th., 1993.
Graph searching and a min-max theorem for tree-width

Remarque :

Ils sont linéaires dans le cas des arbres.

Question :

Ces problèmes sont-ils NP-complet ?

Monotonie et Recontamination

Recontamination

Un sommet est **recontaminé** au cours d'une stratégie S , si il a été nettoyé (occupé par un agent) à une étape, et que le fugitif a la possibilité d'occuper ce sommet lors d'une étape ultérieure.

Monotonie

Une stratégie est **monotone** si aucune recontamination n'est autorisée. Un sommet nettoyé reste propre jusqu'à la fin. Soit $ms(G)$ l'encerclement monotone du graphe G .

Monotonie et Recontamination

Question : La recontamination aide-t'elle ?

Est-il plus difficile de nettoyer un graphe de manière monotone ? $ms(G) > s(G)$?

La recontamination n'aide pas : $s(G) = ms(G)$

- **Bienstock et Seymour**, J.of Alg., 1991
Monotonicity in graph searching.
- **LaPaugh**, J.of ACM, 1993
Recontamination does not help to search a graph.

$vs(G) = ms(G)$ (Cas d'un fugitif visible)

- **Seymour et Thomas**, J. of Comb. Th., 1993.
Graph searching and a min-max theorem for tree-width

Monotonie et Recontamination

Question : La recontamination aide-t'elle ?

Est-il plus difficile de nettoyer un graphe de manière monotone ? $ms(G) > s(G)$?

La recontamination n'aide pas : $s(G) = ms(G)$

- **Bienstock et Seymour**, J.of Alg., 1991
Monotonicity in graph searching.
- **LaPaugh**, J.of ACM, 1993
Recontamination does not help to search a graph.

$vs(G) = mvs(G)$ (Cas d'un fugitif visible)

- **Seymour et Thomas**, J. of Comb. Th., 1993.
Graph searching and a min-max theorem for tree-width

La recontamination n'aide pas : conséquences

En d'autres termes...

Il existe toujours une stratégie S qui capture un fugitif invisible en utilisant le moins d'agents possible et telle que S est **monotone**.

Déterminer $s(G)$ (resp. $vs(G)$) est **NP-complet**.

Une stratégie monotone est un certificat de taille polynomiale qui peut donc être vérifié en temps polynomial.

Pourquoi on est content que la recontamination n'aide pas

Il est (très) difficile de concevoir une stratégie non monotone. Maintenant, on peut se restreindre aux stratégies monotones.

La recontamination n'aide pas : conséquences

En d'autres termes...

Il existe toujours une stratégie S qui capture un fugitif invisible en utilisant le moins d'agents possible et telle que S est **monotone**.

Déterminer $s(G)$ (resp. $vs(G)$) est **NP-complet**

Une stratégie monotone est un certificat de taille polynomiale qui peut donc être vérifié en temps polynomial.

Pourquoi on est content que la recontamination n'aide pas

Il est (très) difficile de concevoir une stratégie non monotone.
Maintenant, on peut se restreindre aux stratégies monotones.

La recontamination n'aide pas : conséquences

En d'autres termes...

Il existe toujours une stratégie S qui capture un fugitif invisible en utilisant le moins d'agents possible et telle que S est **monotone**.

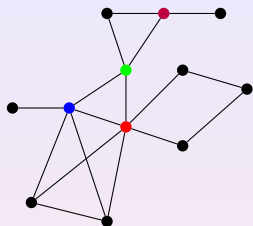
Déterminer $s(G)$ (resp. $vs(G)$) est **NP-complet**

Une stratégie monotone est un certificat de taille polynomiale qui peut donc être vérifié en temps polynomial.

Pourquoi on est content que la recontamination n'aide pas

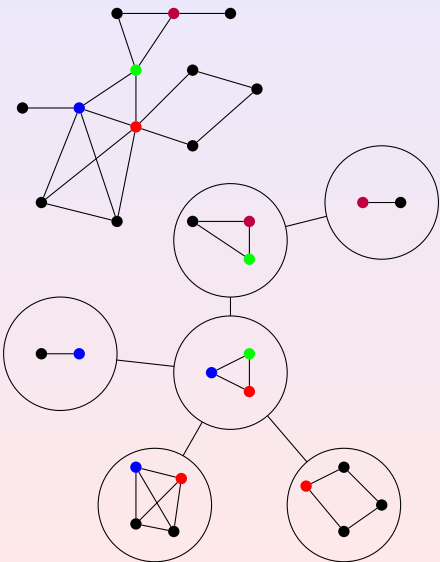
Il est (très) difficile de concevoir une stratégie non monotone. Maintenant, on peut se restreindre aux stratégies monotones.

Décomposition arborescente et linéaire

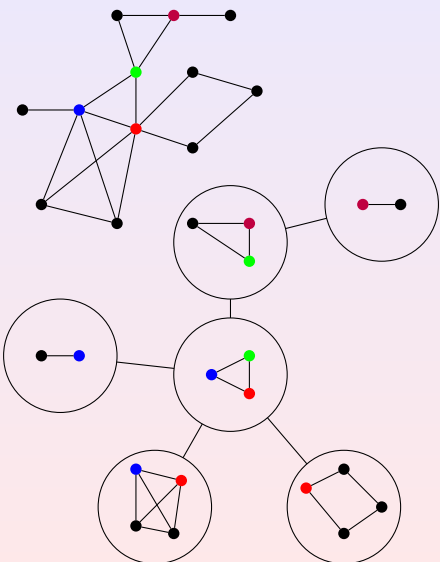


Décomposition arborescente et linéaire

arbre T , des sacs $(X_t)_{t \in V(T)}$



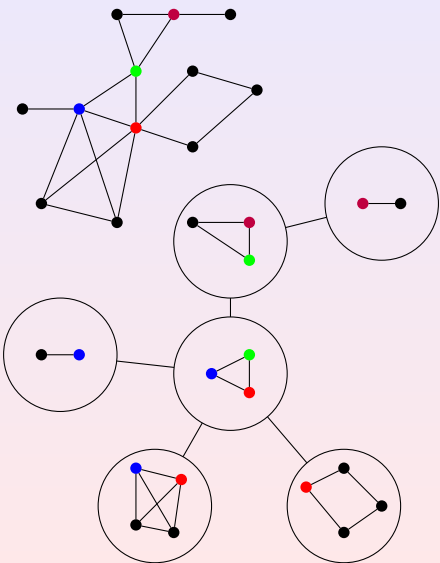
Décomposition arborescente et linéaire



arbre T , des sacs $(X_t)_{t \in V(T)}$

- chaque sommet de G est dans au moins un sac ;

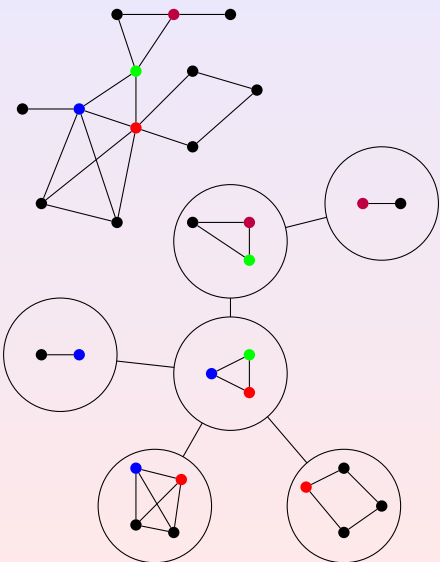
Décomposition arborescente et linéaire



arbre T , des sacs $(X_t)_{t \in V(T)}$

- chaque **sommet** de G est dans au moins un sac ;
- les 2 extrémités d'une arête de G sont dans au moins un sac ;

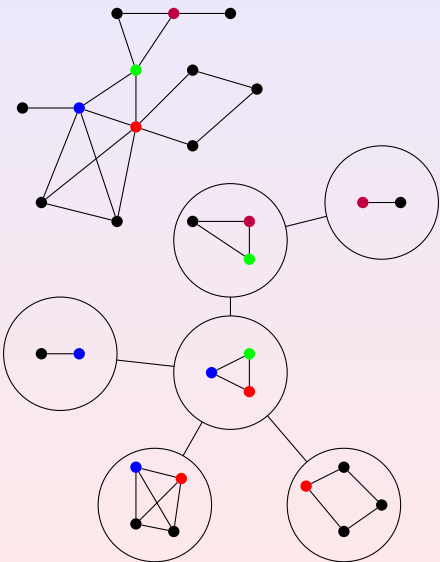
Décomposition arborescente et linéaire



arbre T , des sacs $(X_t)_{t \in V(T)}$

- chaque **sommet** de G est dans au moins un sac ;
- les 2 extrémités d'une **arête** de G sont dans au moins un sac ;
- pour tout sommet de G , les sacs qui le contiennent forment un sous-arbre.

Décomposition arborescente et linéaire

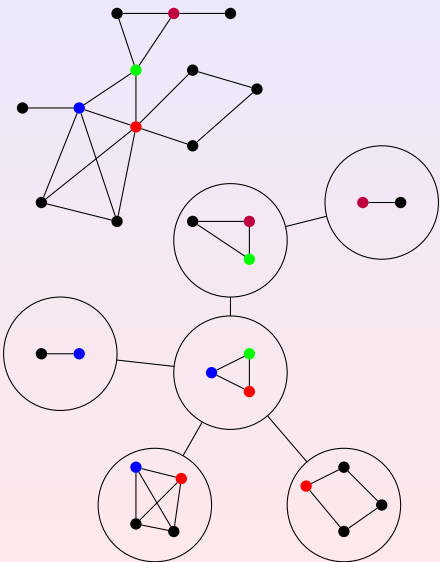


arbre T , des sacs $(X_t)_{t \in V(T)}$

- chaque **sommet** de G est dans au moins un sac ;
- les 2 extrémités d'une **arête** de G sont dans au moins un sac ;
- pour tout sommet de G , les sacs qui le contiennent forment un **sous-arbre**.

Largeur = |plus grand sac| - 1

Décomposition arborescente et linéaire



arbre T , des sacs $(X_t)_{t \in V(T)}$

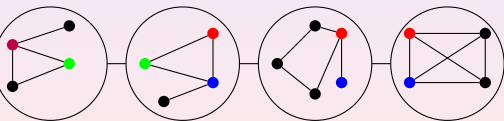
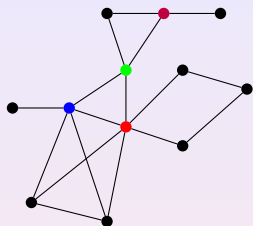
- chaque **sommet** de G est dans au moins un sac ;
- les 2 extrémités d'une **arête** de G sont dans au moins un sac ;
- pour tout sommet de G , les sacs qui le contiennent forment un **sous-arbre**.

Largeur = |plus grand sac| - 1

treewidth de G

tw(G), largeur minimum
parmi les décompositions
arborescentes

Décomposition arborescente et linéaire



chemin P , des sacs $(X_t)_{t \in V(T)}$

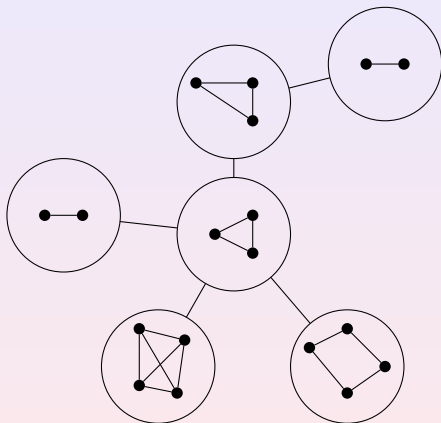
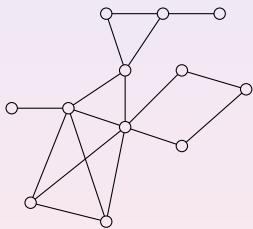
- chaque **sommet** de G est dans au moins un sac ;
- les 2 extrémités d'une **arête** de G sont dans au moins un sac ;
- pour tout sommet de G , les sacs qui le contiennent forment un **sous-chemin**.

Largeur = |plus grand sac| - 1

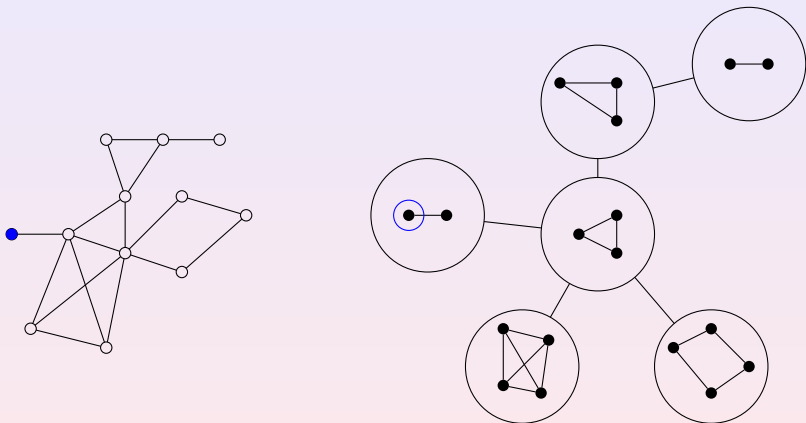
pathwidth de G

pw(G), largeur minimum
parmi les décompositions
linéaires

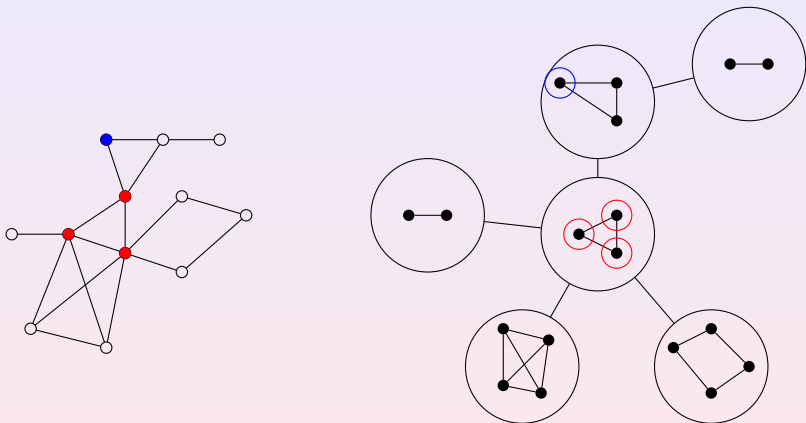
Décomposition arborescente et encerclement visible



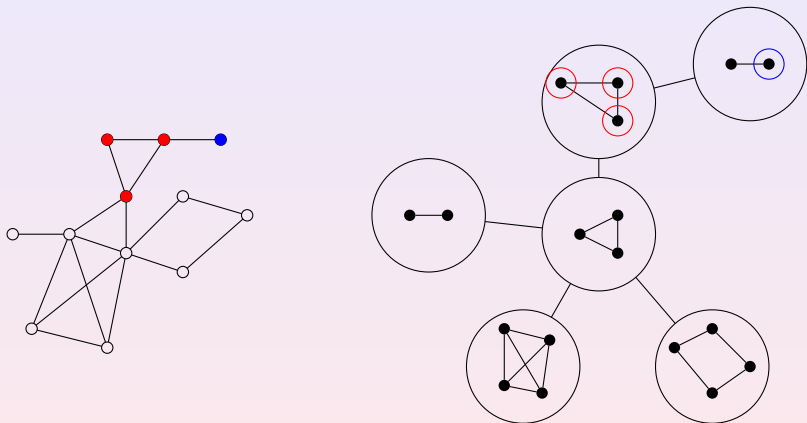
Décomposition arborescente et encerclement visible



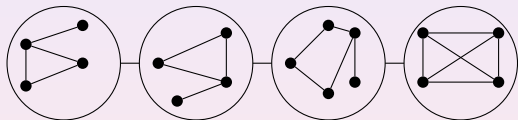
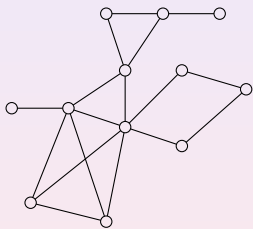
Décomposition arborescente et encerclement visible



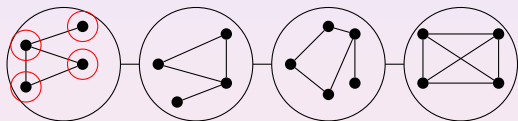
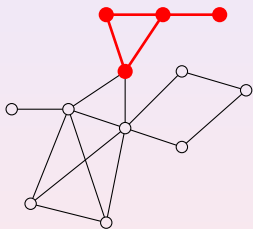
Décomposition arborescente et encerclement visible



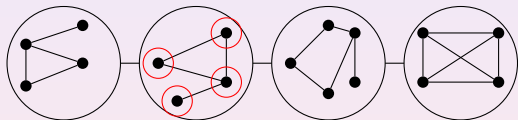
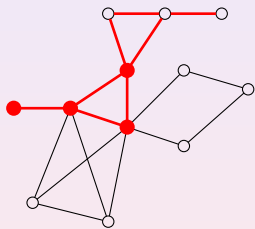
Décomposition linéaire et encerclement



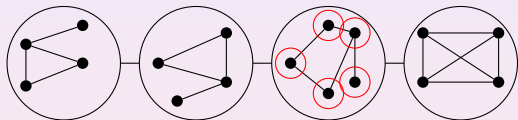
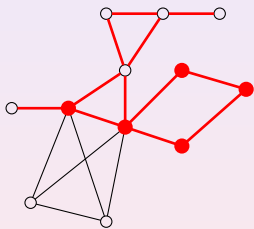
Décomposition linéaire et encerclement



Décomposition linéaire et encerclement



Décomposition linéaire et encerclement



Lien entre encerclement et largeur linéaire

Pour tout graphe G , $s(G) = \text{vertexseparation}(G) + 1$

J.A. Ellis, I.H. Sudborough et J.S. Turner. Vertex Separation and Search Number of a graph. [Inf. Comput.,1994]

Pour tout graphe G , $vs(G) = pw(G)$

N.G. Kinnersley. The Vertex Separation number of a Graph equals its Pathwidth. [IPL.,1992]

Pour tout graphe G :

$$s(G) = \mathbf{pw}(G) + 1$$

Pour tout graphe G : $vs(G) = tw(G) + 1$

Seymour et Thomas. Graph searching and min-max theorem for treewidth. [J.Combin. Theory, 1993.]

Lien entre encerclement et largeur linéaire

Pour tout graphe G , $s(G) = \text{vertexseparation}(G) + 1$

J.A. Ellis, I.H. Sudborough et J.S. Turner. Vertex Separation and Search Number of a graph. [Inf. Comput.,1994]

Pour tout graphe G , $vs(G) = pw(G)$

N.G. Kinnersley. The Vertex Separation number of a Graph equals its Pathwidth. [IPL.,1992]

Pour tout graphe G :

$$s(G) = \mathbf{pw}(G) + 1$$

Pour tout graphe G : $vs(G) = \mathbf{tw}(G) + 1$

Seymour et Thomas. Graph searching and min-max theorem for treewidth. [J.Combin. Theory, 1993.]

Plan

- 1 Introduction
- 2 Stratégie d'encerclement
- 3 Connexité**
 - Encerclement connexe
 - Prix de la connexité

Encerclement connexe

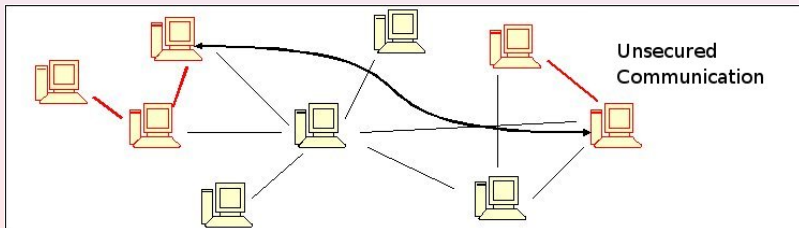
Limites du modèle de Parson

- Impossibilité de se déplacer à volonté dans la réalité
- Il est préférable que les agents restent *groupés*.

Encerclement connexe

Limites du modèle de Parson

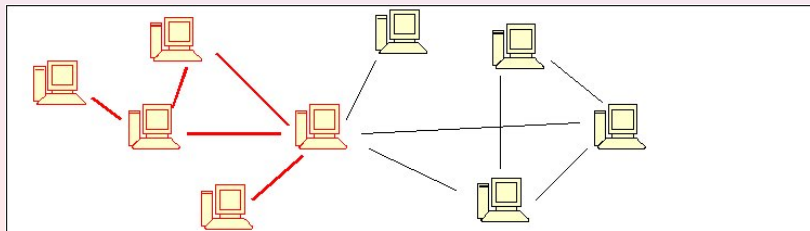
- Impossibilité de se déplacer à volonté dans la réalité
- Il est préférable que les agents restent *groupés*.



Encerclement connexe

Limites du modèle de Parson

- Impossibilité de se déplacer à volonté dans la réalité
- Il est préférable que les agents restent *groupés*.



Encerclement connexe

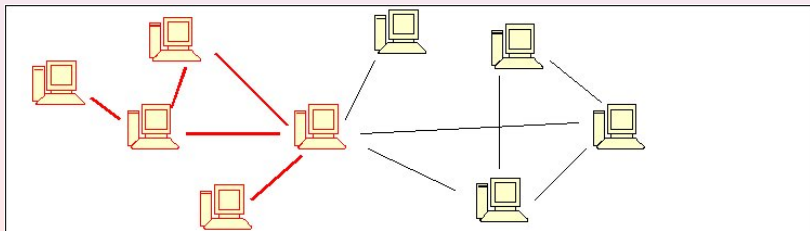
Limites du modèle de Parson

- Impossibilité de se déplacer à volonté dans la réalité
- Il est préférable que les agents restent *groupés*.

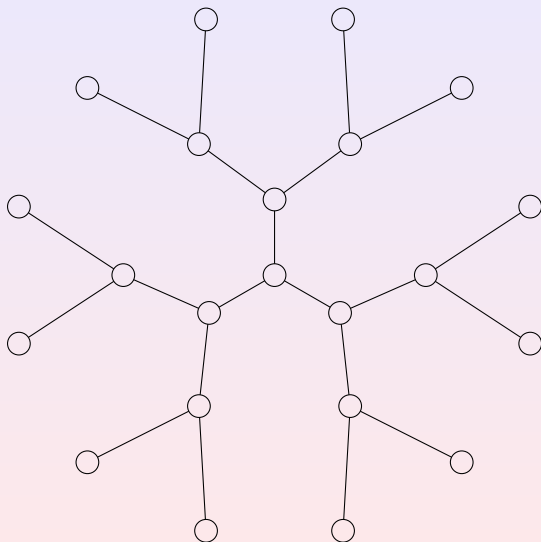
Stratégie d'encercllement connexe

A chaque étape, la partie nettoyée doit être connexe.

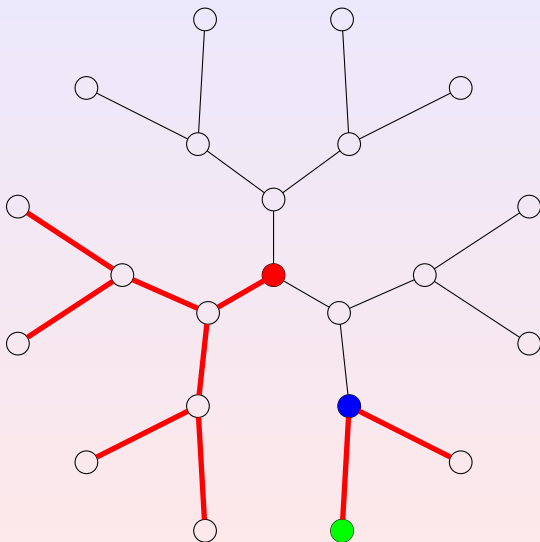
$cs(G)$ est le plus petit nombre d'agents nécessaires à une stratégie d'encercllement connexe dans le graphe.



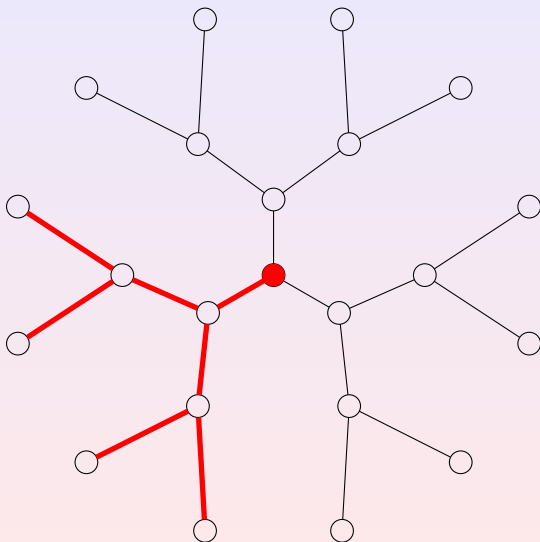
Connexe vs. non connexe



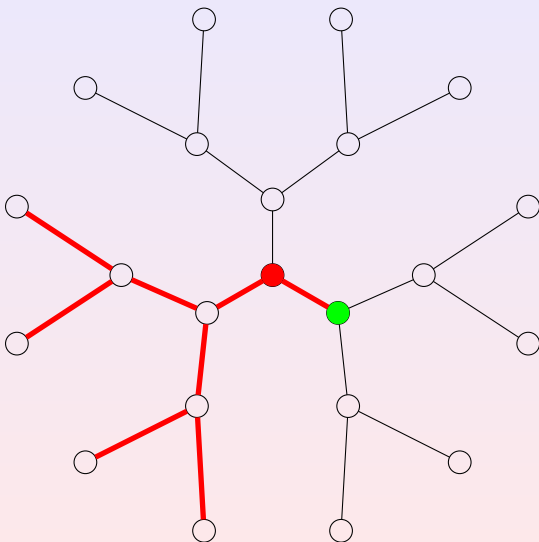
Connexe vs. non connexe



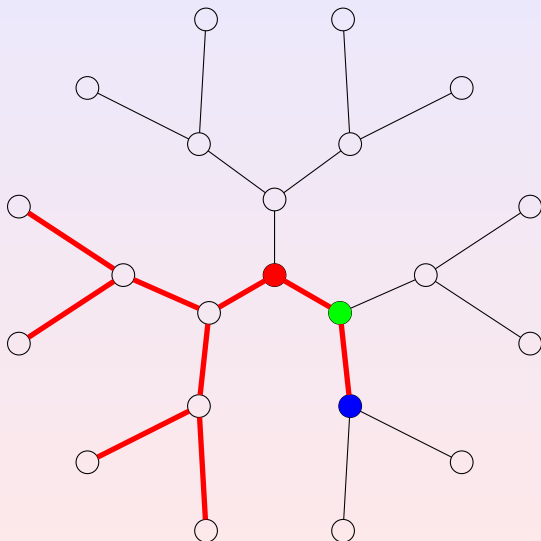
Connexe vs. non connexe



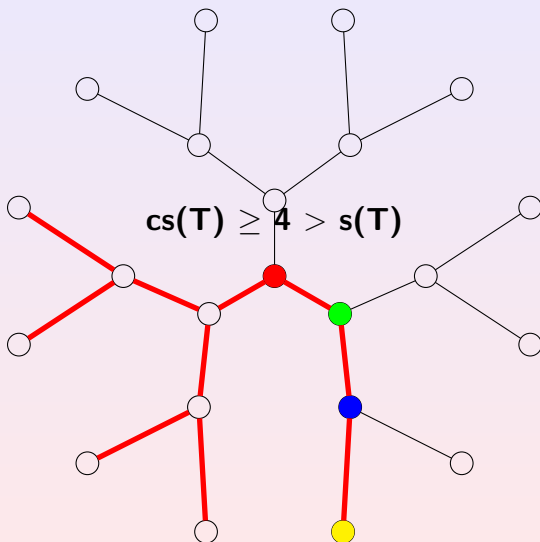
Connexe vs. non connexe



Connexe vs. non connexe



Connexe vs. non connexe



Prix de la connexité : cas des arbres

Barrière, Flocchini, Fraigniaud et Santoro. [SPAA, 2002]

Algorithme linéaire

Barrière, Fraigniaud, Santoro et Thilikos. [WG, 2003]

Pour tout arbre T , $s(T) \leq cs(T) \leq 2s(T) - 2$.

Ces bornes sont optimales.

Prix de la connexité : cas général

Théorème 1

Pour tout graphe connexe G , $\mathbf{cs}(G) \leq \mathbf{s}(G) (1 + \log_2 |V(G)|)$.

Fraigniaud et Nisse. [LATIN06]

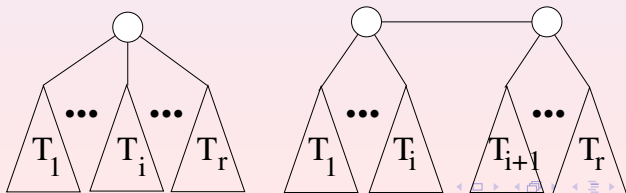
Preuve constructive

Etant donnée une décomposition arborescente du graphe G , notre algorithme calcule une stratégie d'encercllement connexe de G , avec au plus $\mathbf{tw}(G) \log |V(G)|$ agents et en temps polynomial.

Idée de la preuve du Théorème 1

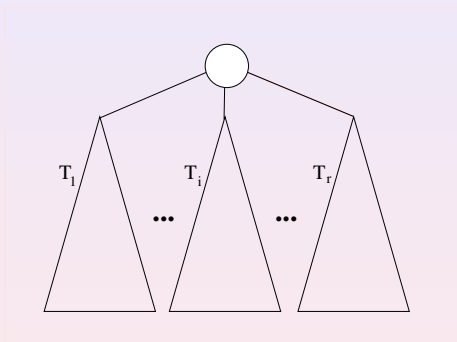
Pour tout graphe connexe G avec n sommets,
 $cs(G) \leq s(G) (1 + \log_2 n)$

- preuve par induction sur n
- **Robertson et Seymour.** Graph Minors II. Algorithmic Aspects of Tree-Width. J. of Alg 7, 1986.
 - Pour toute décomposition (T, X) d'un graphe G avec n sommets, il existe un (ou deux sommets adjacents) de T tel(s) que :
 - pour tout $1 \leq j \leq r$, $|G[T_j]| \leq n/2$



Idée de la preuve du Théorème 1

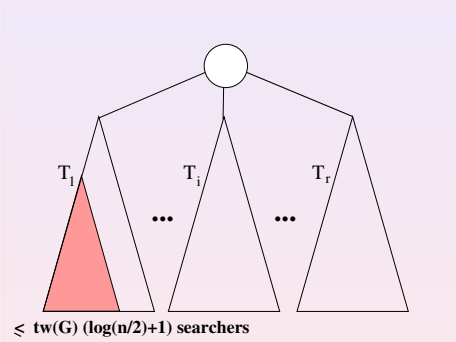
A partir d'une décomposition arborescente connexe de G



Pour tout $1 \leq i \leq r$, $G[T_i]$ est un sous-graphe connexe avec au plus $n/2$ sommets.

Idée de la preuve du Théorème 1

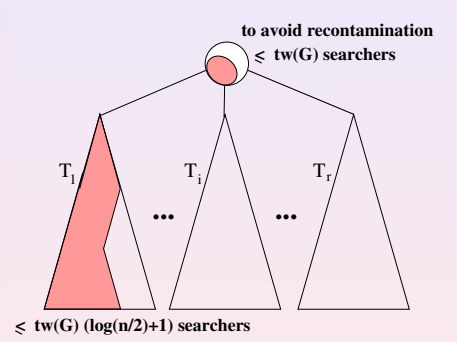
A partir d'une décomposition arborescente connexe de G



il existe une stratégie connexe de $G[T_1]$, avec au plus $\text{tw}(G)(\log(n/2) + 1)$ agents.

Idée de la preuve du Théorème 1

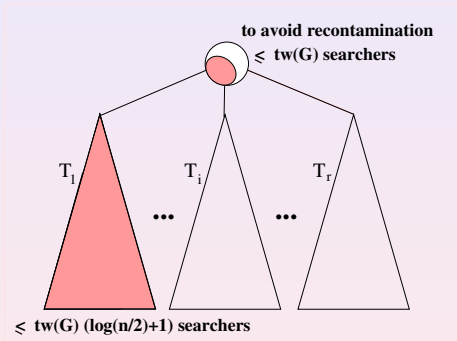
A partir d'une décomposition arborescente connexe de G



Au plus $\text{tw}(G)$ agents sont suffisants pour protéger $G[T_1]$ de la recontamination du reste de G .

Idée de la preuve du Théorème 1

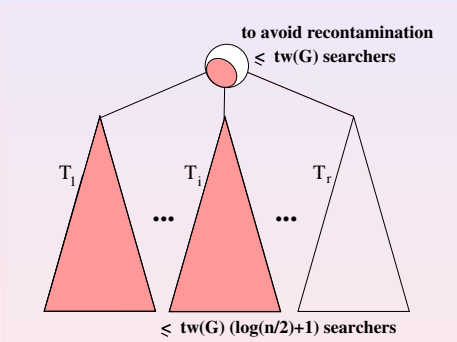
A partir d'une décomposition arborescente connexe de G



On peut terminer le nettoyage de $G[T_1]$.

Idée de la preuve du Théorème 1

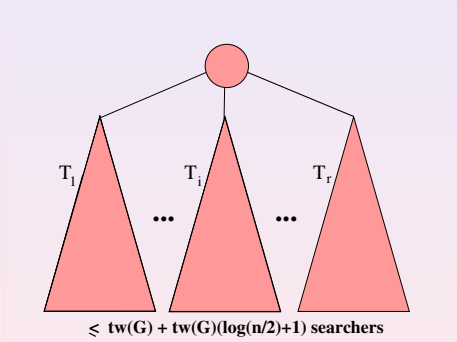
A partir d'une décomposition arborescente connexe de G



Alors, les $\text{tw}(G)(\log(n/2) + 1)$ agents peuvent être utilisés pour nettoyer un autre sous-graphe $G[T_i]$, etc...

Idée de la preuve du Théorème 1

A partir d'une décomposition arborescente connexe de G



Stratégie connexe utilisant au plus $\text{tw}(G)(\log n + 1)$ agents.
D'où, $\text{cs}(G) \leq \text{s}(G)(\log n + 1)$

Prix de la connexité : cas d'un fugitif visible

La borne précédente est optimale

Théorème 2, [Fraigniaud, N., WG06]

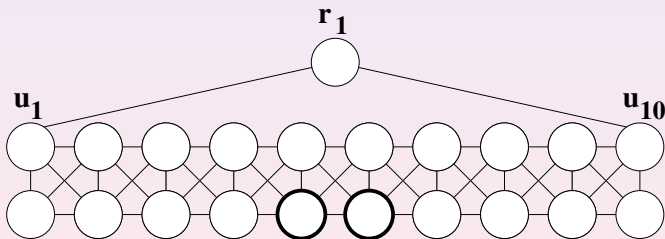
Pour tout n_0 , il existe $n \geq n_0$ et un graphe G avec n sommets tel que toute stratégie monotone visible pour G utilise au moins $\Omega(\mathbf{vs}(G) \cdot \log n)$ agents.

Plus précisément

Nous proposons une séquence de graphes $(\mathcal{G}_i)_{i \in \mathbb{N}}$, avec n_i sommets, tels que la séquence $(n_i)_{i \in \mathbb{N}}$ est strictement croissante, et, pour tout $i > 1$, $\mathbf{vs}(\mathcal{G}_i) \leq 5$ and $\mathbf{mcvs}(\mathcal{G}_i) \geq \Omega(\log n_i)$.

Idée de la preuve par induction sur i

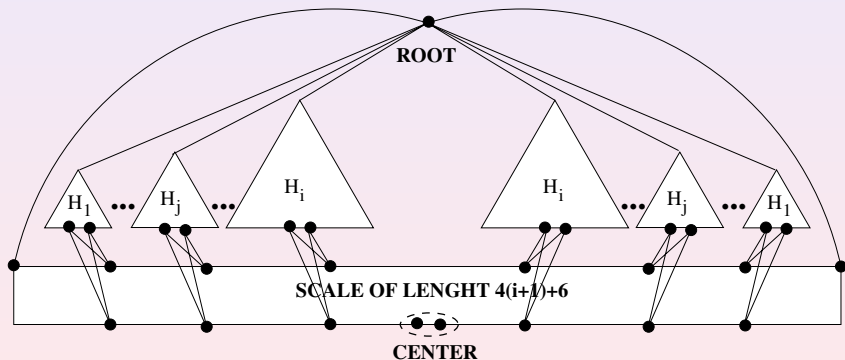
La séquence de graphes $(H_i)_{i \in \mathbb{N}}$ est définie récursivement.
 H_1 est constitué d'une échelle de longueur 10 plus une racine r_1 .



Graph H_1

Idée de la preuve par induction sur i

Pour tout $i > 1$, le graphe H_{i+1} est constitué de 2 copies de H_1, \dots, H_i et d'une échelle de longueur $4(i+1) + 6$.

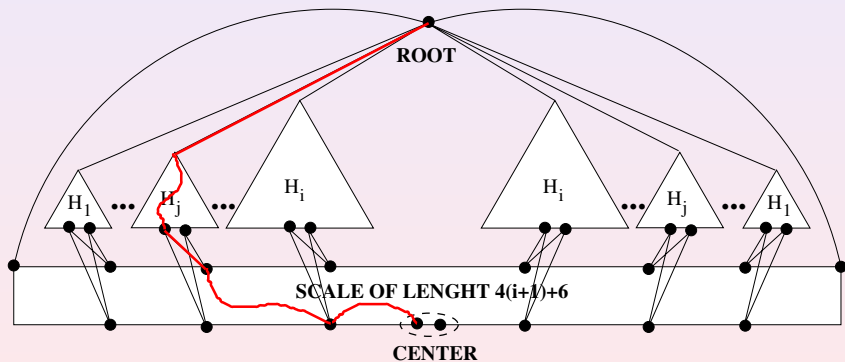


Graph H_{i+1}

27/30

Idée de la preuve par induction sur i

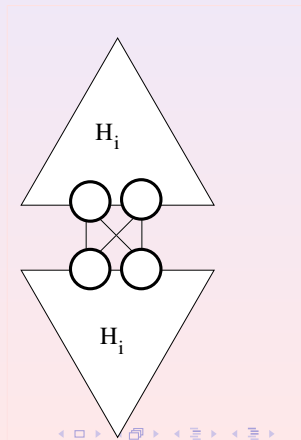
Lemme : Capturer le fugitif à la racine, en commençant du centre requiert au moins $2i$ agents.



Graph H_{i+1}

Idée de la preuve par induction sur i

Pour tout $i \geq 1$, $\mathcal{G}_i = 2$ copies de H_i liées par une 4-clique.

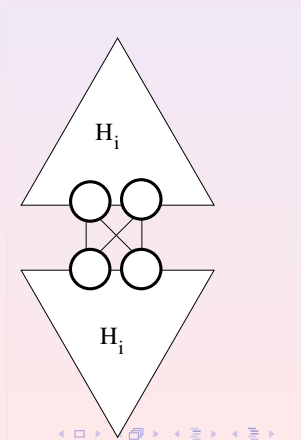


Idée de la preuve par induction sur i

Pour tout $i \geq 1$, $G_i = 2$ copies de H_i liées par une 4-clique.

Encerclement visible de G_i

Par induction : $vs(G_i) = 5$.



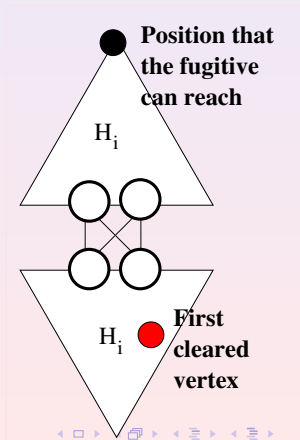
Idée de la preuve par induction sur i

Pour tout $i \geq 1$, $G_i = 2$ copies de H_i liées par une 4-clique.

Encerclement visible de G_i

Par induction : $vs(G_i) = 5$.

Quel que soit le premier sommet propre, le fugitif peut fuir dans l'autre copie de H_i .



Idée de la preuve par induction sur i

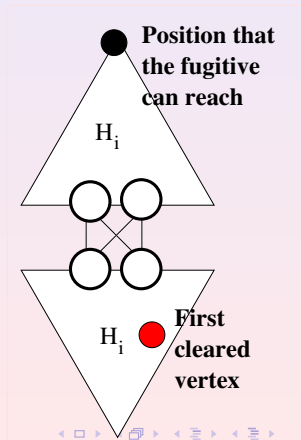
Pour tout $i \geq 1$, $G_i = 2$ copies de H_i liées par une 4-clique.

Encerclement visible de G_i

Par induction : $vs(G_i) = 5$.

Quel que soit le premier sommet propre, le fugitif peut fuir dans l'autre copie de H_i .

Donc, $2i$ agents sont nécessaires, i.e. $vs(G_i) \log n$ agents.



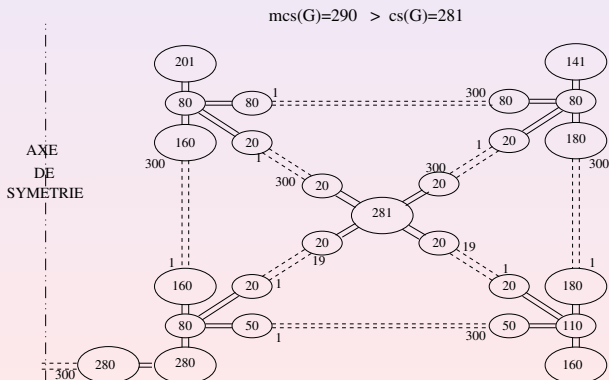
Connexe vs. non connexe

Déterminer $cs(G)$ (resp. $cvs(G)$) est NP-difficile.
Question : est-ce NP-complet ?

Connexe vs. non connexe

Dans le cas connexe, la recontamination aide

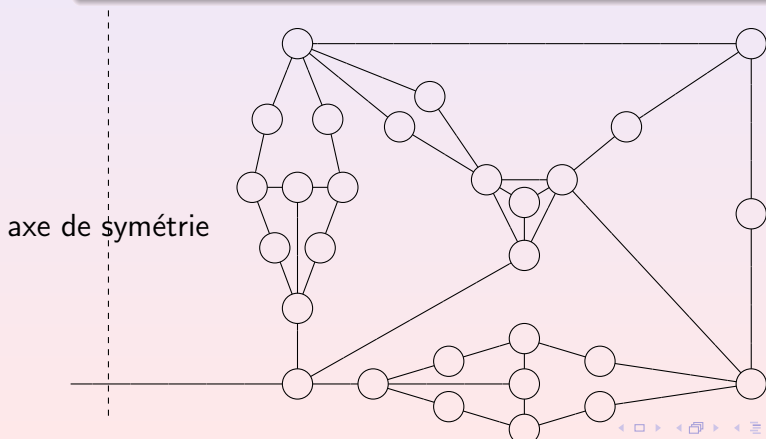
Yang, Dyer, Alspach. Sweeping Graphs and Digraphs.
[ISAAC04] (cas d'un fugitif invisible)



Connexe vs non connexe

Dans le cas connexe visible, la recontamination aide

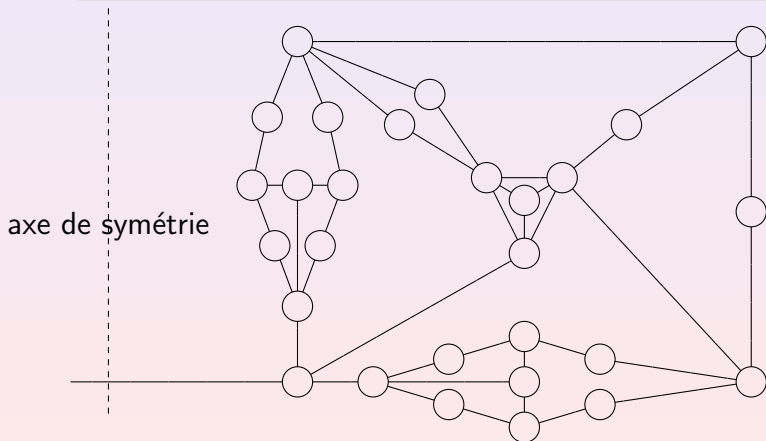
\exists des graphes G tel que $mcvs(G) > cvs(G)$. [Fraigniaud, N., WG06]



Connexe vs non connexe

Cas d'un fugitif visible

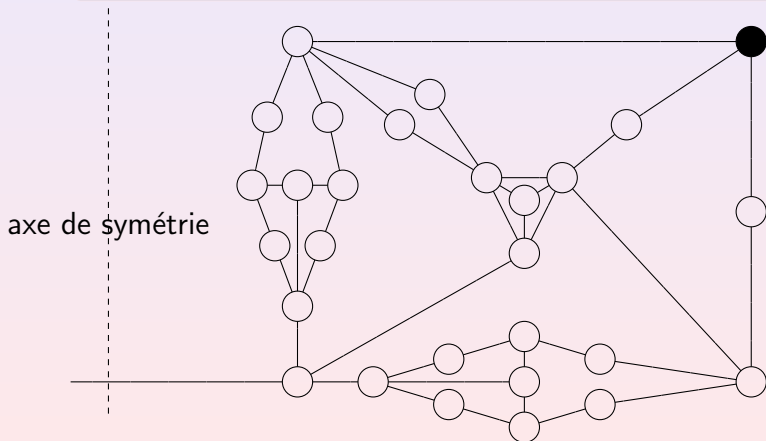
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

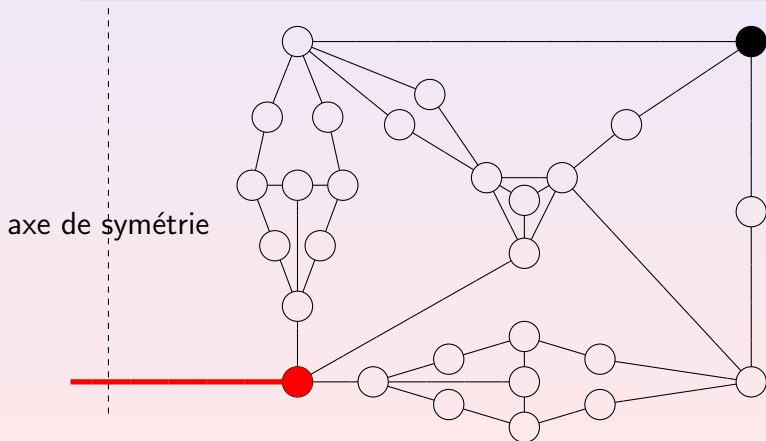
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

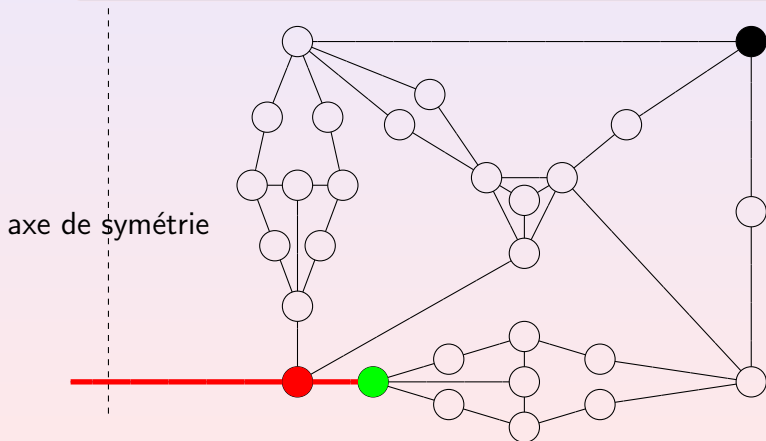
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

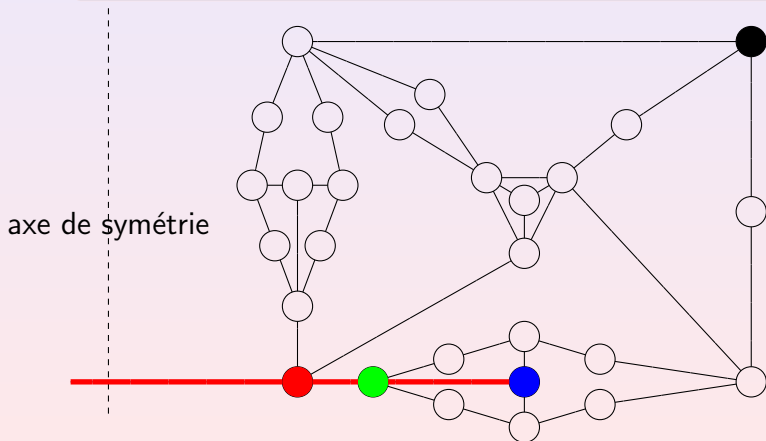
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

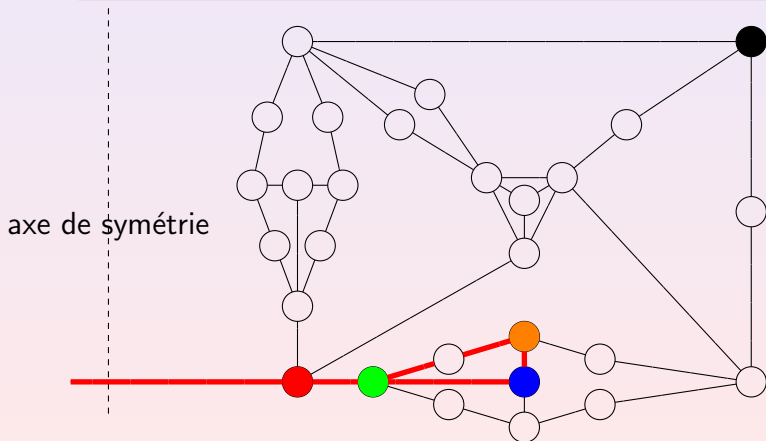
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

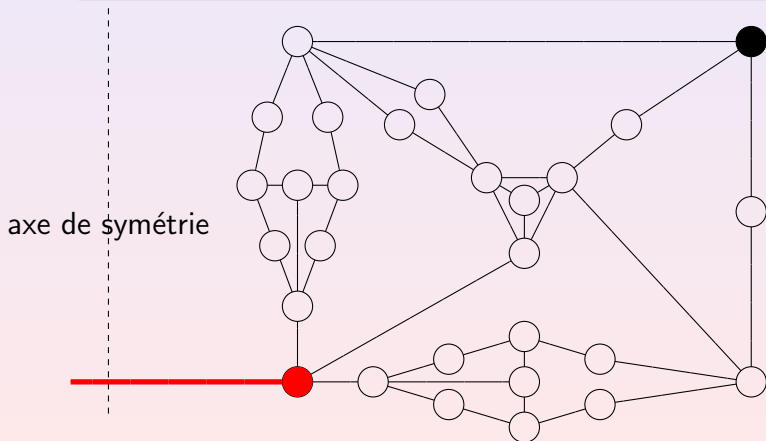
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

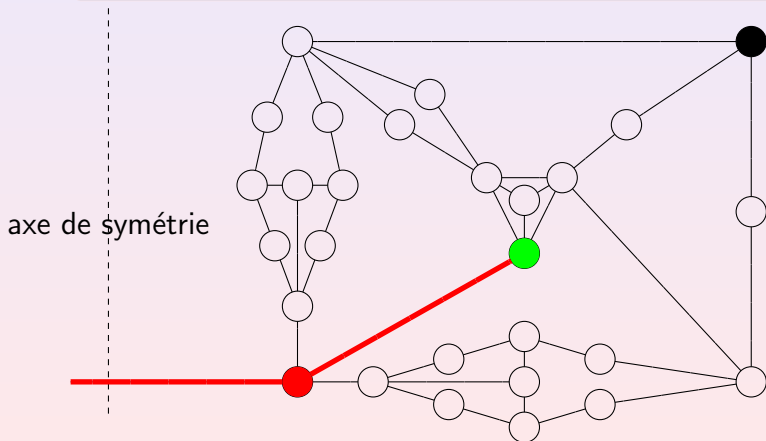
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

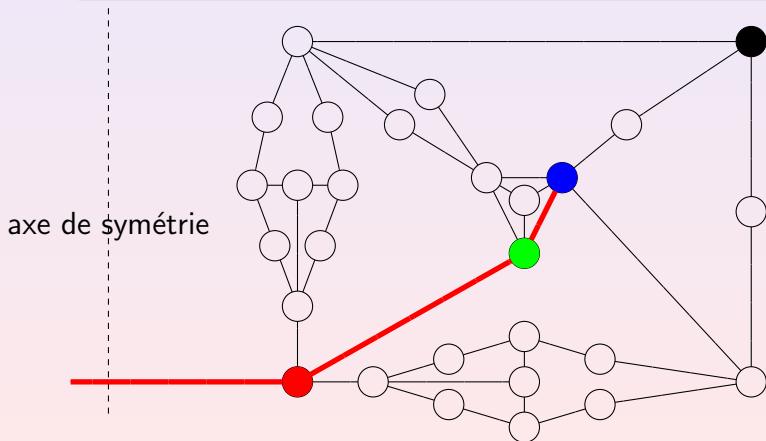
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

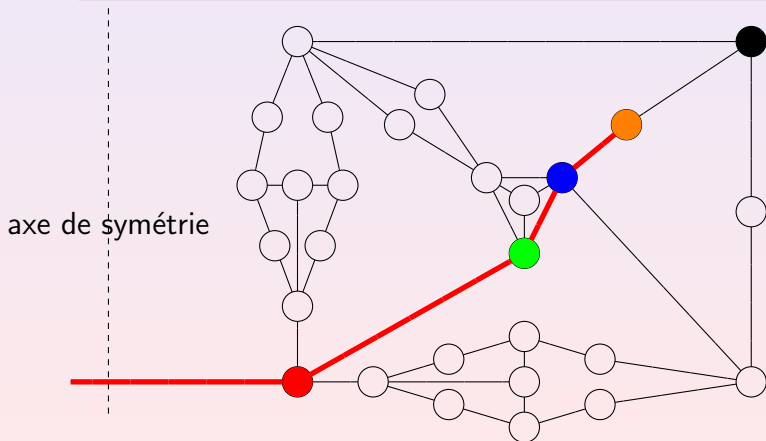
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

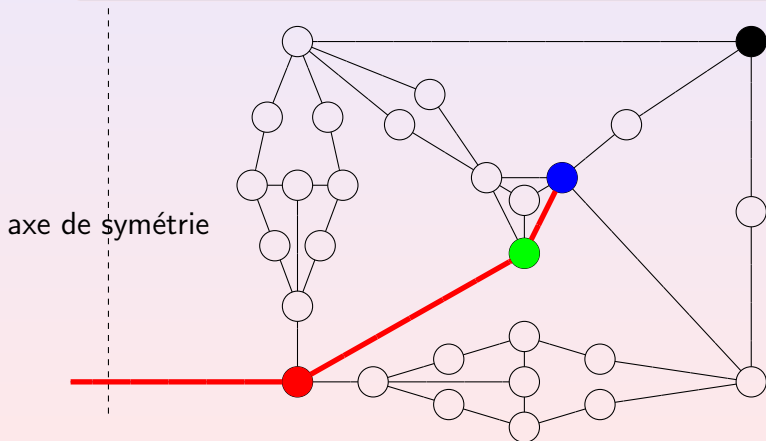
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

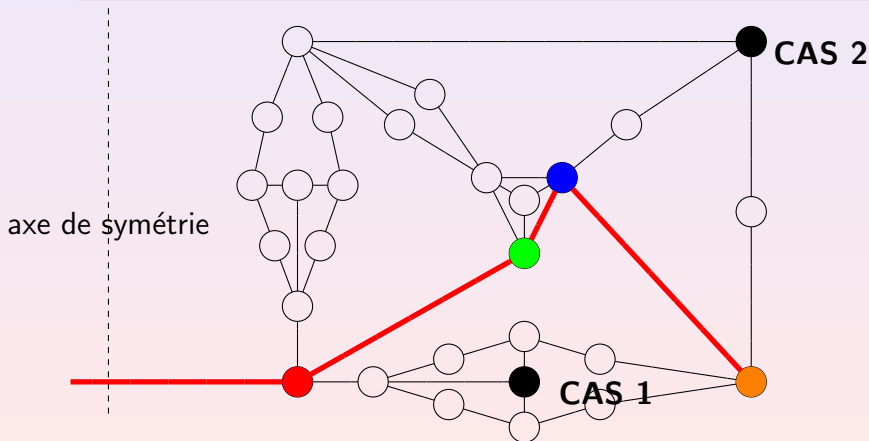
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

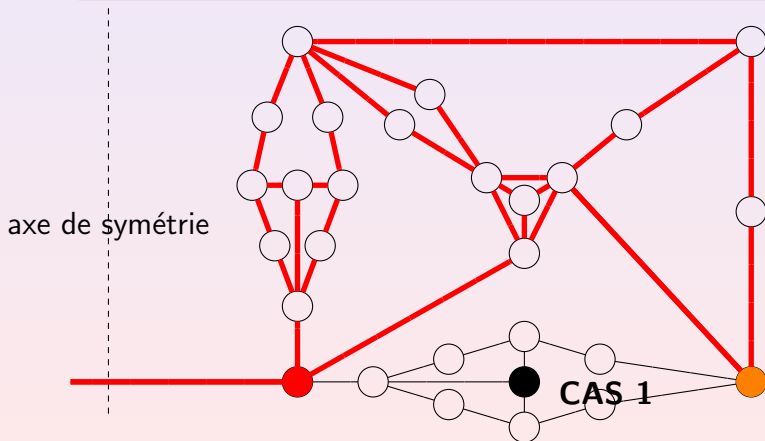
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

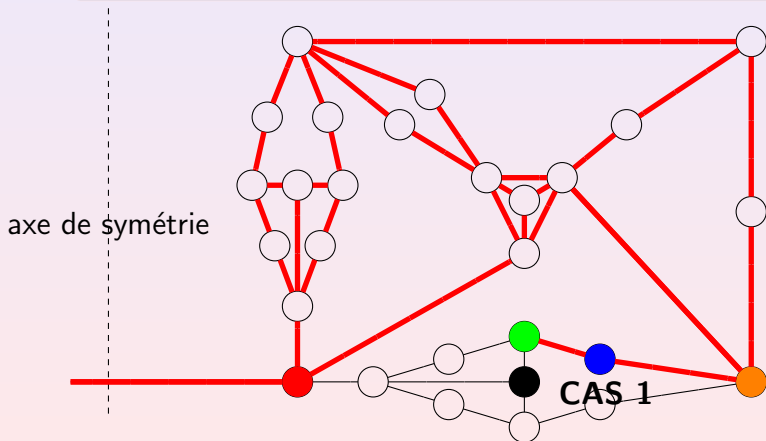
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

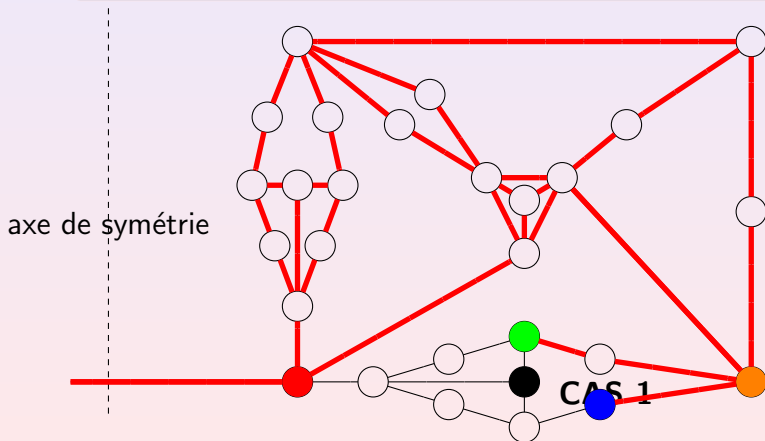
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

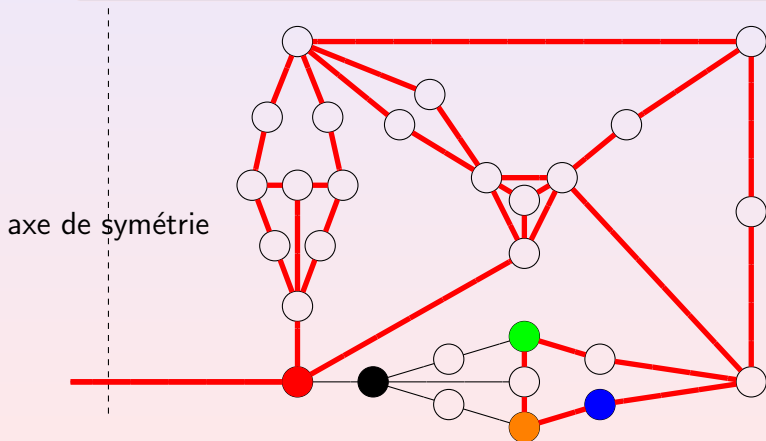
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

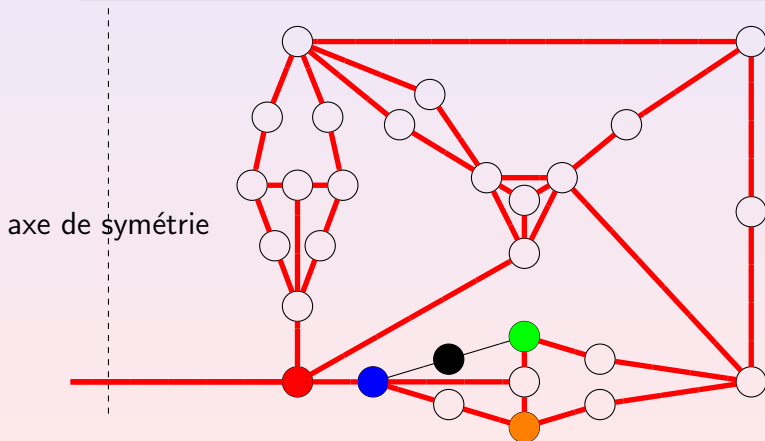
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

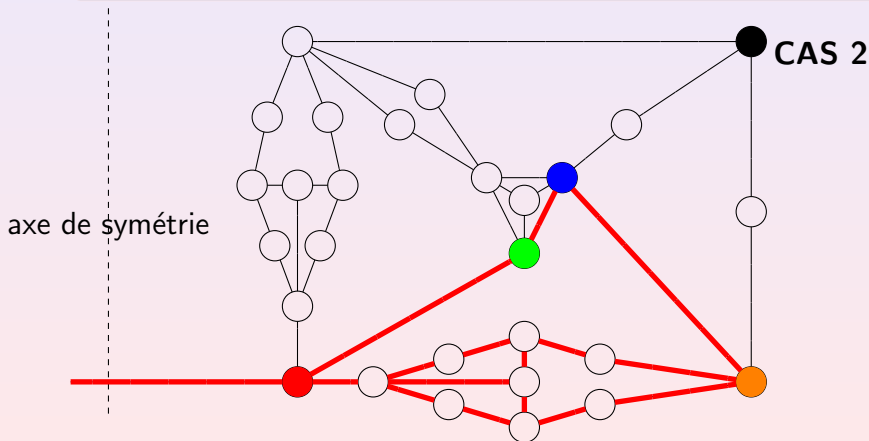
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

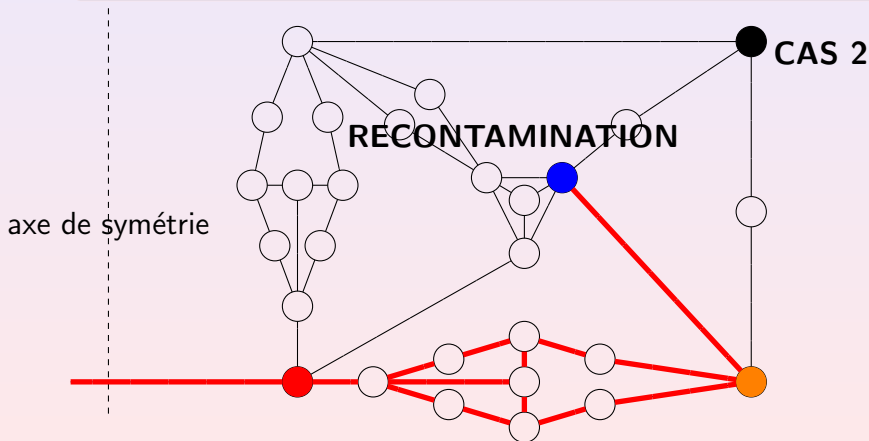
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

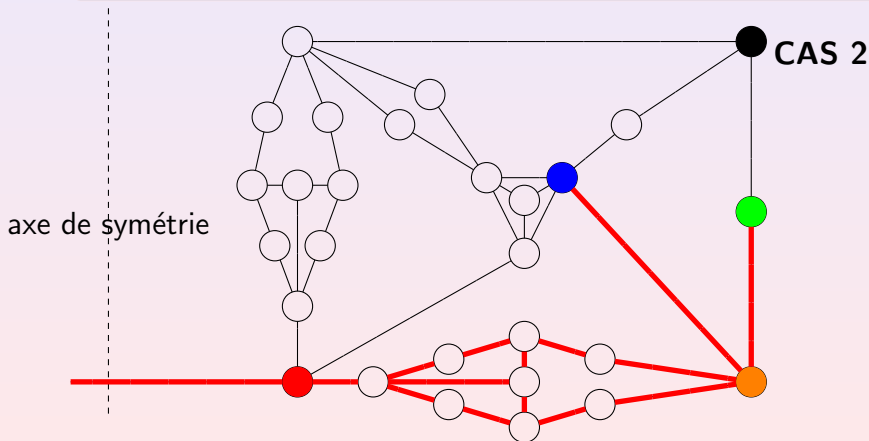
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

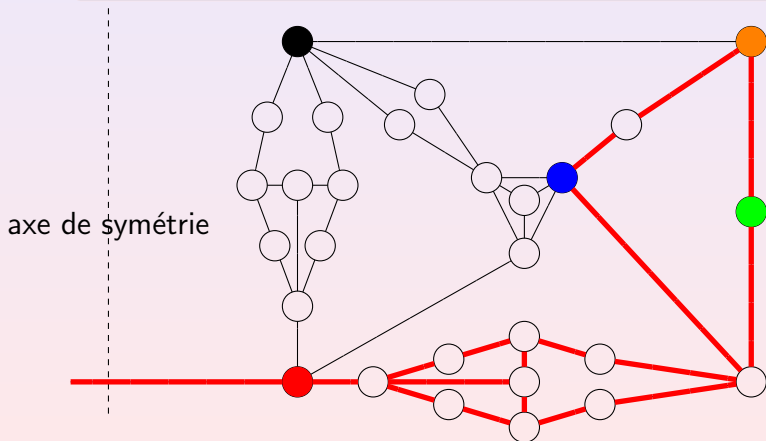
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

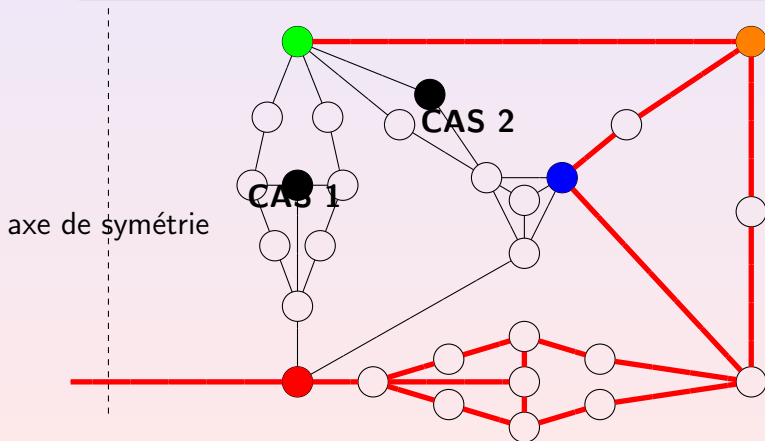
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

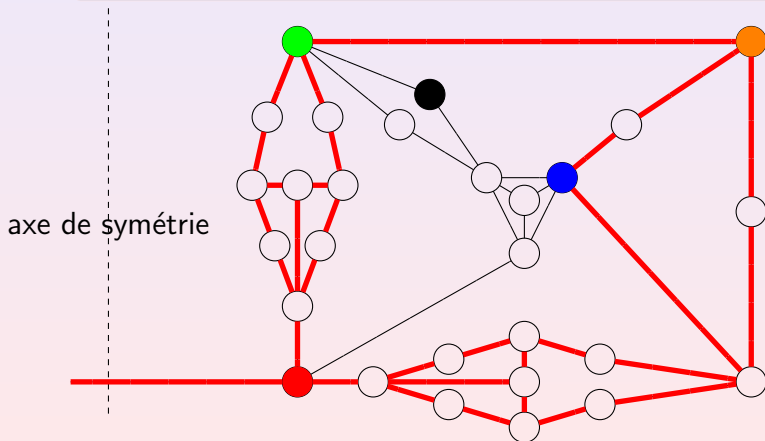
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

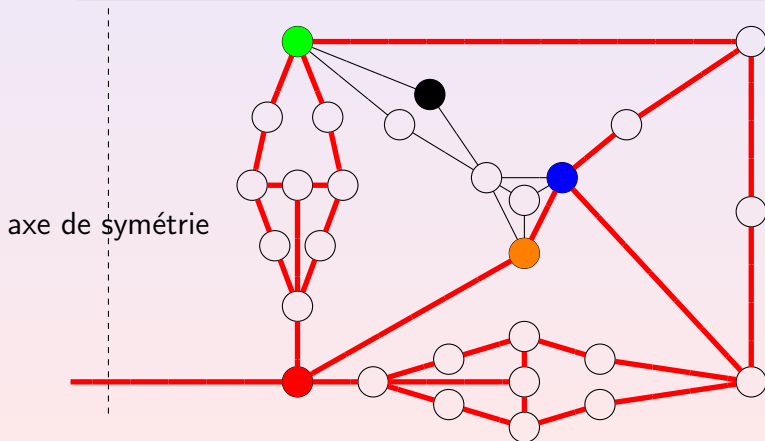
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

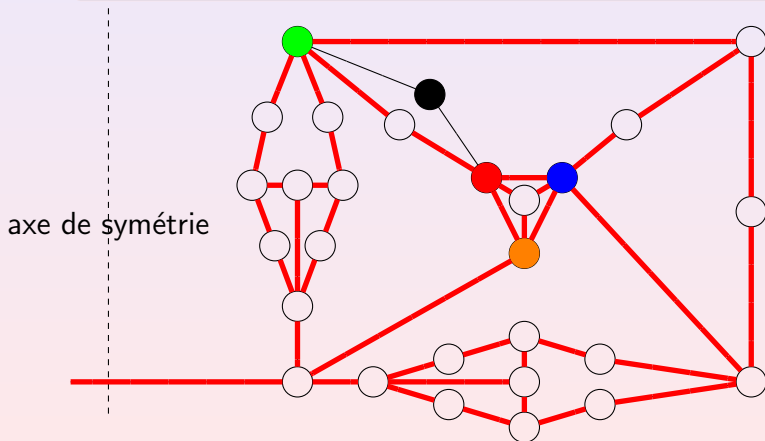
Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Connexe vs non connexe

Cas d'un fugitif visible

Soit G le graphe ci-dessous : $mcvs(G) > cvs(G) = 4$.



Conclusion et Perspectives

Problèmes ouverts

- Quelle est la borne optimale ?
Dans les arbres : $cs(T)/s(T) \leq 2$, borne optimale [Barrière et al.].
Si le fugitif est visible : $cs(G)/s(G) \leq \log n$, borne optimale [Fraigniaud, Nisse]
- Le problème de calculer $cs(G)$ est-il NP-complet ?
problème NP-difficile.
- FPT ?