Stratégie d'encerclement avec conseil

Nicolas Nisse David Soguet

LRI, Université Paris-Sud, France.

AlgoTel, mai 2007

Encerclement dans les graphes

Nettoyage de réseaux contaminés

Etant donné un réseau,

- dont les arêtes sont contaminées ;
- une équipe d'agents doit nettoyer ce réseau.

Nous voulons trouver une stratégie qui nettoie le réseau **en utilisant le moins d'agents possible.**

Motivations

- Sécurité des réseaux informatiques,
- Nettoyage d'un réseau de pipeline contaminé,
- ...

Encerclement dans les graphes

Nettoyage de réseaux contaminés

Etant donné un réseau,

- dont les arêtes sont contaminées ;
- une équipe d'agents doit nettoyer ce réseau.

Nous voulons trouver une stratégie qui nettoie le réseau **en utilisant le moins d'agents possible.**

Motivations

- Sécurité des réseaux informatiques,
- Nettoyage d'un réseau de pipeline contaminé,
- ...

Encerclement réparti dans les graphes

Nettoyage réparti de réseaux contaminés

- les agents doivent déterminer la stratégie eux-même ;
- la stratégie doit être rapide (temps polynomial).

Problème :

Proposer un *protocole réparti* qui permet au *nombre minimum* d'agents possible de nettoyer le réseau.

Les agents doivent déterminer *eux-même*, et *de manière répartie*, une stratégie d'encerclement, et exécuter cette stratégie.

Stratégies d'encerclement

Les agents se déplacent le long des arêtes du graphe.

Une arête est nettoyée lorsqu'elle est traversée par un agent.

Une arête e propre est recontaminée s'il existe un chemin libre d'agent entre e et une arête contaminée.

Une stratégie consiste à :

- Initialement, tous les agents sont placer sur la base v_0 ;
- succession de déplacements d'un agent le long d'une arête *si cela n'induit pas de recontamination* ;
- jusqu'à ce que le graphe soit entièrement propre.

$\mathbf{s}(G, v_0)$: nombre minimum d'agents pour nettoyer le graphe G de cette façon, à partir de v_0.

Stratégies d'encerclement

Les agents se déplacent le long des arêtes du graphe.

Une arête est nettoyée lorsqu'elle est traversée par un agent.

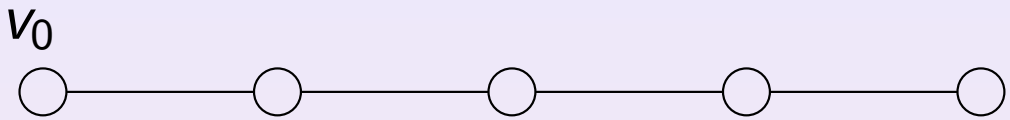
Une arête e propre est recontaminée s'il existe un chemin libre d'agent entre e et une arête contaminée.

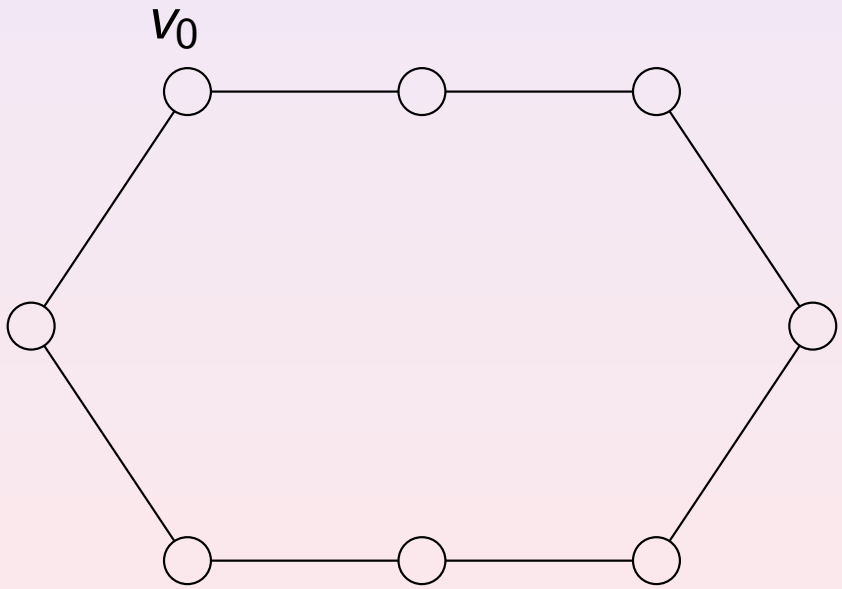
Une stratégie consiste à :

- Initialement, tous les agents sont placer sur la base v_0 ;
- succession de déplacements d'un agent le long d'une arête *si cela n'induit pas de recontamination* ;
- jusqu'à ce que le graphe soit entièrement propre.

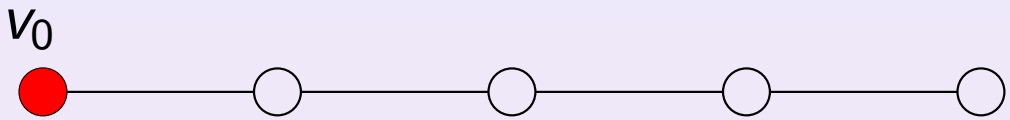
$\mathbf{s}(G, v_0)$: nombre minimum d'agents pour nettoyer le graphe G de cette façon, à partir de v_0.

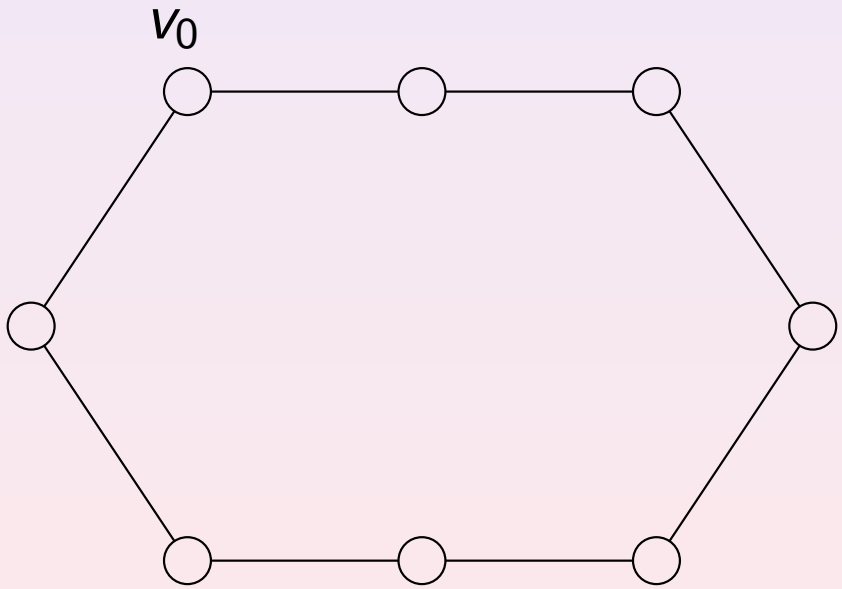
Deux exemples simples : le chemin et l'anneau



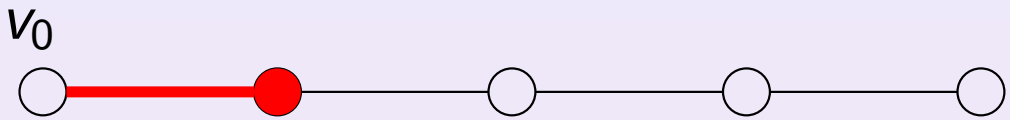


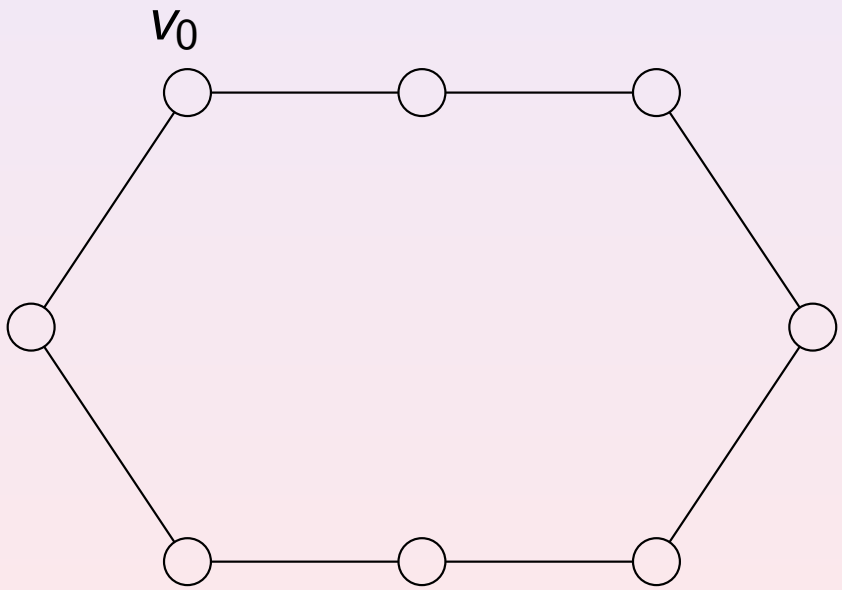
Deux exemples simples : le chemin et l'anneau



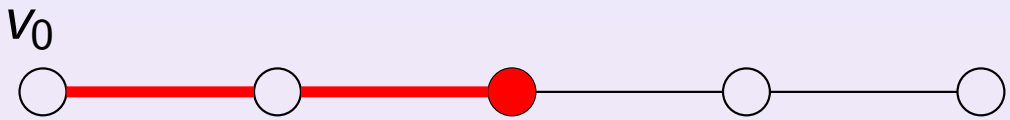


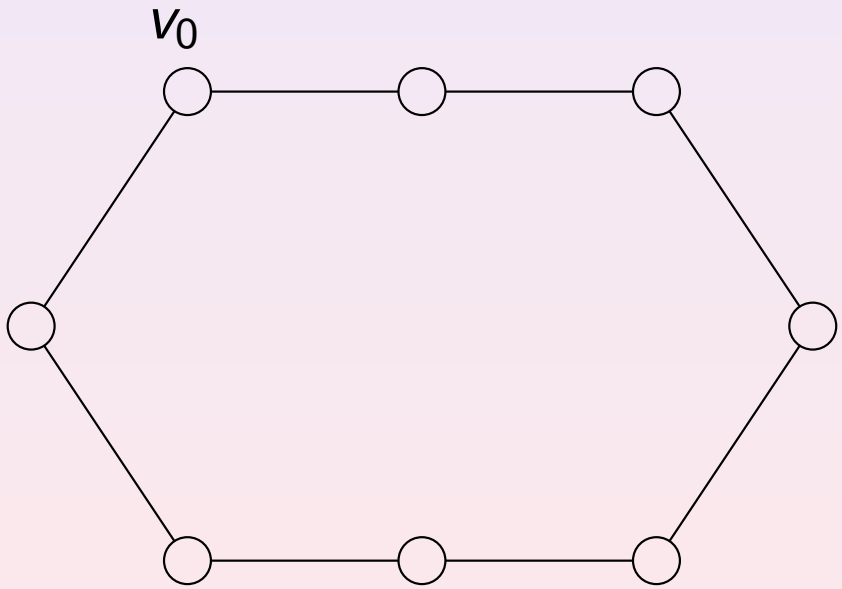
Deux exemples simples : le chemin et l'anneau



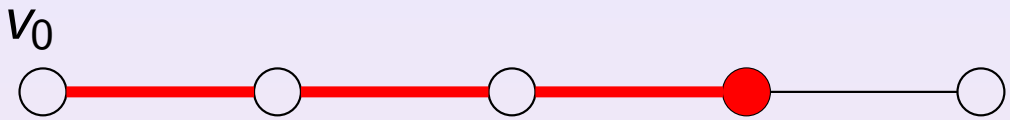


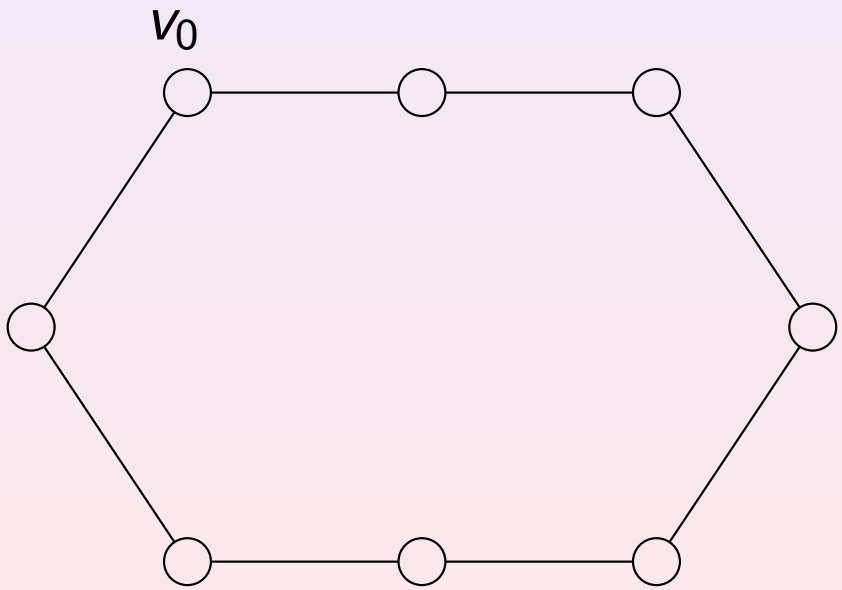
Deux exemples simples : le chemin et l'anneau



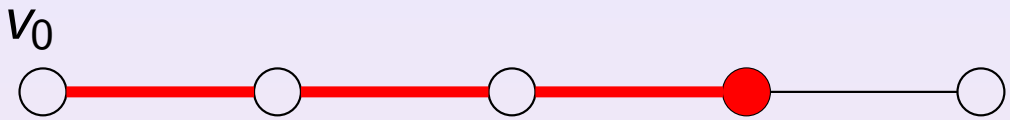


Deux exemples simples : le chemin et l'anneau

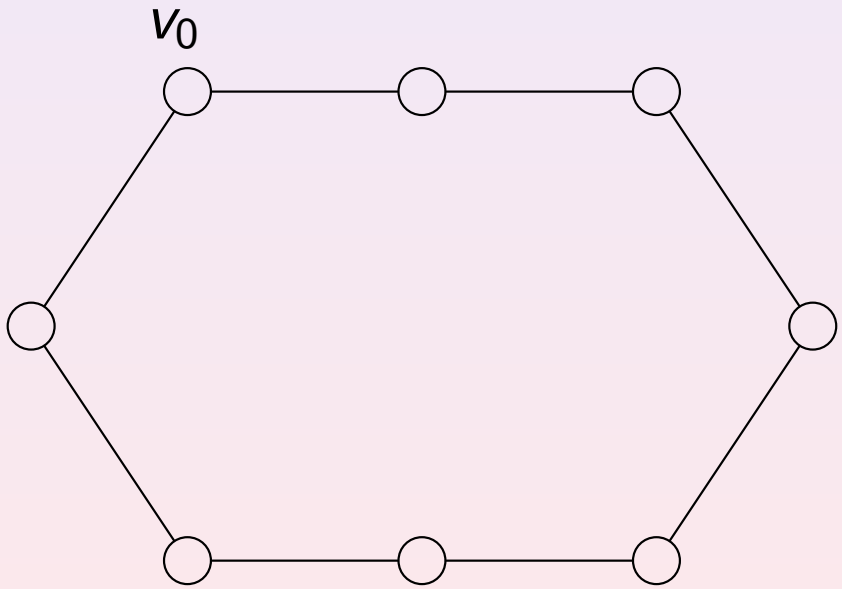




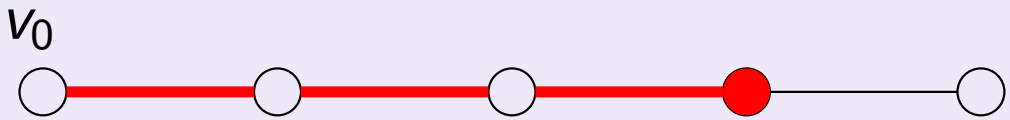
Deux exemples simples : le chemin et l'anneau



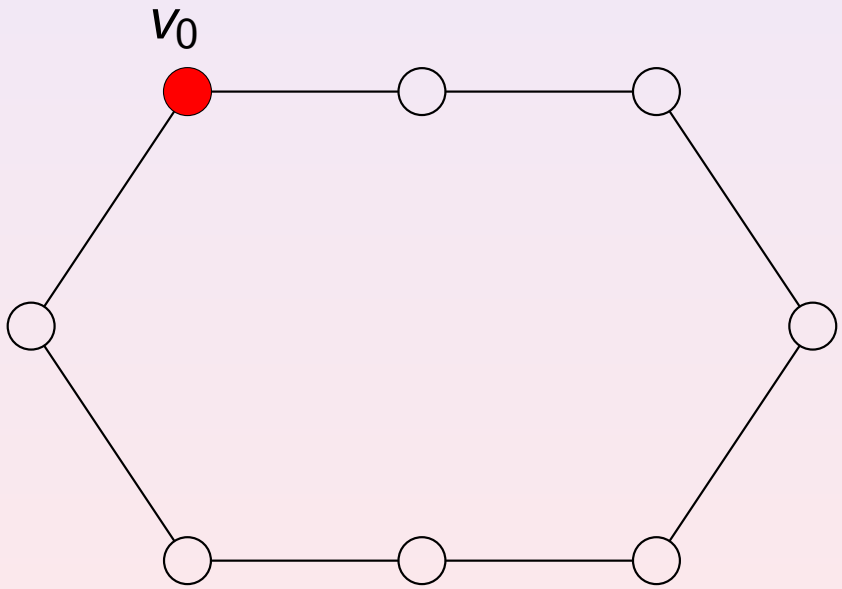
$\mathbf{s(Chemin, v_0)=1}$



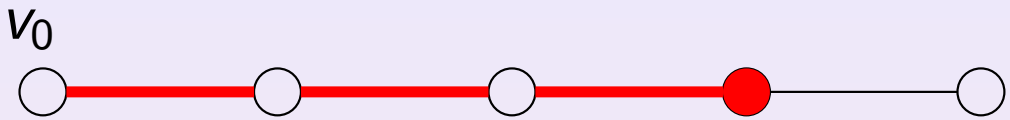
Deux exemples simples : le chemin et l'anneau



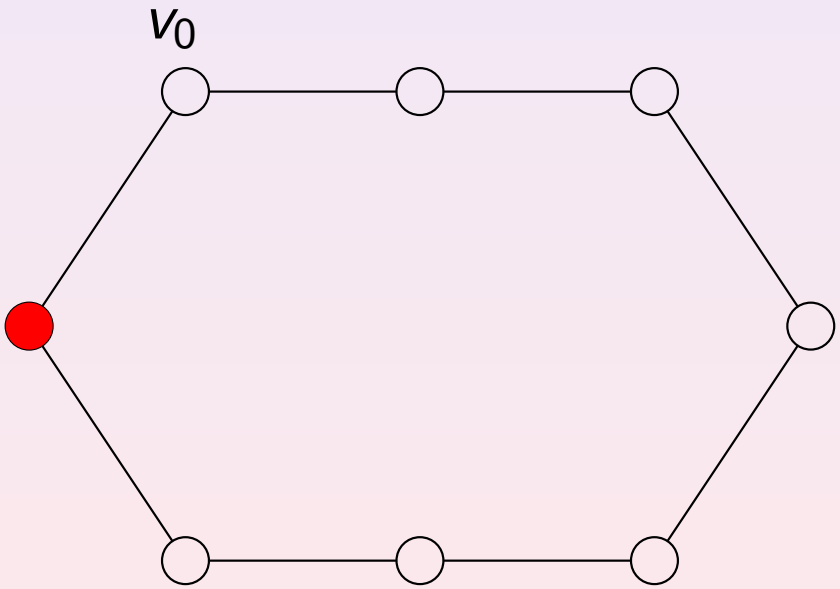
s(Chemin, v_0)=1



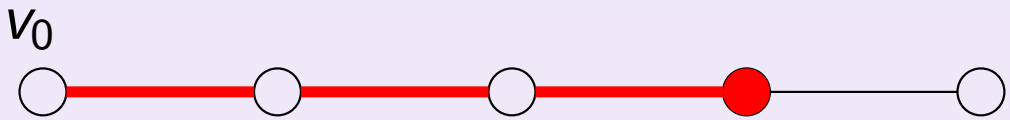
Deux exemples simples : le chemin et l'anneau



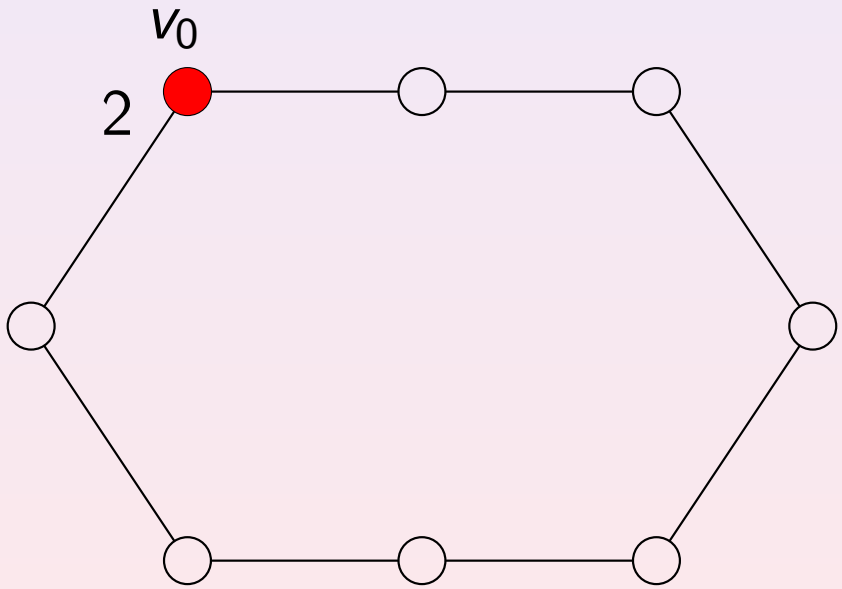
s(Chemin, v_0)=1



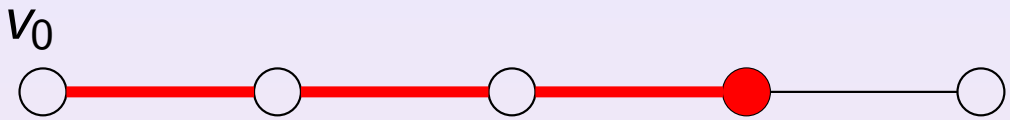
Deux exemples simples : le chemin et l'anneau



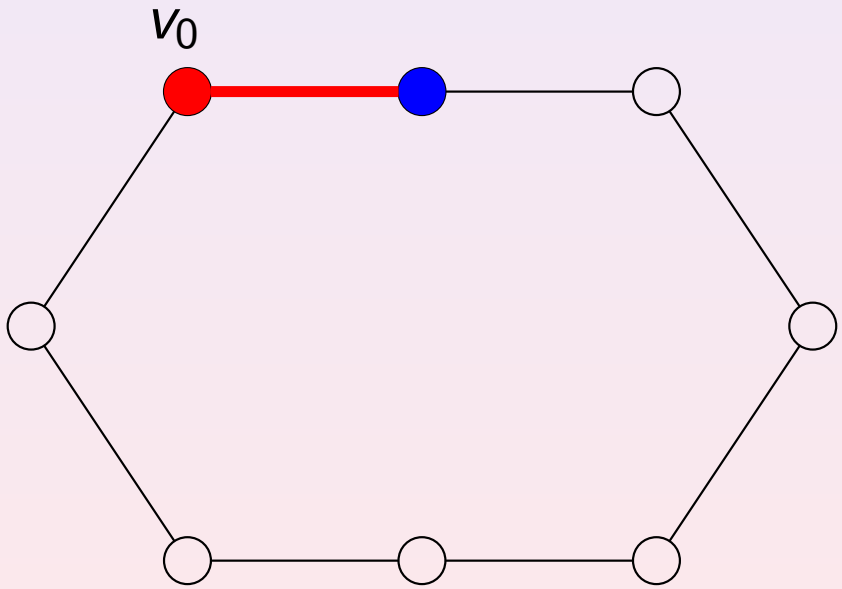
s(Chemin, v_0)=1



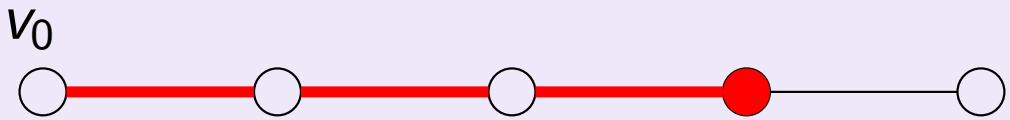
Deux exemples simples : le chemin et l'anneau



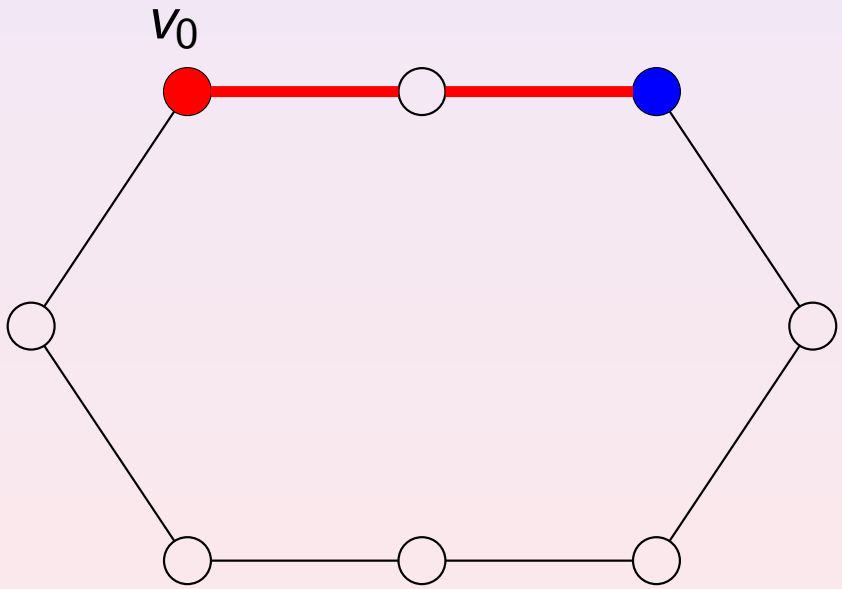
s(Chemin, v_0)=1



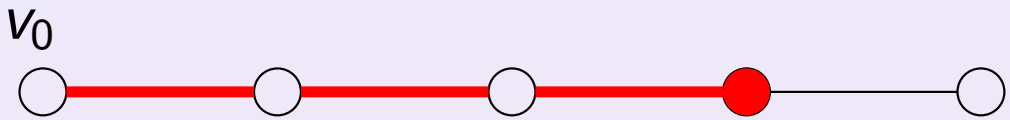
Deux exemples simples : le chemin et l'anneau



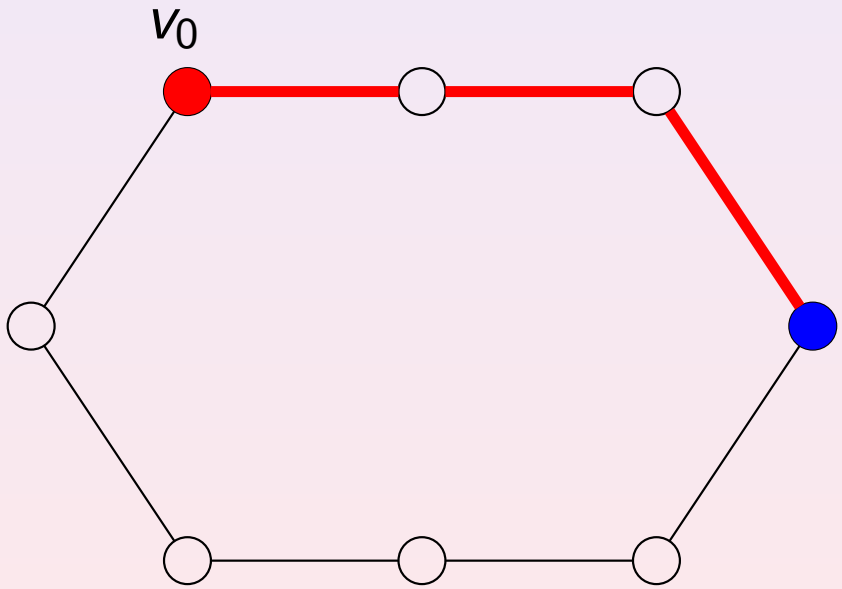
s(Chemin, v_0)=1



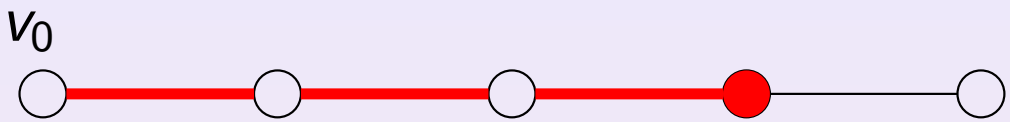
Deux exemples simples : le chemin et l'anneau



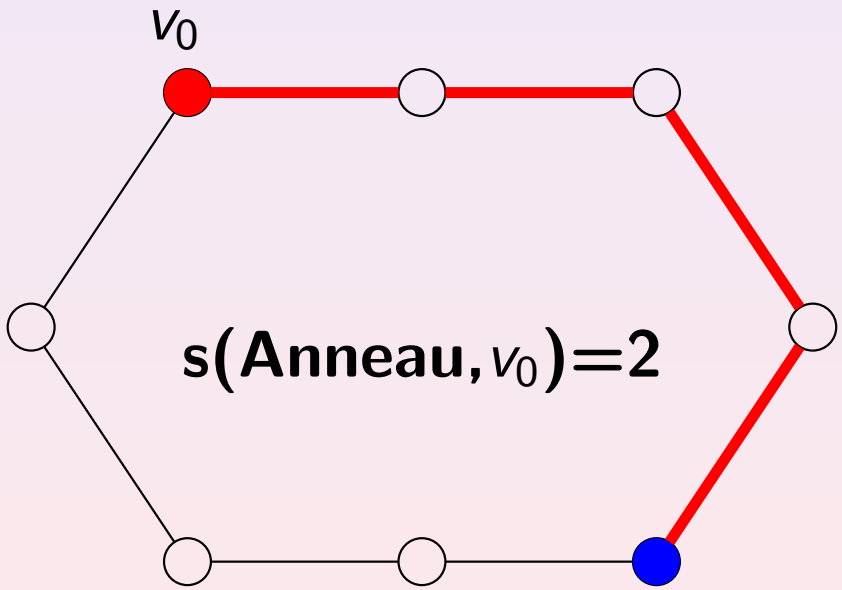
$\mathbf{s(Chemin, v_0)=1}$



Deux exemples simples : le chemin et l'anneau



s(Chemin, v_0)=1



Stratégies d'encerclement monotone connexe

Stratégies monotones connexes

- Monotonie : la zone propre ne peut pas diminuer (i.e., il n'y a jamais de recontamination)
  $\Rightarrow$ **temps polynomial**
- Connexité : la zone propre est connexe
  $\Rightarrow$ **communications sûres**.

Remarque : déterminer $\mathbf{s}(G, v_0)$ et la stratégie monotone et connexe correspondante est NP-complet dans un contexte centralisé [Megiddo *at al.* 88]

Stratégies d'encerclement monotone connexe

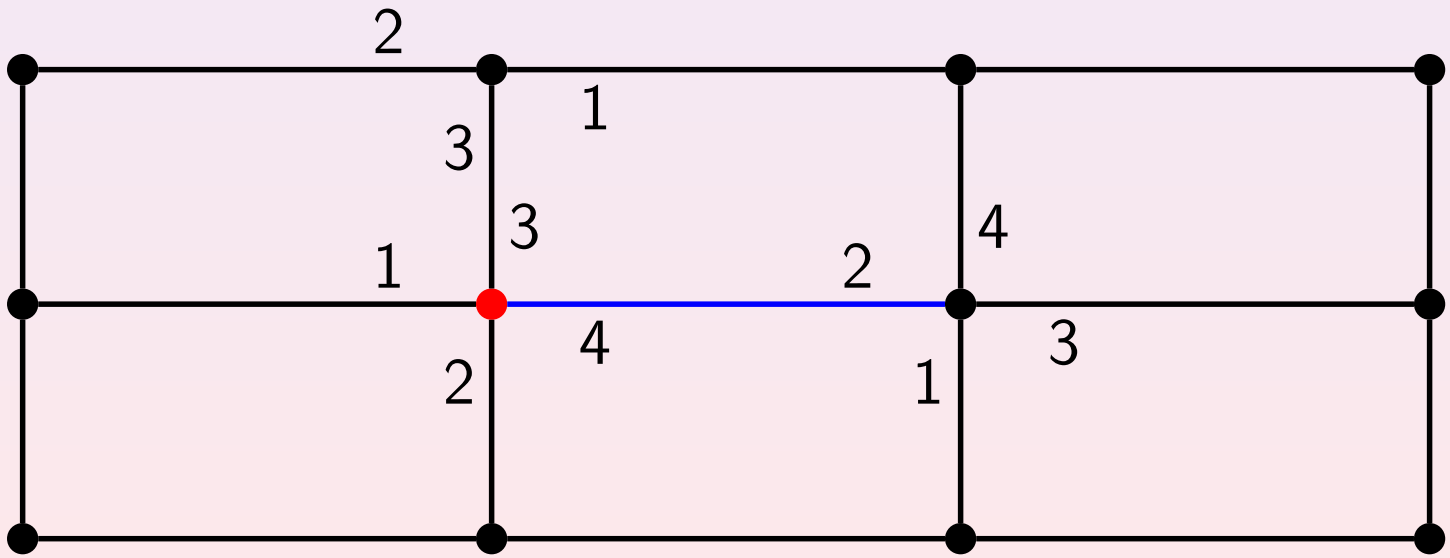
Stratégies monotones connexes

- Monotonie : la zone propre ne peut pas diminuer (i.e., il n'y a jamais de recontamination)
  $\Rightarrow$ **temps polynomial**
- Connexité : la zone propre est connexe
  $\Rightarrow$ **communications sûres**.

Remarque : déterminer $\mathbf{s}(G, v_0)$ et la stratégie monotone et connexe correspondante est NP-complet dans un contexte centralisé [Megiddo *at al.* 88]

Modèle : Environnement

- graphe connexe non orienté ;
- anonyme (pas d'étiquetage des sommets) ;
- orientation locale des arêtes ;
- environnement synchrone/asynchrone ;



Modèle : Agents

- entités mobiles autonomes avec identifiants distincts ;
- automates de mémoire $O(\log n)$ bits.

Décision locale qui dépend de :

- l'état courant ;
- l'état des autres agents présents sur le sommet ;
- éventuellement, le numéro de port d'arrivée.

L'agent peut décider de :

- quitter un sommet par un certain port ;
- changer d'état ;

Etat de l'art 1/2

La topologie du graphe est connue à l'avance par les agents.

Nettoyage de topologies particulières

- **Arbres** [Barrière, Flocchini, Fraigniaud et Santoro. 2002]
- **Grilles** [Flocchini, Luccio et Song. 2005]
- **Hypercubes** [Flocchini, Huang et Luccio. 2005]
- **Tores** [Flocchini, Luccio et Song. 2006]
- **Graphe de Siperski** [Luccio, 2007]

Le nettoyage est réalisé de manière monotone, connexe et optimale (i.e., utilise $\mathbf{s}(G, v_0)$ agents).
Remarque : en asynchrone, un agent supplémentaire est nécessaire et suffisant [FLS05].

Etat de l'art 1/2

La topologie du graphe est connue à l'avance par les agents.

Nettoyage de topologies particulières

- **Arbres** [Barrière, Flocchini, Fraigniaud et Santoro. 2002]
- **Grilles** [Flocchini, Luccio et Song. 2005]
- **Hypercubes** [Flocchini, Huang et Luccio. 2005]
- **Tores** [Flocchini, Luccio et Song. 2006]
- **Graphe de Siperski** [Luccio, 2007]

Le nettoyage est réalisé de manière monotone, connexe et optimale (i.e., utilise $\mathbf{s}(G, v_0)$ agents).

Remarque : en asynchrone, un agent supplémentaire est nécessaire et suffisant [FLS05].

Etat de l'art 1/2

La topologie du graphe est connue à l'avance par les agents.

Nettoyage de topologies particulières

- **Arbres** [Barrière, Flocchini, Fraigniaud et Santoro. 2002]
- **Grilles** [Flocchini, Luccio et Song. 2005]
- **Hypercubes** [Flocchini, Huang et Luccio. 2005]
- **Tores** [Flocchini, Luccio et Song. 2006]
- **Graphe de Siperski** [Luccio, 2007]

Le nettoyage est réalisé de manière monotone, connexe et optimale (i.e., utilise $\mathbf{s}(G, v_0)$ agents).
Remarque : en asynchrone, un agent supplémentaire est nécessaire et suffisant [FLS05].

Etat de l'art 2/2

Les agents sont dans un graphe dont ils ne connaissent rien.

Nettoyage d'un graphe inconnu

Distributed chasing of network intruders
[Blin, Fraigniaud, Nisse et Vial, 2006]

Le nettoyage est réalisé de manière connexe et optimale.
En asynchrone, un agent supplémentaire est suffisant.

Problème : la stratégie n'est pas monotone et peut prendre un temps exponentiel.

Etat de l'art 2/2

Les agents sont dans un graphe dont ils ne connaissent rien.

Nettoyage d'un graphe inconnu

Distributed chasing of network intruders
[Blin, Fraigniaud, Nisse et Vial, 2006]

Le nettoyage est réalisé de manière connexe et optimale.
En asynchrone, un agent supplémentaire est suffisant.

Problème : la stratégie n'est pas monotone et peut prendre un temps exponentiel.

Problème

Une question se pose :

De quelle information doivent disposer les agents pour qu'il existe un protocole réparti leur permettant de nettoyer tout graphe de manière monotone, connexe et optimale ?

Quel type de connaissance ?

Information qualitative

- Topologie, taille, diamètre du réseau ...

Information quantitative : le conseil [Fraigniaud *et al.* 06]

- Mesure le **plus petit nombre de bits d'information** permettant de résoudre **efficacement** un problème réparti.

Problème

Une question se pose :

De quelle information doivent disposer les agents pour qu'il existe un protocole réparti leur permettant de nettoyer tout graphe de manière monotone, connexe et optimale ?

Quel type de connaissance ?

Information qualitative

- Topologie, taille, diamètre du réseau ...

Information quantitative : le conseil [Fraigniaud *et al.* 06]

- Mesure le **plus petit nombre de bits d'information** permettant de résoudre **efficacement** un problème réparti.

Problème

Une question se pose :

De quelle information doivent disposer les agents pour qu'il existe un protocole réparti leur permettant de nettoyer tout graphe de manière monotone, connexe et optimale ?

Quel type de connaissance ?

Information qualitative

- Topologie, taille, diamètre du réseau ...

Information quantitative : le conseil [Fraigniaud *et al.* 06]

- Mesure le **plus petit nombre de bits d'information** permettant de résoudre **efficacement** un problème réparti.

Conseil, taille de conseil [Fraigniaud *et al.* 06]

Un problème réparti $\mathcal{P}$ à résoudre
G un graphe, instance de $\mathcal{P}$.
Conseil : information à utiliser pour résoudre $\mathcal{P}$ *efficacement*

Modélisation de l'information fournie

- Un oracle $\mathcal{O}$ associe à chaque instance G du problème une chaîne de bits $\mathcal{O}(G)$ qui est distribuée sur les sommets de G.
- taille de conseil $max_G|\mathcal{O}(G)|$

Exemples

- réveil (nombre linéaire de messages) : $\Theta(n \log n)$ bits ;
- diffusion (nombre linéaire de messages) : $O(n)$ bits ;
- exploration d'arbres, MST, coloriage de graphe ...

Conseil, taille de conseil [Fraigniaud *et al.* 06]

Un problème réparti $\mathcal{P}$ à résoudre
G un graphe, instance de $\mathcal{P}$.
Conseil : information à utiliser pour résoudre $\mathcal{P}$ *efficacement*

Modélisation de l'information fournie

- Un oracle $\mathcal{O}$ associe à chaque instance G du problème une chaîne de bits $\mathcal{O}(G)$ qui est distribuée sur les sommets de G.
- taille de conseil $max_G|\mathcal{O}(G)|$

Exemples

- réveil (nombre linéaire de messages) : $\Theta(n \log n)$ bits ;
- diffusion (nombre linéaire de messages) : $O(n)$ bits ;
- exploration d'arbres, MST, coloriage de graphe ...

Conseil, taille de conseil [Fraigniaud *et al.* 06]

Nettoyer un graphe de façon répartie connexe et optimale.
Instance : G un graphe et une base $v_0 \in V(G)$.
Conseil : pour que le nettoyage soit monotone.

Modélisation de l'information fournie

- Un oracle $\mathcal{O}$ distribue une chaîne de bits $\mathcal{O}(G, v_0)$ sur les sommets.
- taille de conseil $max_{G,v_0}|\mathcal{O}(G, v_0)|$

Question :
Quelle est la taille de conseil nécessaire et suffisante telle qu'il existe un protocole réparti qui résolve le problème du nettoyage réparti *efficacement* (i.e., monotone connexe et optimale) ?

Nos résultats

Borne supérieure

$O(n \log n)$ bits de conseil sont suffisants pour résoudre le problème de l'encerclement réparti.

Nous proposons un oracle $\mathcal{O}$ de taille $O(n \log n)$ bits et un protocole réparti $\mathcal{P}$ utilisant $\mathcal{O}$ qui nettoie tout graphe de manière monotone connexe et optimale.

Borne inférieure

$\Omega(n \log n)$ bits de conseil sont nécessaires pour résoudre le problème de l'encerclement réparti.

Nous exhibons une classe de graphes $\mathcal{C}$ telle que tout protocole réparti $\mathcal{P}$ qui utilise un conseil de taille inférieure à $\Omega(n \log n)$ ne peut nettoyer tous les graphes de $\mathcal{C}$.

Nos résultats

Borne supérieure

$O(n \log n)$ bits de conseil sont suffisants pour résoudre le problème de l'encerclement réparti.

Nous proposons un oracle $\mathcal{O}$ de taille $O(n \log n)$ bits et un protocole réparti $\mathcal{P}$ utilisant $\mathcal{O}$ qui nettoie tout graphe de manière monotone connexe et optimale.

Borne inférieure

$\Omega(n \log n)$ bits de conseil sont nécessaires pour résoudre le problème de l'encerclement réparti.

Nous exhibons une classe de graphes $\mathcal{C}$ telle que tout protocole réparti $\mathcal{P}$ qui utilise un conseil de taille inférieure à $\Omega(n \log n)$ ne peut nettoyer tous les graphes de $\mathcal{C}$.

Idée de la borne supérieure : $O(n \log n)$

Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.

Idée de la borne supérieure : $O(n \log n)$

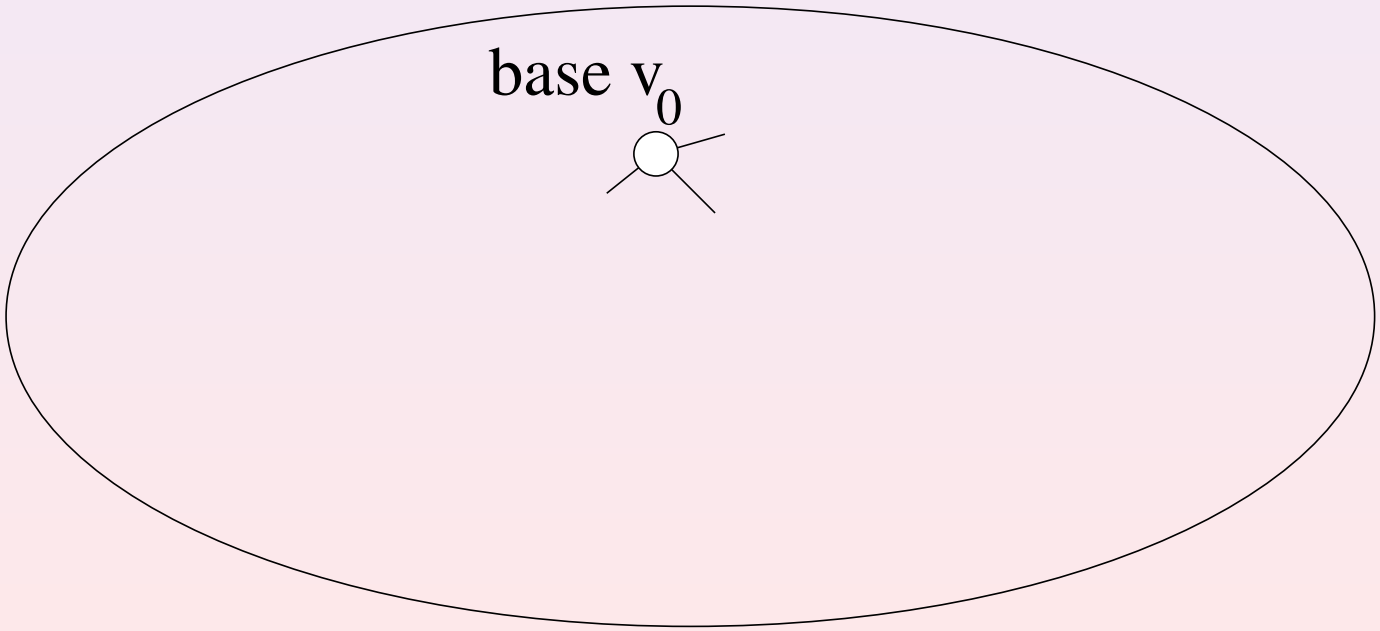
Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.

Idée de la borne supérieure : $O(n \log n)$

Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

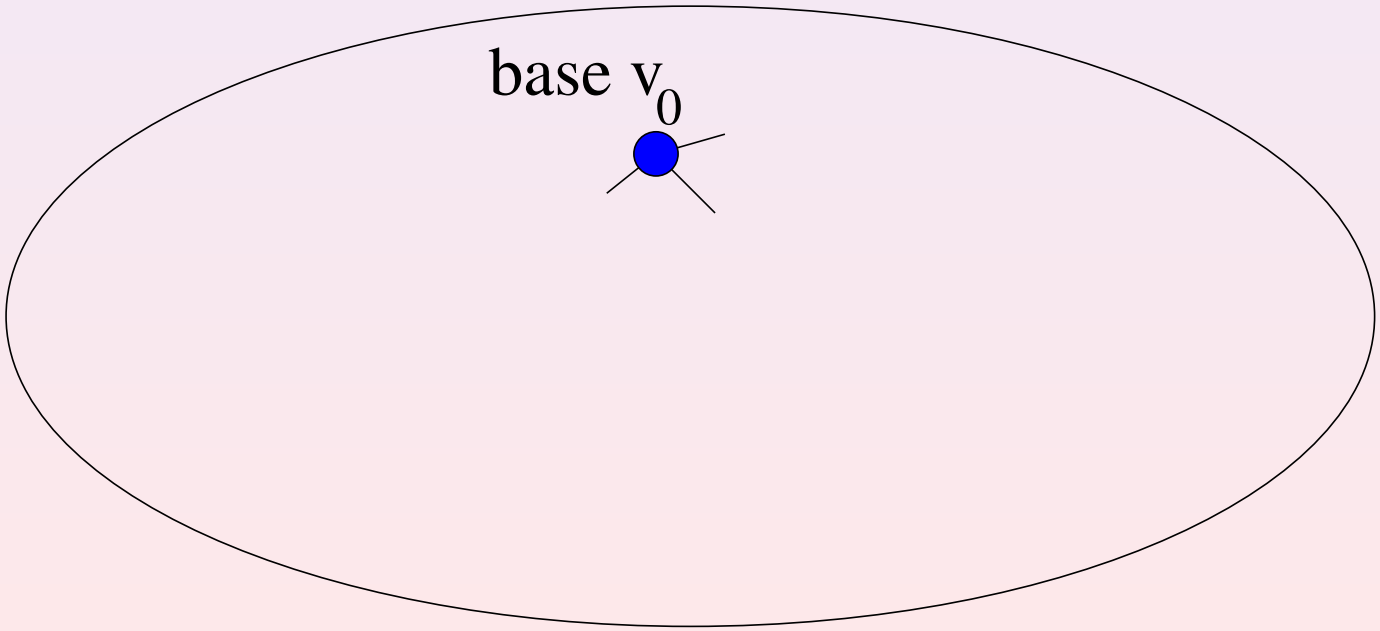
$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.



Idée de la borne supérieure : $O(n \log n)$

Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

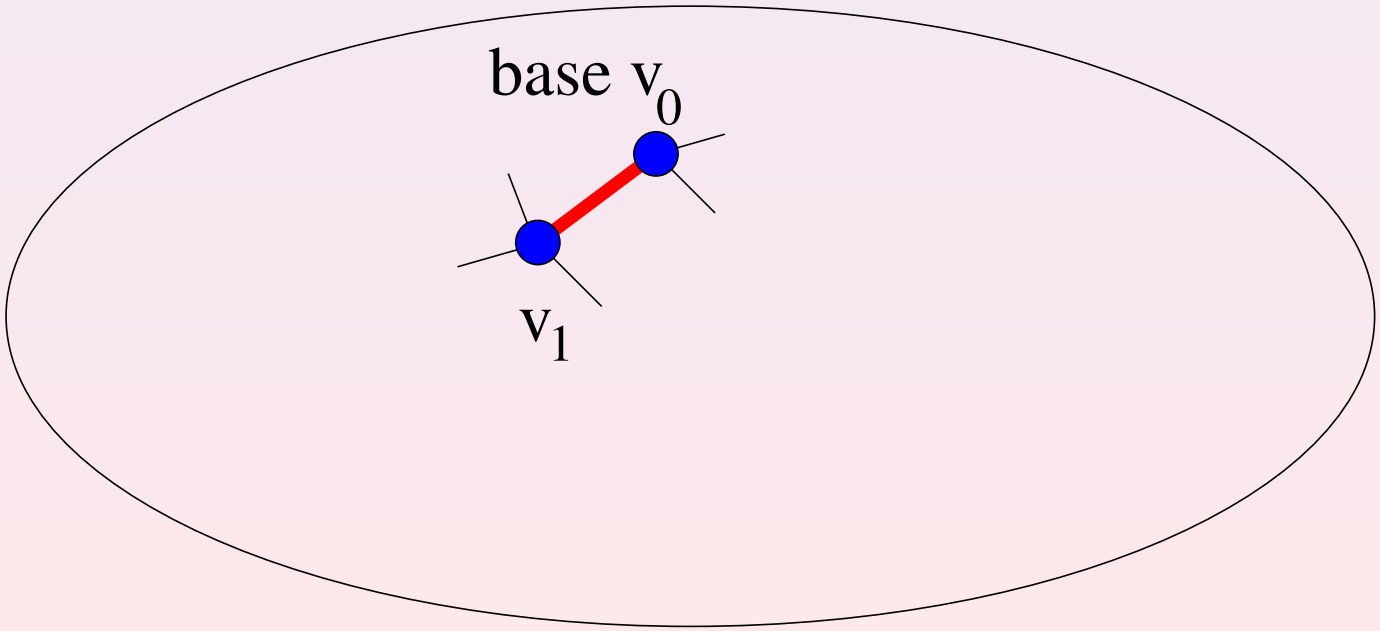
$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.



Idée de la borne supérieure : $O(n \log n)$

Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

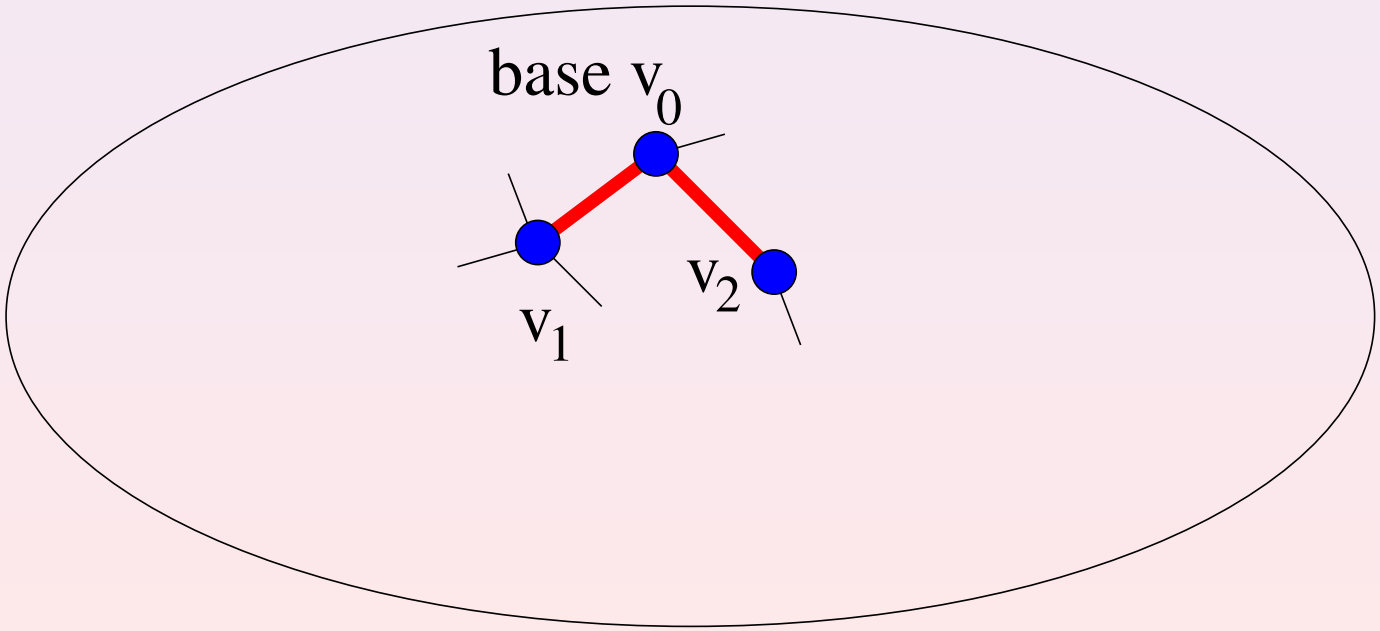
$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.



Idée de la borne supérieure : $O(n \log n)$

Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

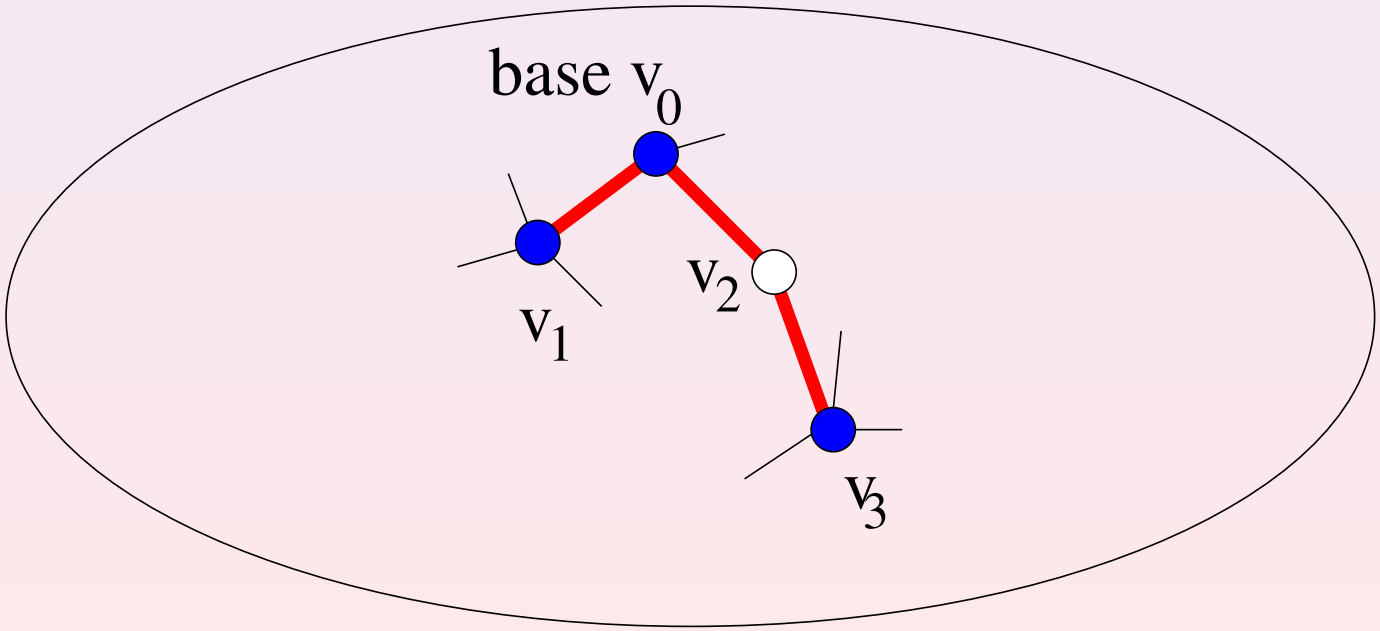
$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.



Idée de la borne supérieure : $O(n \log n)$

Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

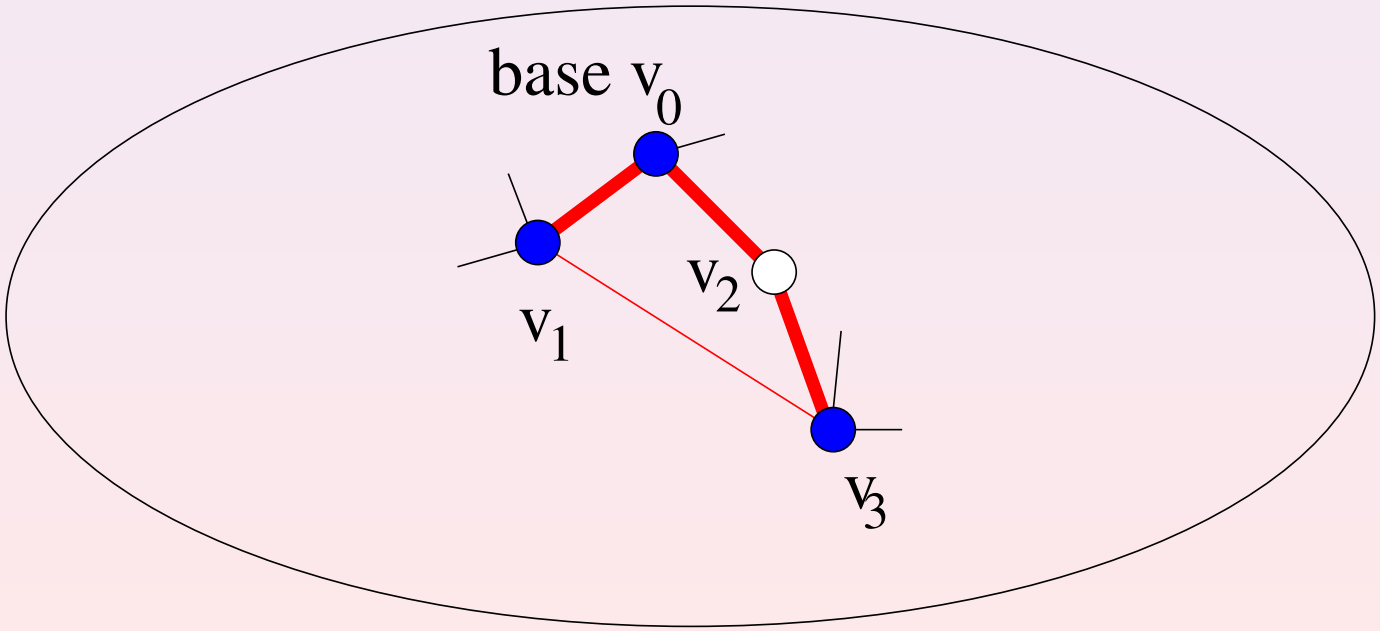
$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.



Idée de la borne supérieure : $O(n \log n)$

Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

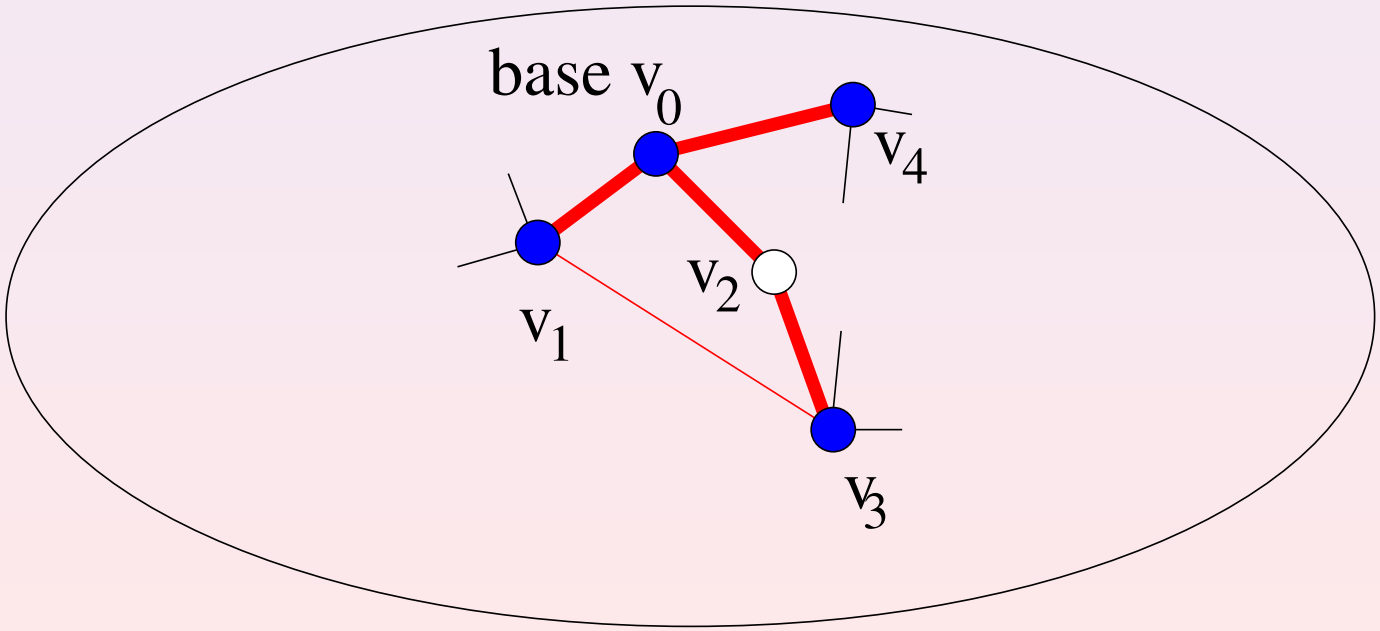
$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.



Idée de la borne supérieure : $O(n \log n)$

Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

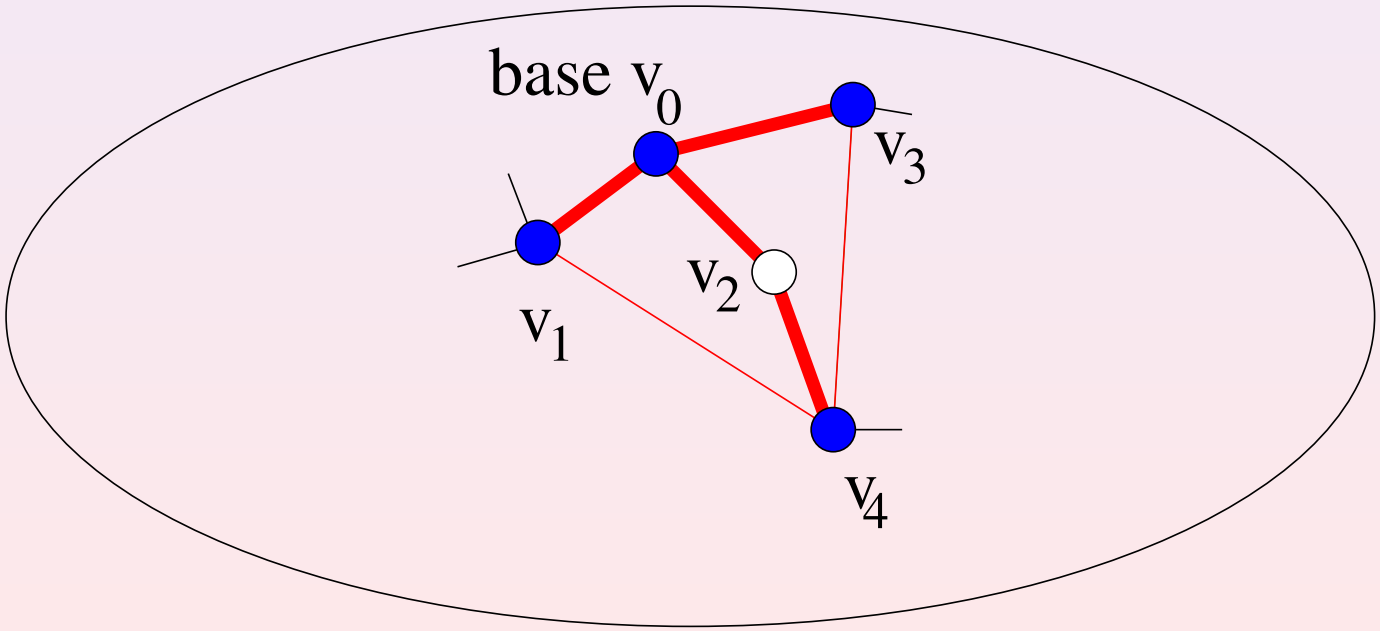
$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.



Idée de la borne supérieure : $O(n \log n)$

Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

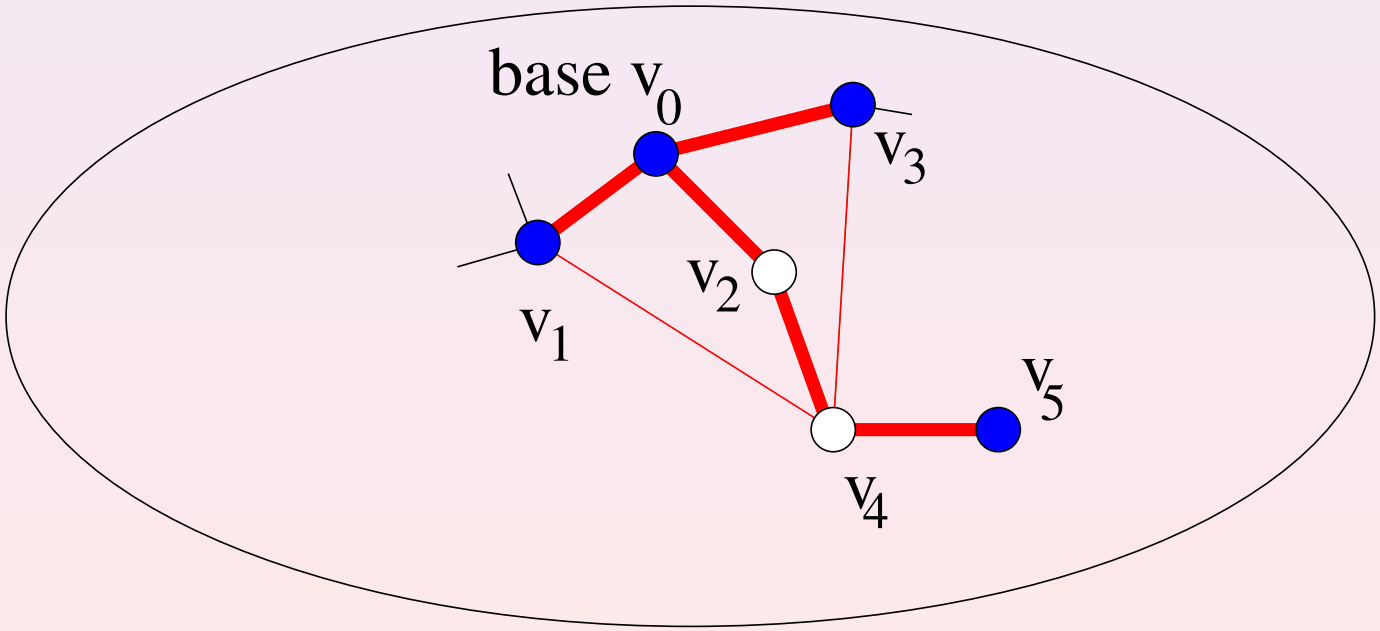
$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.



Idée de la borne supérieure : $O(n \log n)$

Soit G un graphe, $v_0 \in V(G)$
S une stratégie monotone connexe optimale pour G.
Notre oracle "code" S sur les sommets de G.

$S \longrightarrow$ ordre de nettoyage des sommets $\{v_0, v_1, \cdots, v_{n-1}\}$,
et des arbres $T_0 \subset \cdots \subset T_{n-1}$ avec T_i couvre $\{v_0, \cdots, v_i\}$.



Idée de la borne supérieure : l'Oracle

Protocole divisé en **n+1 phases**.
Pour chaque sommet v_i, l'oracle distingue 3 types d'arêtes :

1. celles qui appartiennent à l'arbre couvrant T_n (l'oracle fournit les fils de v_i correspondants) ;
2. l'arête par lequel il faudra le quitter ;
3. les autres.

L'oracle fournit également 2 numéros de phase : **la** phase de nettoyage des arêtes de type 3 et la phase à laquelle le sommet v_i peut être abandonné.

Au total : codage d'un arbre couvrant + 2 numéros de phase par sommet $= O(n \log n)$ bits de conseil.

Idée de la borne supérieure : le Protocole

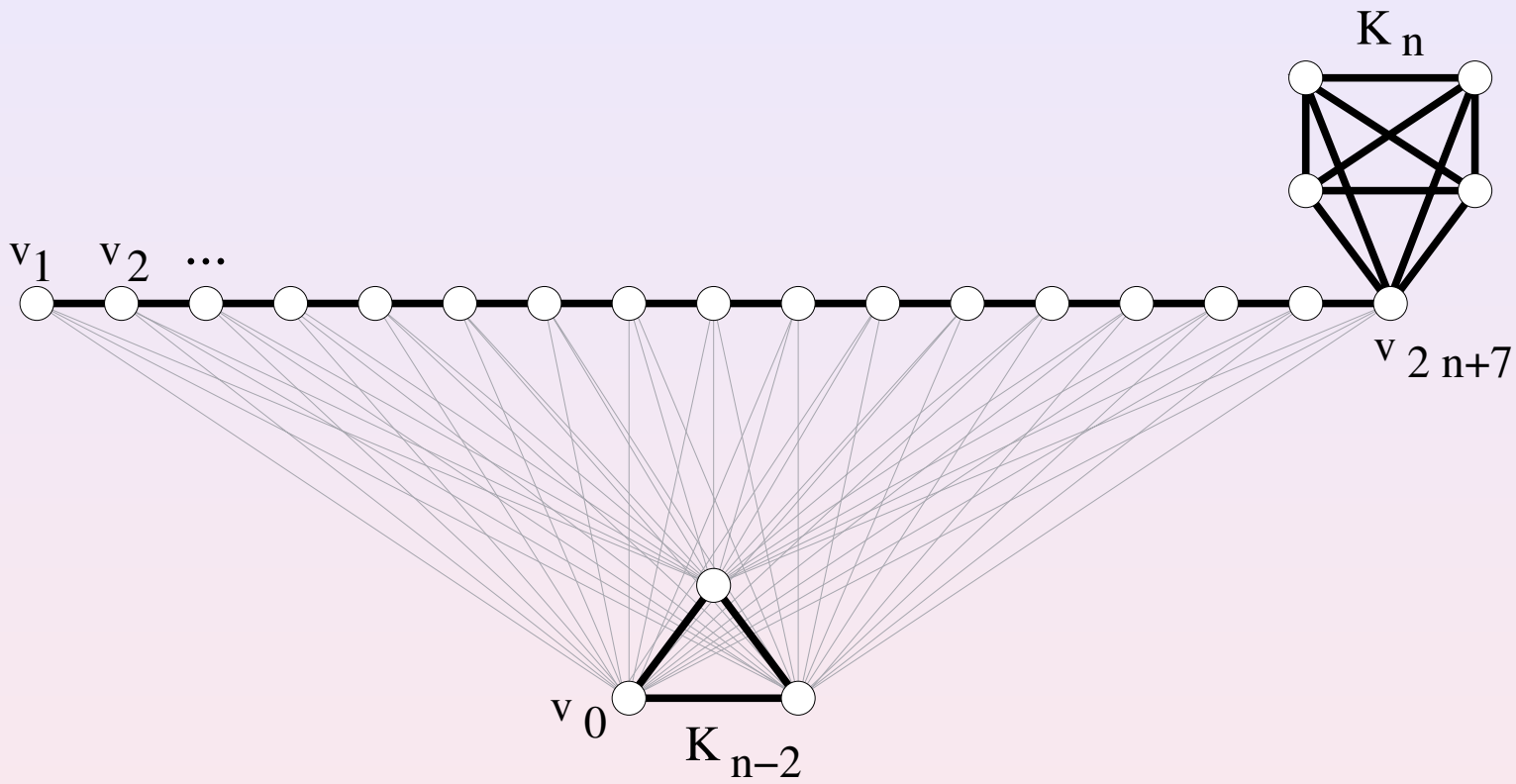
Phase i du protocole $(0 \leq i \leq n)$:
Prérequis :

- T_i est propre + éventuellement d'autres arêtes entre les sommets $\{v_0, \cdots, v_i\}$.
- Un agent garde les sommets de $\{v_0, \cdots, v_i\}$ incident à une arête contaminée.

Idée du Protocole :

1. Tout agent libre parcourt T_i.
2. Si il rencontre un sommet dont i est la phase de nettoyage des arêtes de type 3, il les nettoie.
3. Pour finir la phase, l'arête qui permet d'atteindre v_{i+1} est découverte et nettoyée.

La borne inférieure : $\Omega(n \log n)$



Classe de graphes $(G_n)_{n \in N}$ (G_5 est représenté).
La structure de toute stratégie monotone connexe et optimale est très contrainte.

La borne inférieure

Soit G_n le graphe précédent.
L l'ensemble des orientations locales pour G_n

Lemme 1

Soit f une chaîne de bits de conseil fixée. Soit $\mathcal{P}$ un protocole qui résout le protocole de l'encerclement réparti. $\mathcal{P}$ peut nettoyer au plus $A = |L|(\frac{1}{n-2})^n$ instances de L.

Lemme 2 [Fraigniaud *et al.* 06]

Un oracle $\mathcal{O}$ de taille q peut fournir au plus $t = (q+1)2^q C_{n+q}^n$ chaîne de bits de conseil différentes.

La borne inférieure

- Lemme 1 : $\mathcal{P}$ ne peut nettoyer plus de A instances de L.
- Lemme 2 : Un oracle $\mathcal{O}$ de taille q fournit au plus t chaînes.

D'après le lemme 2 :

Il existe au moins $B = |L|/t$ instances qui ont le même conseil.

Or, si $q = \alpha n \log n$ et $\alpha < 1/4$, $B > A$ asymptotiquement.

Donc, $\mathcal{P}$ ne peut pas nettoyer toutes les instances de L en utilisant un oracle de taille $\Omega(n \log n)$.

Perspectives

Quelle est la quantité de conseil minimum qui doit être distribuée **sur chaque sommet** pour résoudre le problème de l'encerclement réparti ?

Comment **relacher les contraintes** sur la stratégie pour diminuer la quantité de conseil ?
Par exemple, quelle est la quantité de conseil minimum pour nettoyer un graphe **en temps polynomial** (avec une stratégie pas forcement monotone) ?