Algorithmes d'énumération de réparations de graphes

Journées Graphes et Algorithmes 2021

Aurélie Lagoutte
LIMOS, Université Clermont Auvergne

Travaux réalisés en collaboration avec
Caroline Brosse, Vincent Limouzy, Arnaud Mary, Lucas Pastor

Enumération : principe

Certains problèmes nécessitent en réponse une **liste** de solution, plutôt qu'**une** solution. Par exemple :

Enumération : principe

Certains problèmes nécessitent en réponse une **liste** de solution, plutôt qu'**une** solution. Par exemple :

- Réponse à une requête de base de données

```
$ select appellation, vignoble, type from AOC
Côte-Rôtie                  |   Vallée du Rhône   |   Rouge
Saint-Emilion               |   Bordeaux          |   Rouge
Saint-Nicolas-de-Bourgueil  |   Val de Loire      |   Rouge
```

Enumération : principe

Certains problèmes nécessitent en réponse une **liste** de solution, plutôt qu'**une** solution. Par exemple :

- Réponse à une requête de base de données

```
$ select appellation, vignoble, type from AOC
Côte-Rôtie                  |  Vallée du Rhône  |  Rouge
Saint-Emilion               |  Bordeaux         |  Rouge
Saint-Nicolas-de-Bourgueil  |  Val de Loire     |  Rouge
```

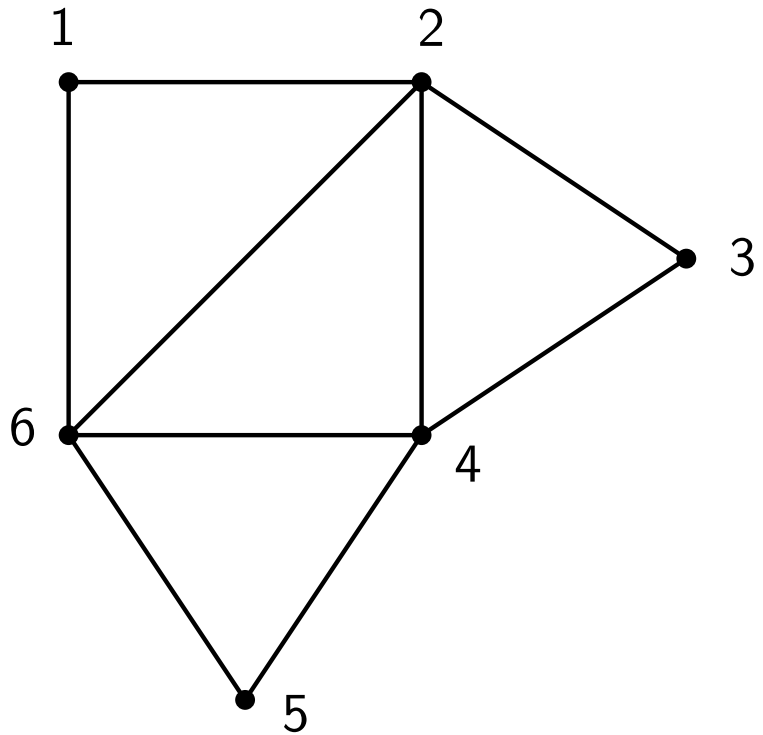
- Table de vérité : lister toutes les combinaisons d'entrées

e_1	e_2	e_3	S
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Énumération : un exemple typique

Entrée: Graphe G

Sortie : La liste de tous les stables maxim**aux** (par inclusion) de G

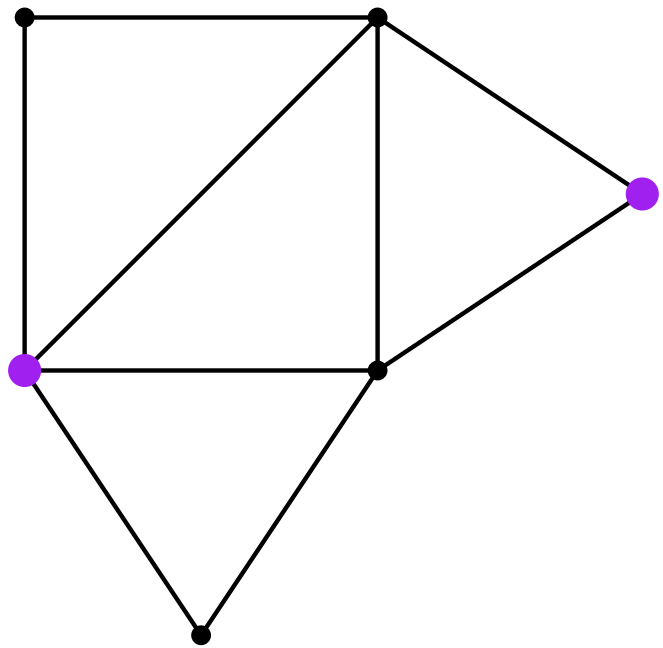


$\{1, 3, 5\}, \{1, 4\}, \{2, 5\}, \{3, 6\}$

Se restreindre aux problèmes faciles

Entrée: Graphe G
Sortie : un stable **maximal par inclusion**.

$\in$ P (glouton)



A ne pas confondre :

Entrée: Graphe G
Sortie : un stable de taille maximum

NP-complet

Enumération dans les graphes : cas d'usage

- Bases de données orientée graphes : réponse à une requête
- Modélisation par le graphe inexacte : certaines solutions sont *meilleures* par des critères qualitatifs, il faut les examiner une par une
- Identifier tous les motifs problématiques (ou intéressants!) dans un réseau

Domaines d'applications: bioinformatique (arbres phylogénétiques), chimie (structure de molécules), modélisation de systèmes complexes, bases de données...

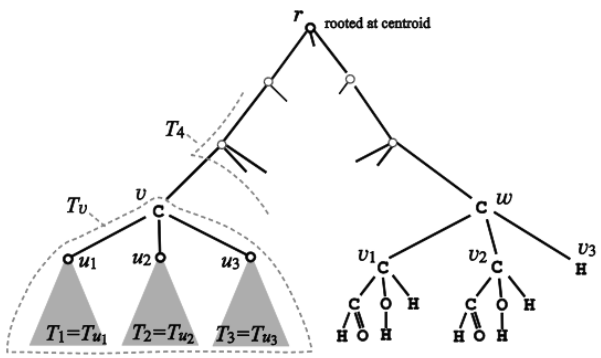


Schéma d'un stéréoisomère[1]

[1] Comparison and Enumeration of Chemical Graphs, T. Akutsu, H. Nagamochi, *Comp. and Struct. Biotechnology Journal*

Complexité de l'énumération

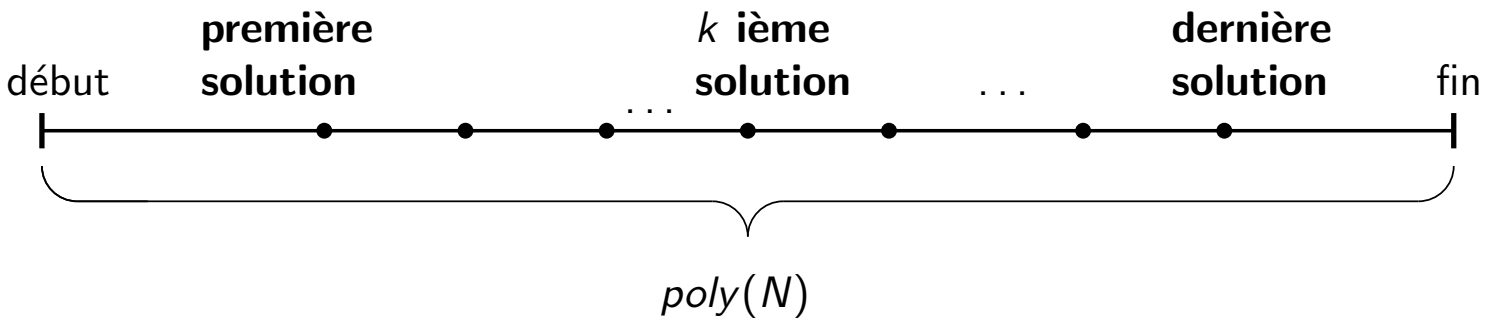
En général: nombre exponentiel de solutions à lister
(ex: $3^{n/3}$ stables max.)
$\Rightarrow$ *Bonne mesure de complexité?*

Complexité de l'énumération

En général: nombre exponentiel de solutions à lister
(ex: $3^{n/3}$ stables max.)
$\Rightarrow$ *Bonne mesure de complexité?*

1. Output-polynomial

Entrée de taille n, N solutions à générer.

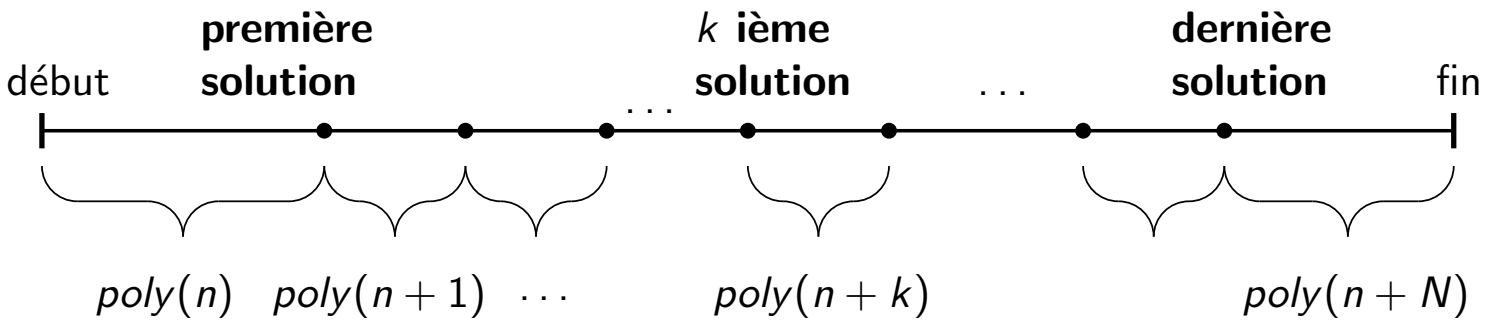


Complexité de l'énumération

En général: nombre exponentiel de solutions à lister
(ex: $3^{n/3}$ stables max.)
$\Rightarrow$ *Bonne mesure de complexité?*

1. Output-polynomial
2. Incrémental polynomial

Entrée de taille n, N solutions à générer.

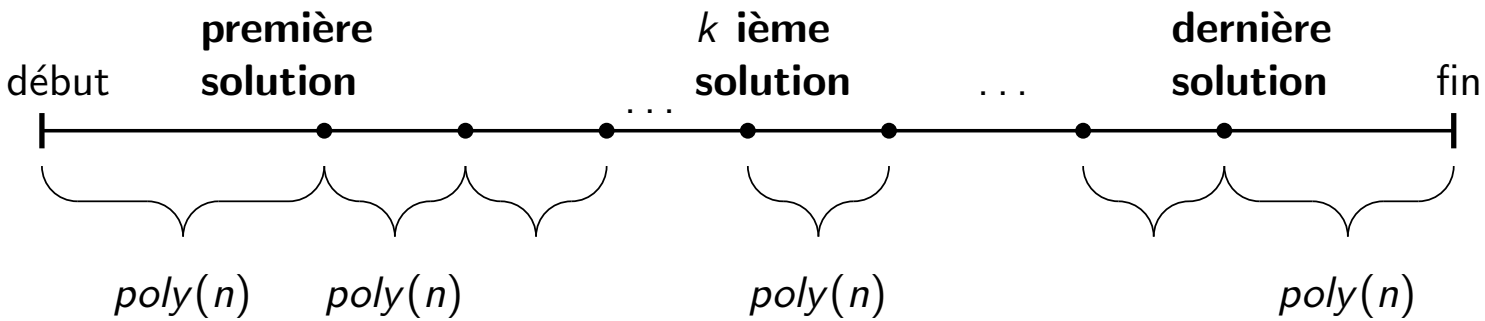


Complexité de l'énumération

En général: nombre exponentiel de solutions à lister
(ex: $3^{n/3}$ stables max.)
$\Rightarrow$ *Bonne mesure de complexité?*

1. Output-polynomial
2. Incrémental polynomial
3. À délai polynomial

Entrée de taille n, N solutions à générer.



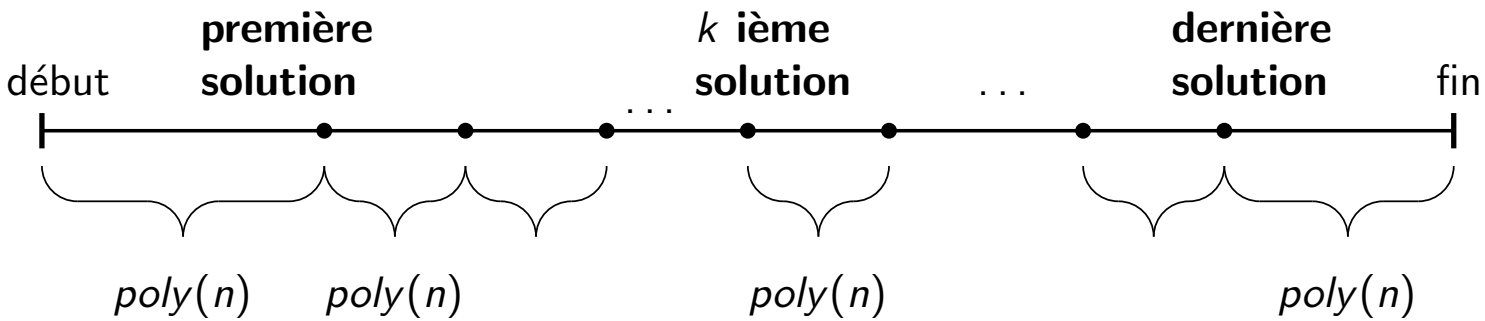
Complexité de l'énumération

En général: nombre exponentiel de solutions à lister
(ex: $3^{n/3}$ stables max.)
$\Rightarrow$ *Bonne mesure de complexité?*

1. Output-polynomial
2. Incrémental polynomial
3. À délai polynomial

Espace poly vs. espace exponentiel

Entrée de taille n, N solutions à générer.



Objets intéressants à énumérer

1. Transversaux minim**aux** d'un hypergraphe

Objets intéressants à énumérer

1. Transversaux minim**aux** d'un hypergraphe
2. Dominants minim**aux**

Objets intéressants à énumérer

1. Transversaux minim**aux** d'un hypergraphe
2. Dominants minim**aux**
3. Arbres couvrants

Objets intéressants à énumérer

1. Transversaux minim**aux** d'un hypergraphe
2. Dominants minim**aux**
3. Arbres couvrants
4. **"Motifs" structurés**: stables max., cliques max., ...

Objets intéressants à énumérer

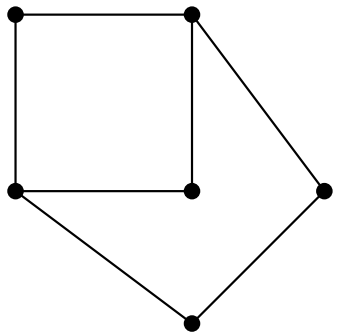
1. Transversaux minim**aux** d'un hypergraphe
2. Dominants minim**aux**
3. Arbres couvrants
4. **"Motifs" structurés**: stables max., cliques max., ...
5. Π-**"Réparations" minimales** du graphe
   $\rightarrow$ Complétion, déletion, sous-graphe induit, ...
   ... satisfaisant une propriété donnée Π

Réparations minimales

3 variantes

On veut satisfaire une propriété Π

Exemple: $\Pi =$ sans C_4 induit

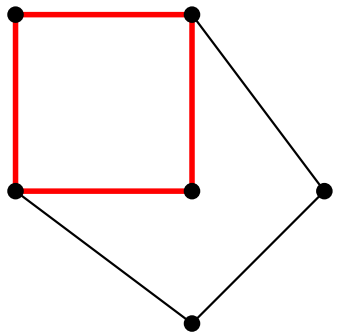


Réparations minimales

3 variantes

On veut satisfaire une propriété Π

Exemple: $\Pi =$ sans C_4 induit

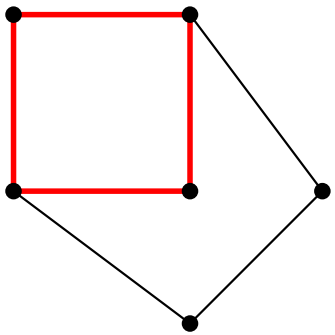


Réparations minimales

3 variantes

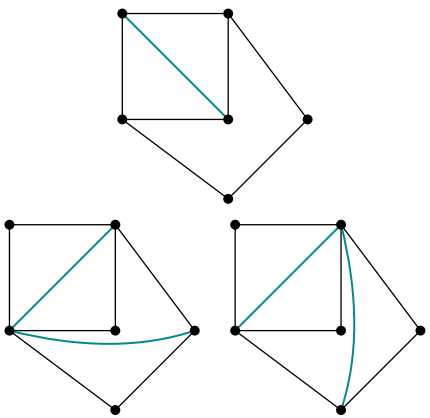
On veut satisfaire une propriété Π

Exemple: $\Pi =$ sans C_4 induit



Réparation par ajout d'arêtes

Π-**Complétion min.**

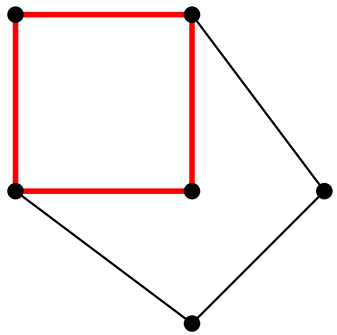


Réparations minimales

3 variantes

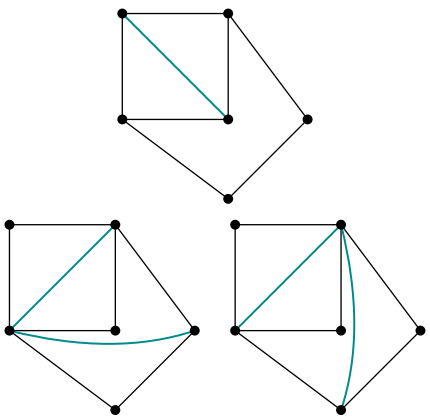
On veut satisfaire une propriété Π

Exemple: $\Pi =$ sans C_4 induit



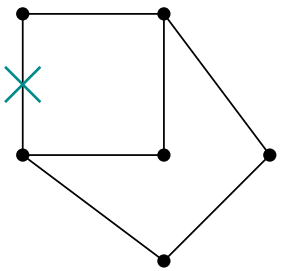
Réparation par ajout d'arêtes

Π-**Complétion min.**



Réparation par suppression d'arêtes

Π-**Délétion min.**



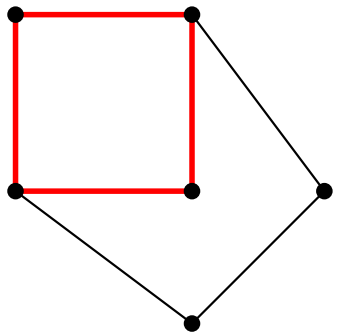
$+$ 3 autres

Réparations minimales

3 variantes

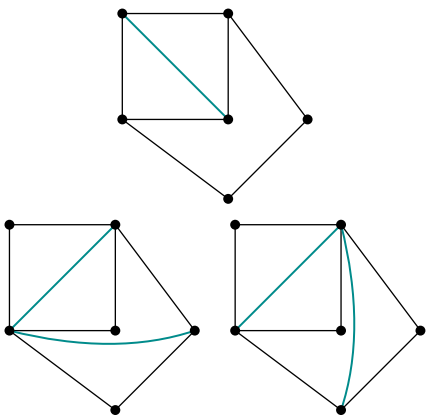
On veut satisfaire une propriété Π

Exemple: $\Pi =$ sans C_4 induit



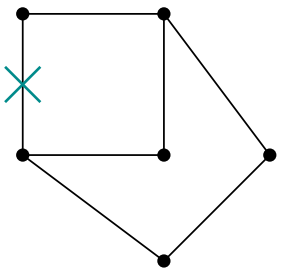
Réparation par ajout d'arêtes

Π-**Complétion min.**



Réparation par suppression d'arêtes

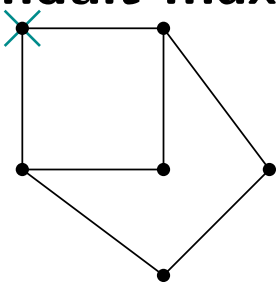
Π-**Délétion min.**



+ 3 autres

Réparation par suppr. de sommets

Π-**Sous-graphe induit max.**



+ 3 autres

Méthodes algorithmiques pour l'énumération

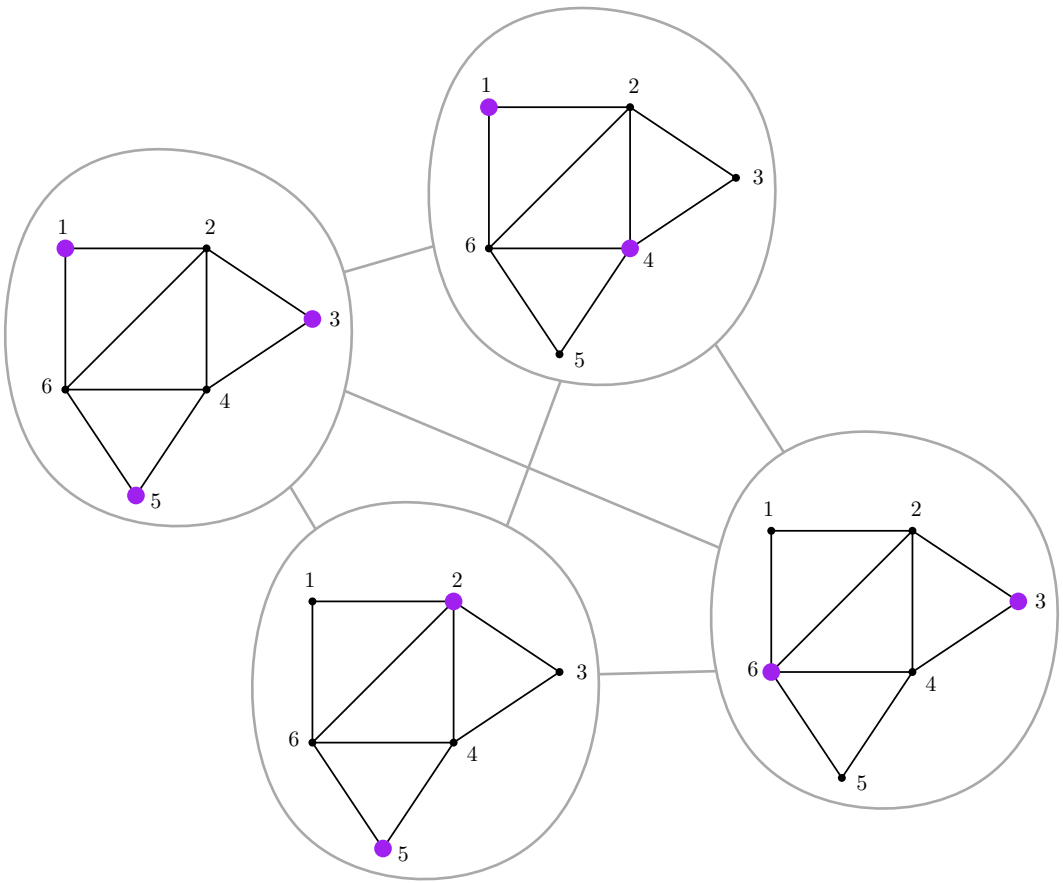
(Un petit quiz avant d'enchaîner ? `http://wooclap.com/JGA21`)

Principe d'un algo. d'énumération

- Parcourir l'espace des solutions

Exemple avec les stables maximaux

*(meta-)*Graphe des solutions

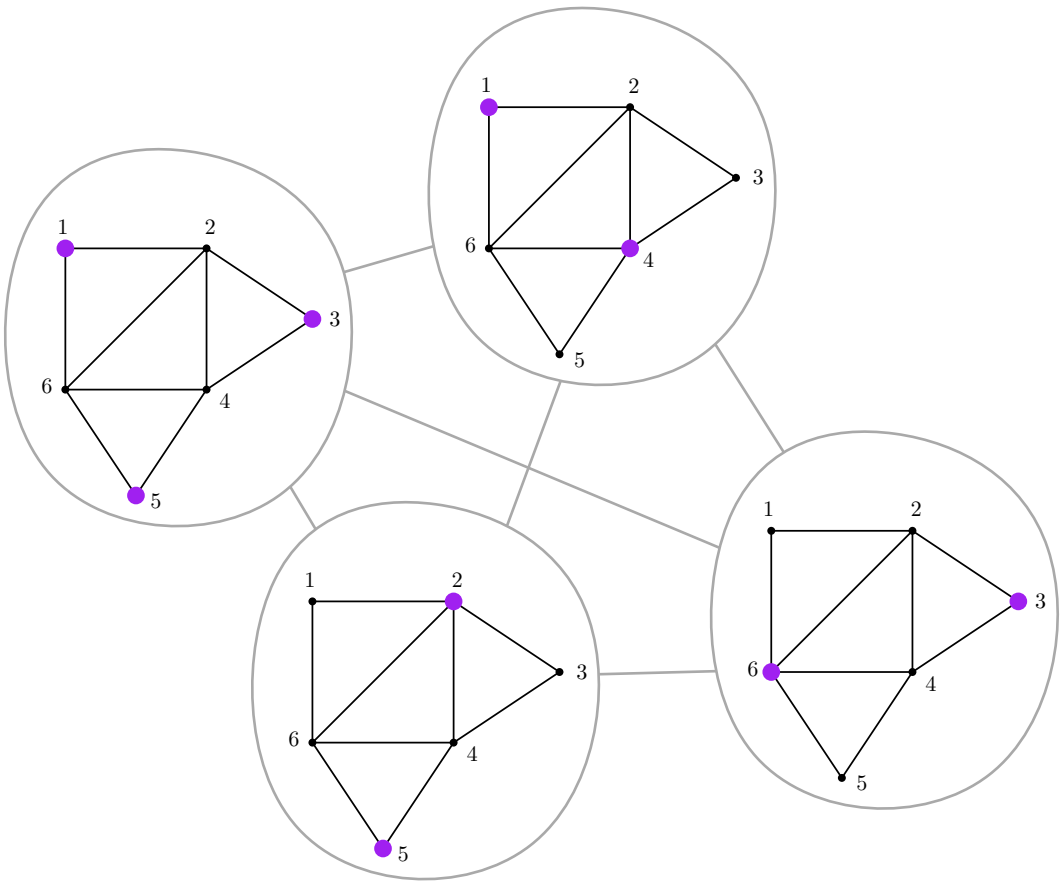


Principe d'un algo. d'énumération

- Parcourir l'espace des solutions
- en écrivant sur la sortie chaque solution une fois

Exemple avec les stables maximaux

*(meta-)*Graphe des solutions

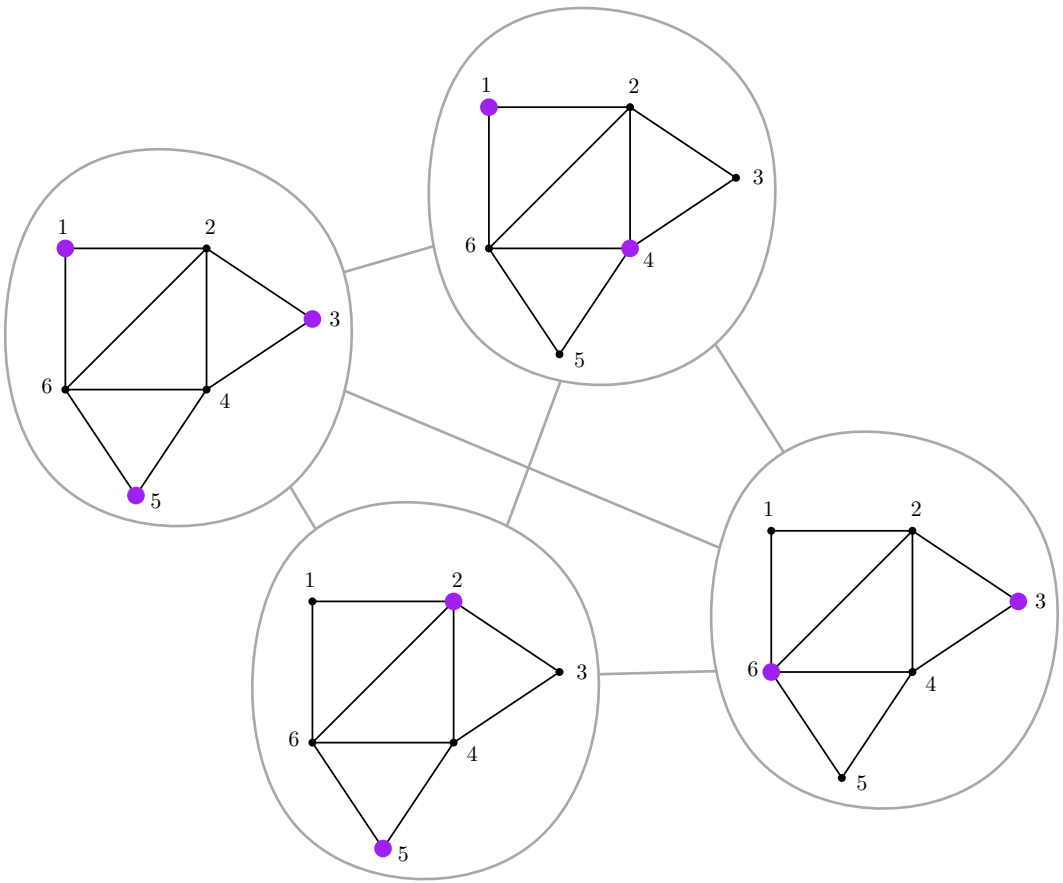


Principe d'un algo. d'énumération

- Parcourir l'espace des solutions
- en écrivant sur la sortie chaque solution une fois
- et **une seule** fois

Exemple avec les stables maximaux

*(meta-)*Graphe des solutions

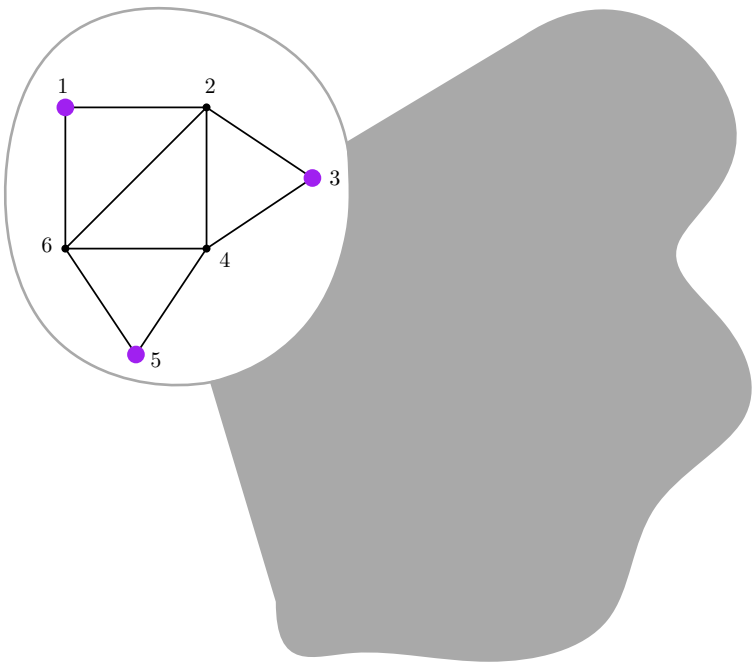


Principe d'un algo. d'énumération

- Parcourir l'espace des solutions
- en écrivant sur la sortie chaque solution une fois
- et **une seule** fois

Exemple avec les stables maximaux

*(meta-)*Graphe des solutions



Trois méthodes classiques:

But : avoir un délai polynomial
+ espace poly pour 1 et 2 (et parfois 3)

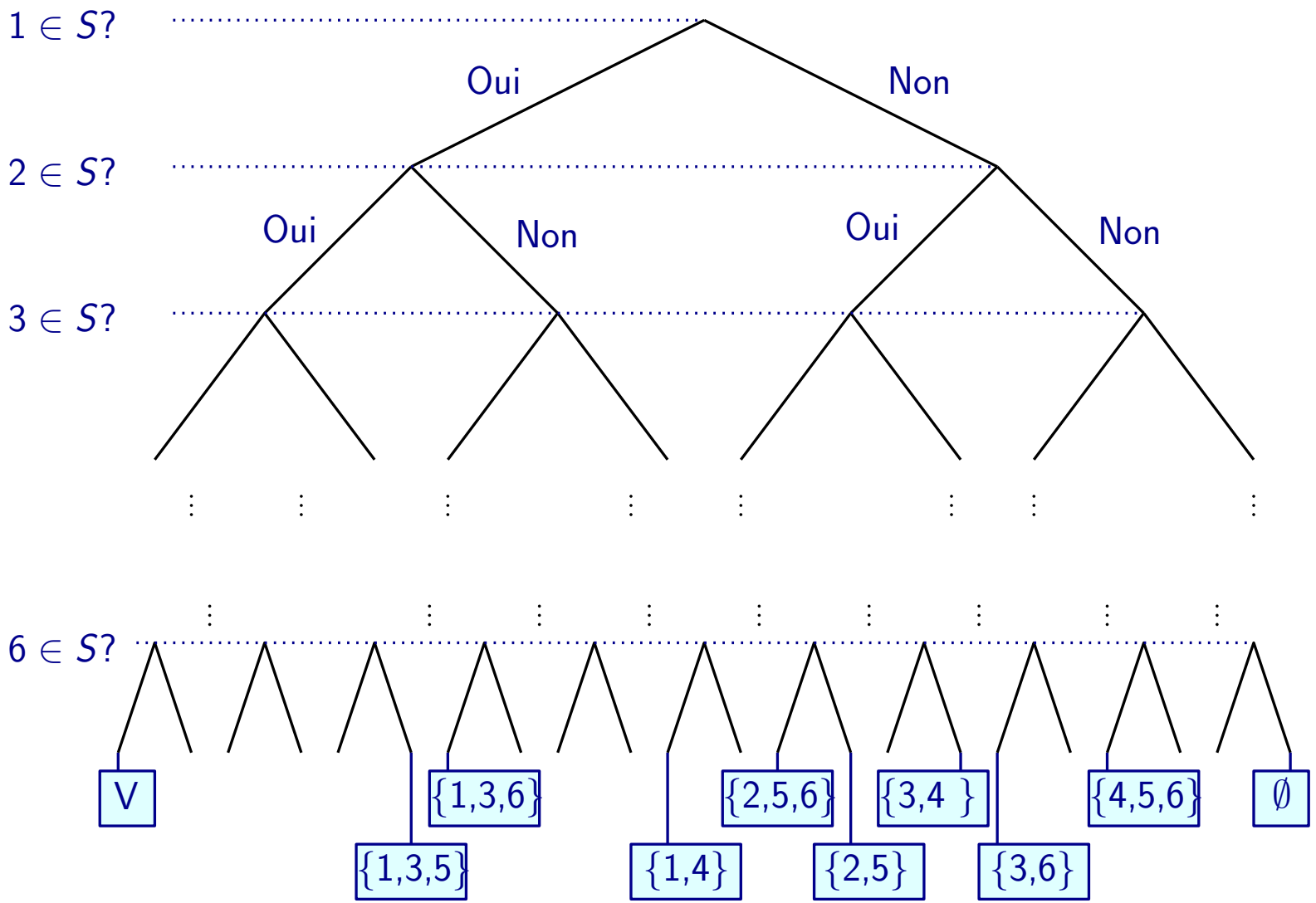
1. *Flashlight search* ou *Binary partition*
   [Read, Tarjan '75]
2. *Reverse search*
   [Avis, Fukuda '96]
3. *Proximity Search*
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

Trois méthodes classiques:

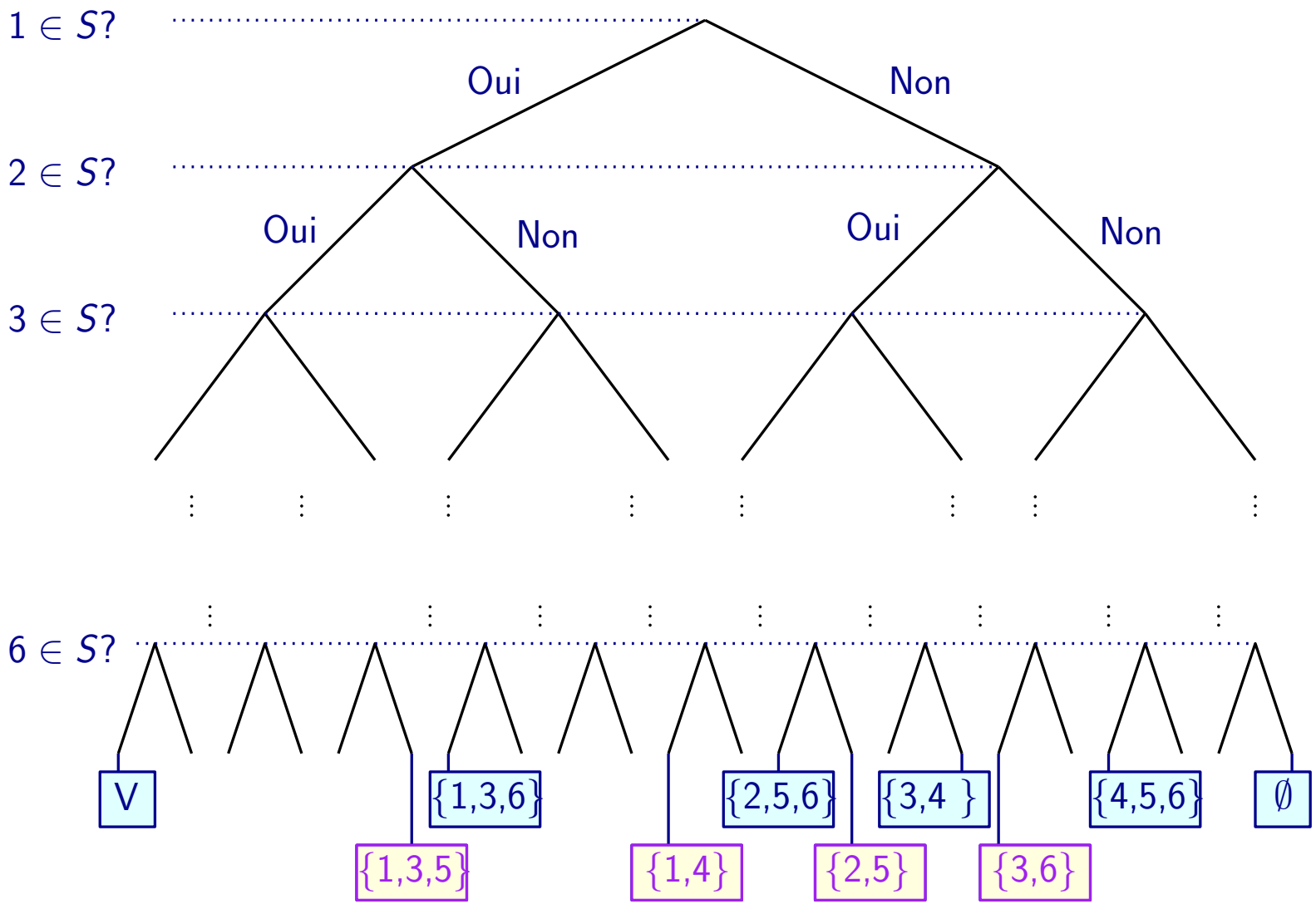
But : avoir un délai polynomial
+ espace poly pour 1 et 2 (et parfois 3)

1. ***Flashlight search*** ou *Binary partition*
   [Read, Tarjan '75]
2. *Reverse search*
   [Avis, Fukuda '96]
3. *Proximity Search*
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

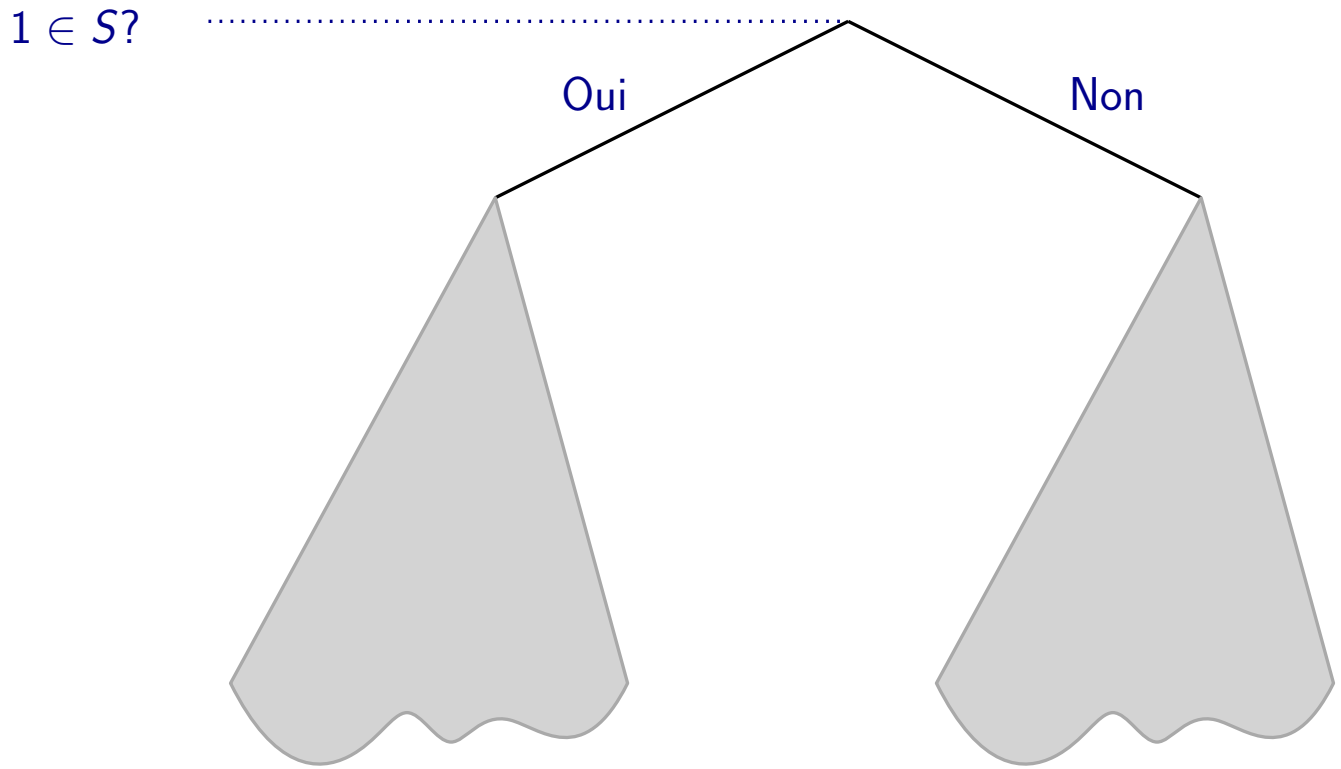
Flashlight search ou *Binary partition*



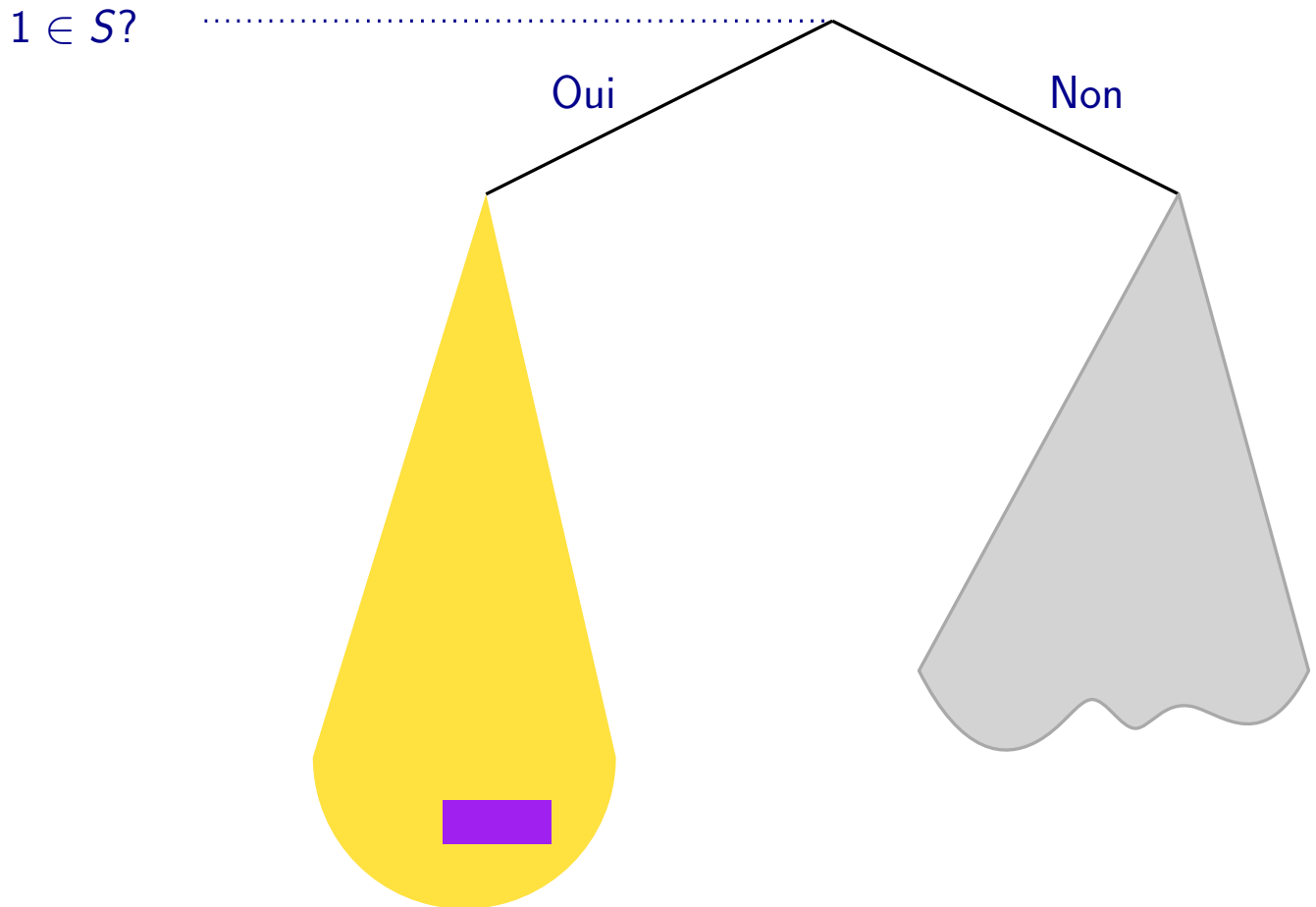
Flashlight search ou *Binary partition*



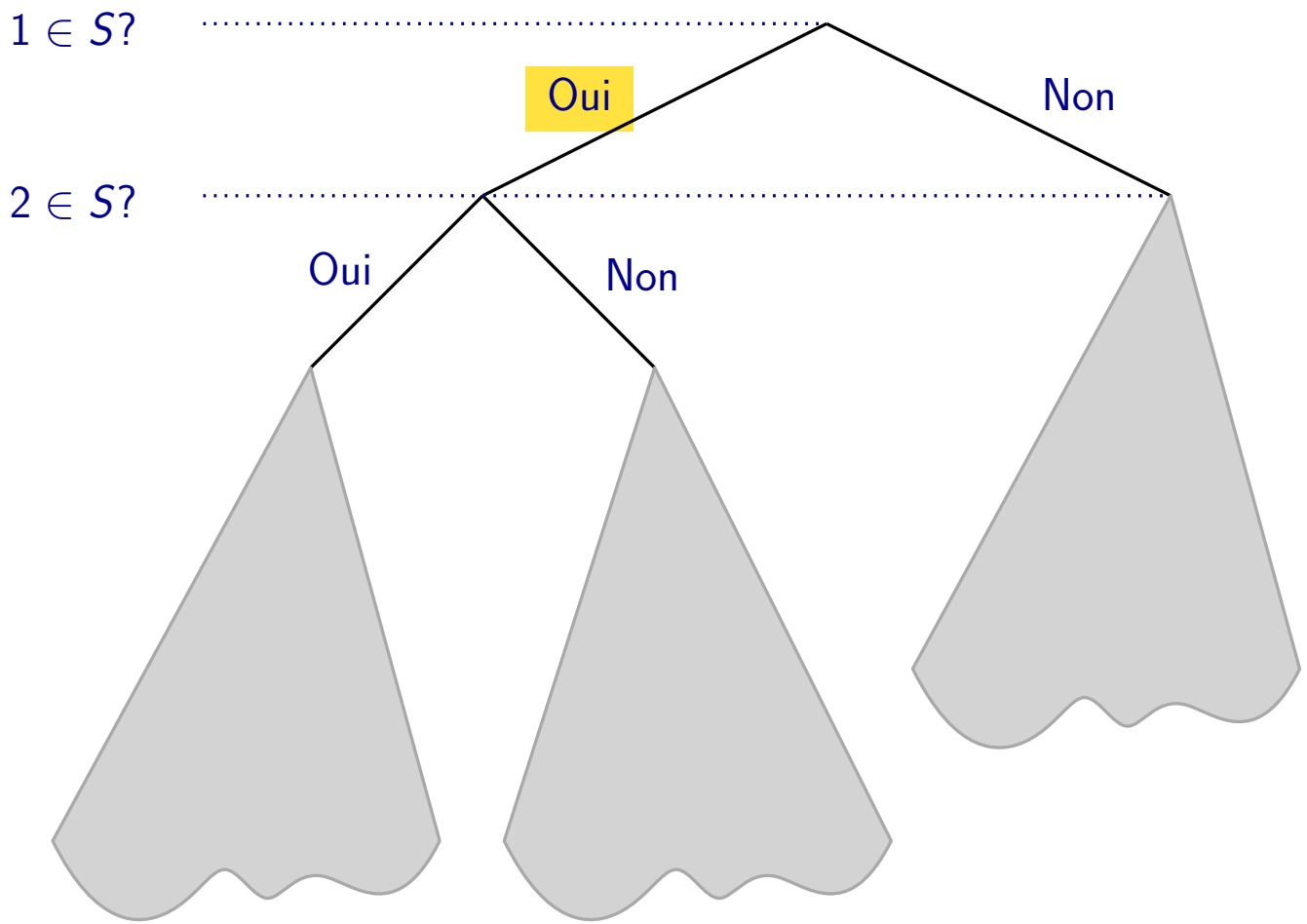
Flashlight search ou *Binary partition*



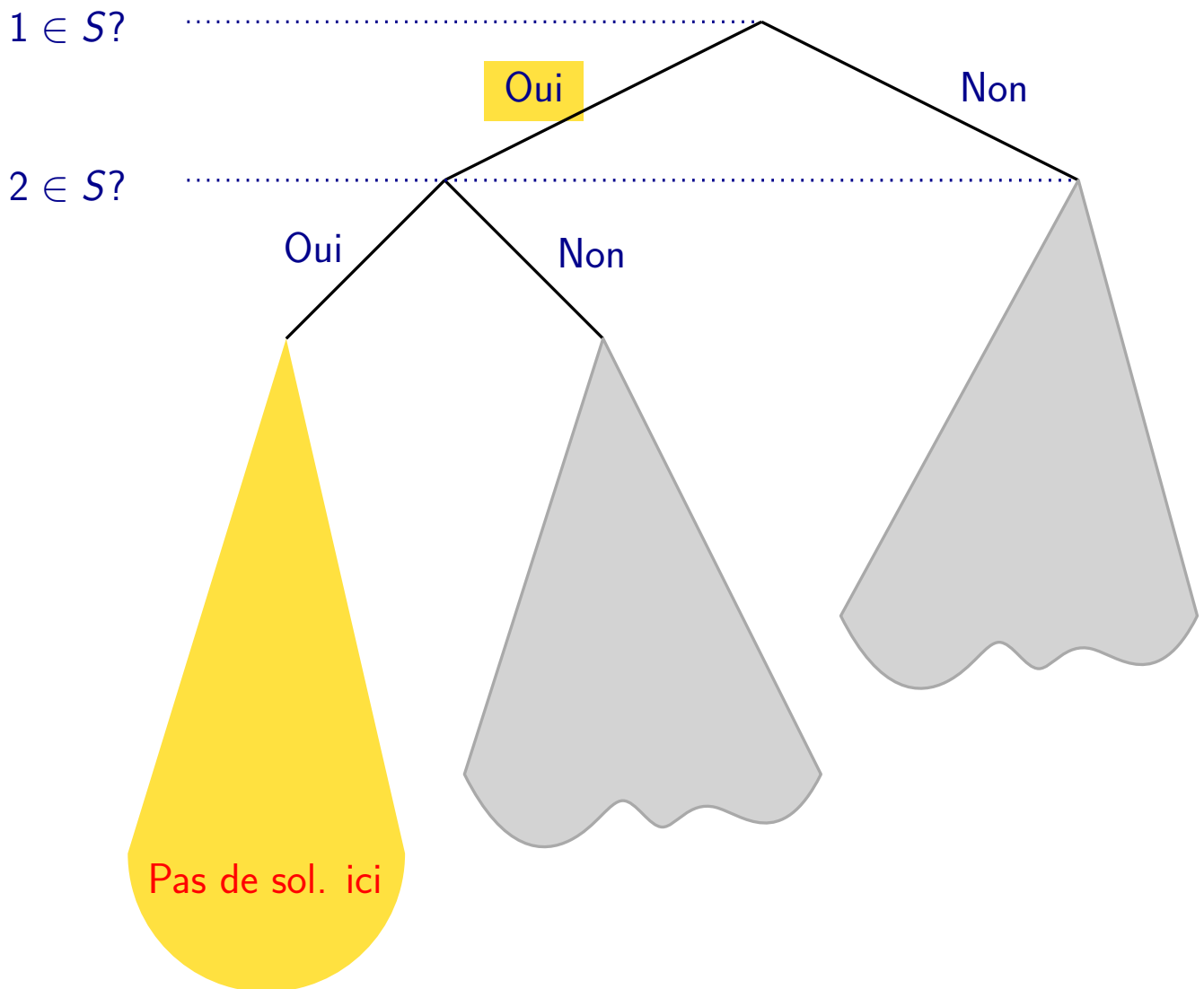
Flashlight search ou *Binary partition*



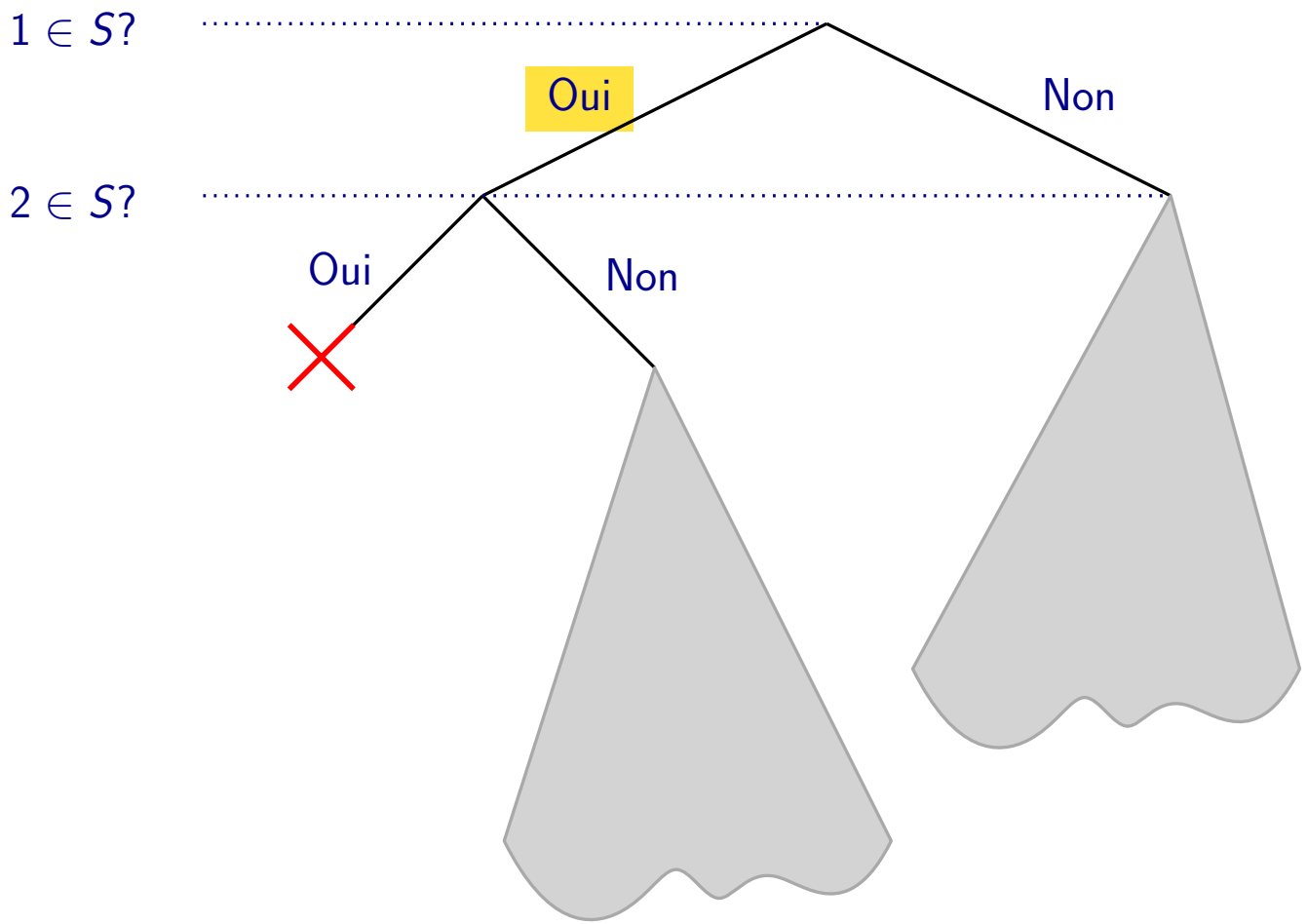
Flashlight search ou *Binary partition*



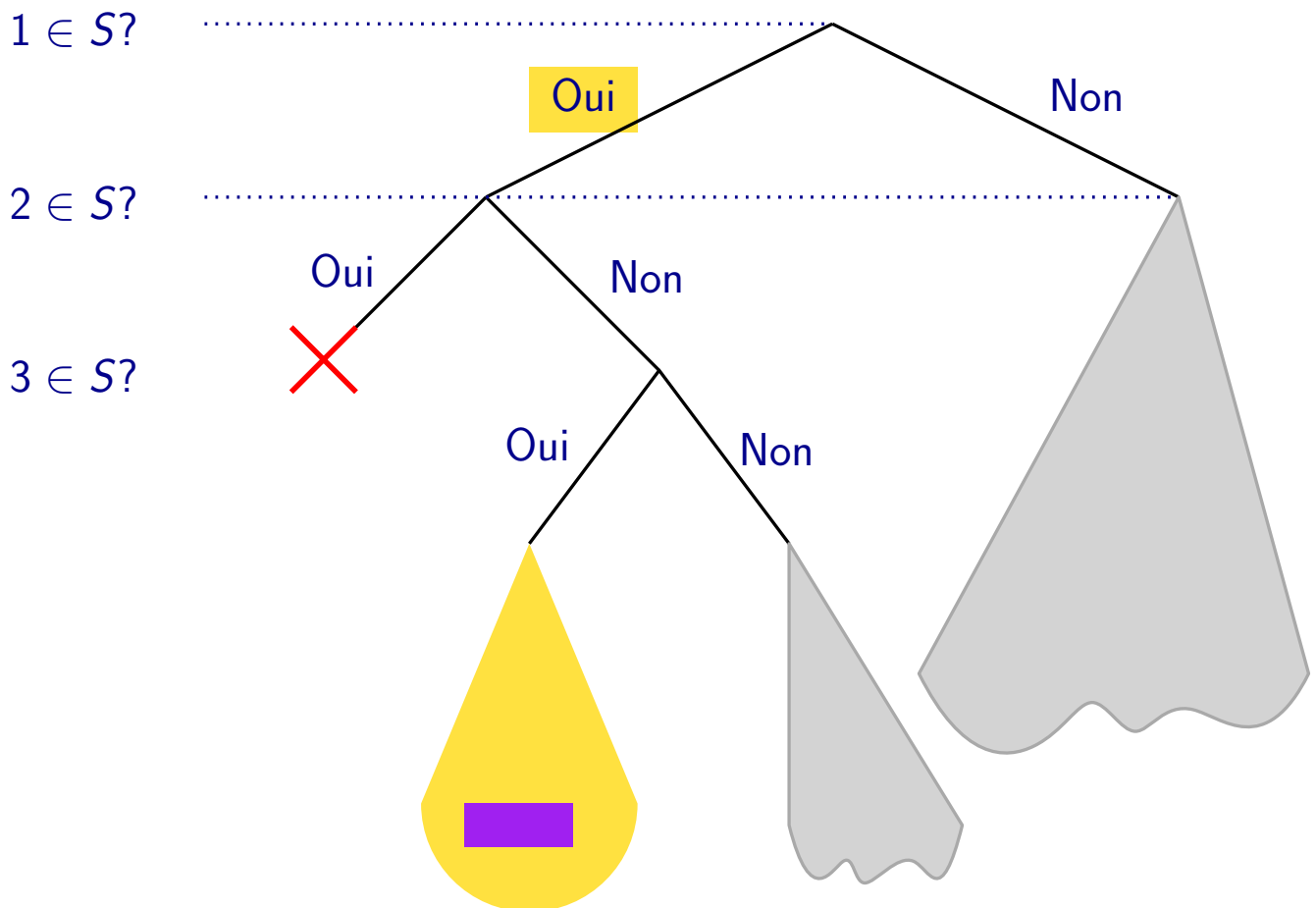
Flashlight search ou *Binary partition*



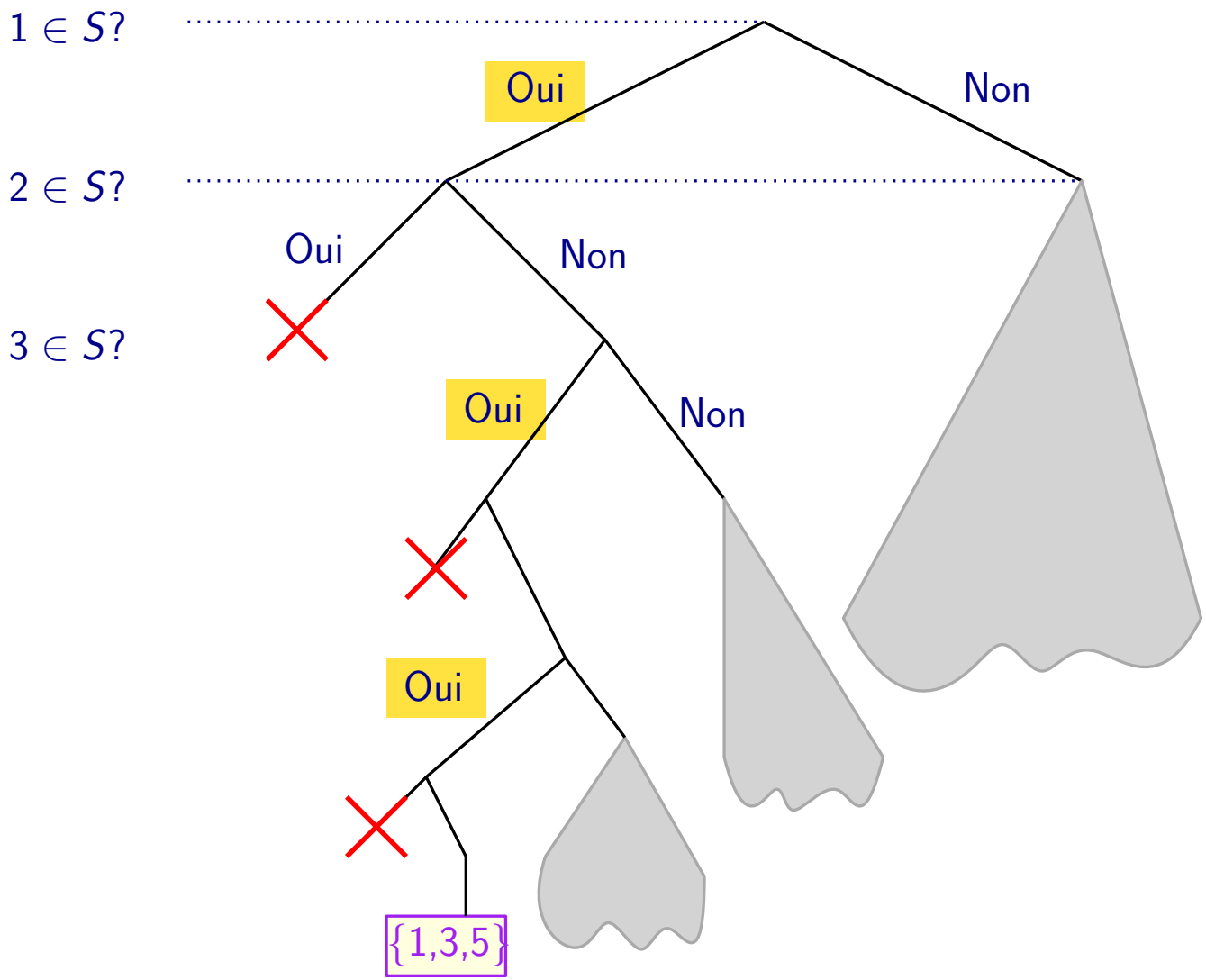
Flashlight search ou *Binary partition*



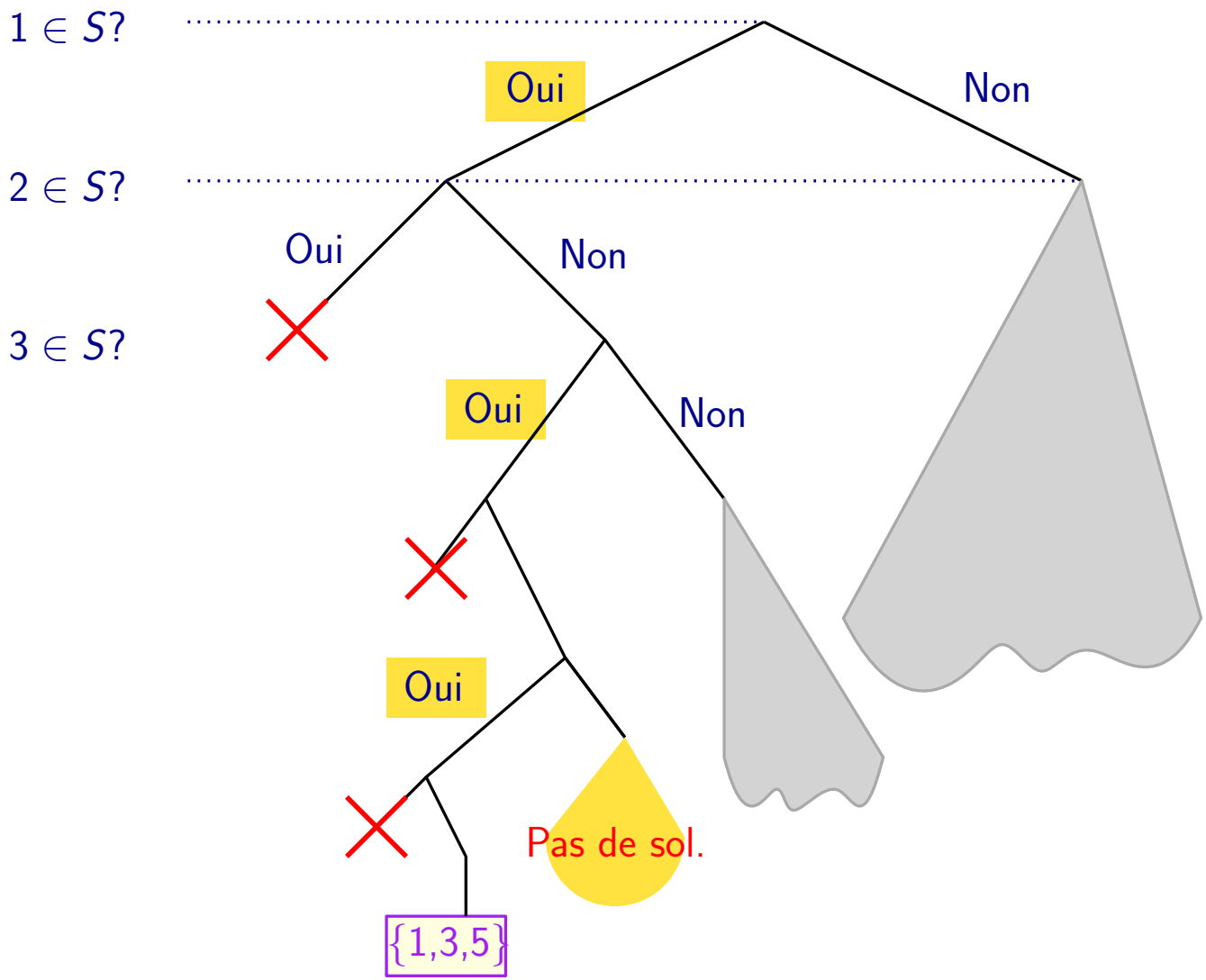
Flashlight search ou *Binary partition*



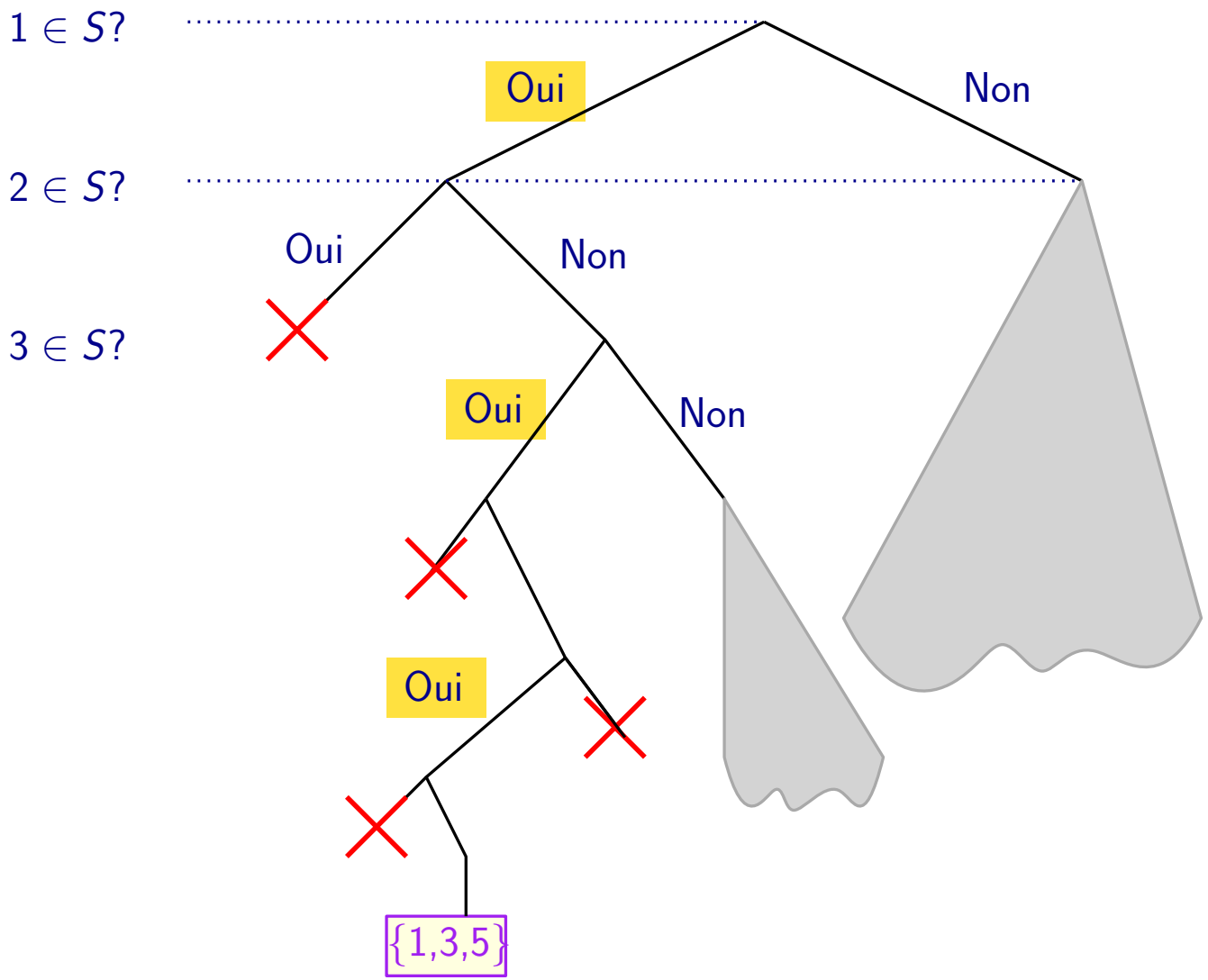
Flashlight search ou *Binary partition*



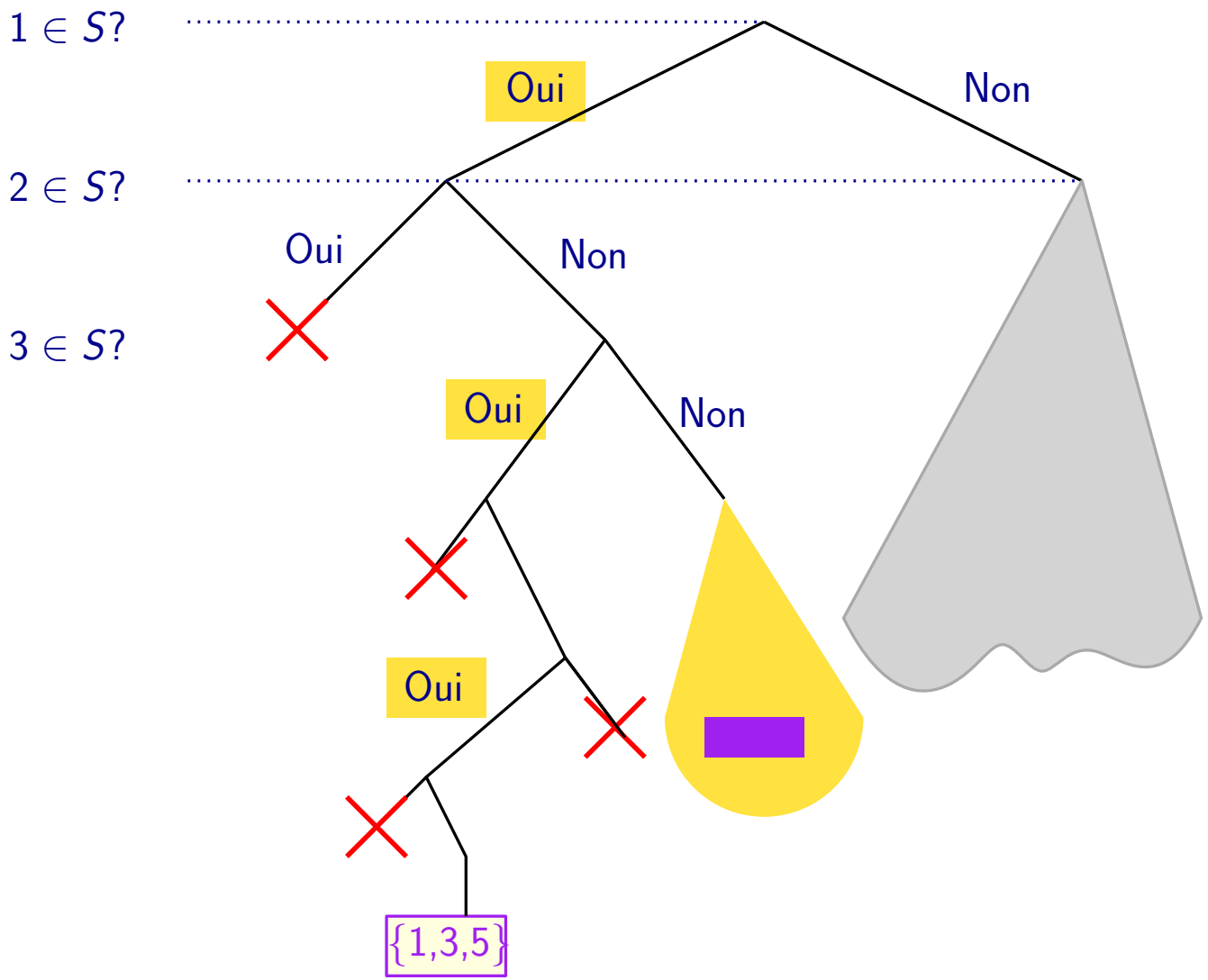
Flashlight search ou *Binary partition*



Flashlight search ou *Binary partition*



Flashlight search ou *Binary partition*

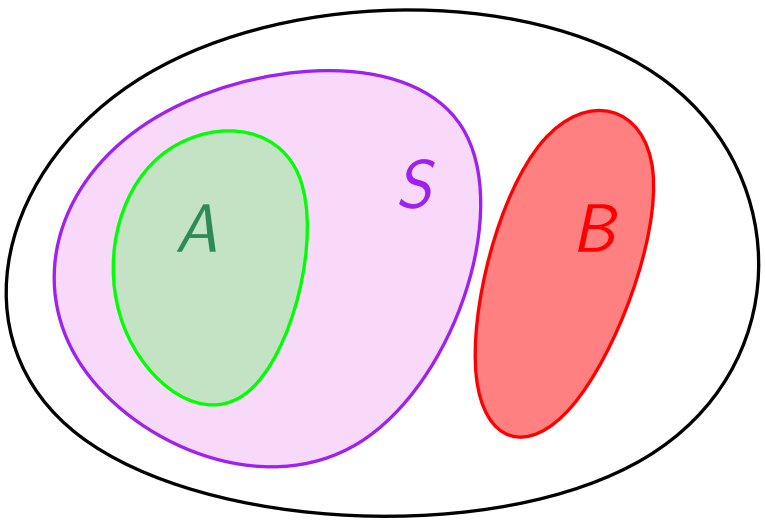


Flashlight search ou *Binary partition*

Pour faire marcher *Flashlight search* en délai (et espace) polynomial :

Répondre en temps poly au *problème d'extension* de $\mathcal{P}$:

Soit A et et B deux ensembles disjoints
y'-a-t'il une solution S de $\mathcal{P}$ tel que $A \subseteq S$ et $S \cap B = \emptyset$?



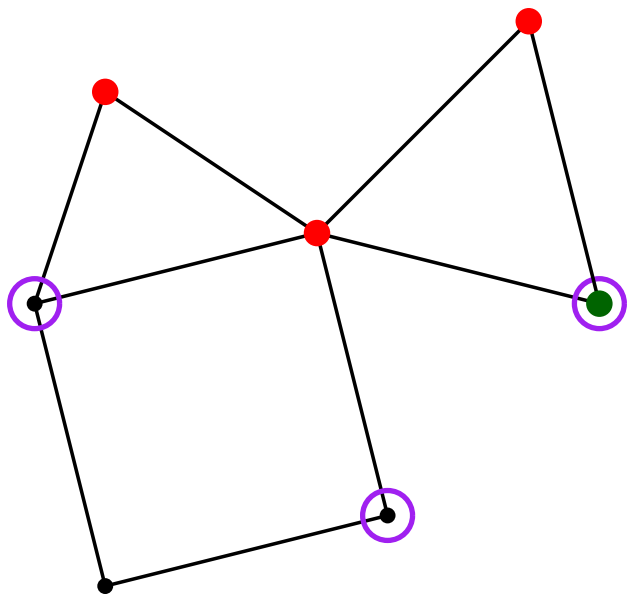
A est imposé dans la solution
et B est interdit dans la solution.

Problème d'extension : résultat négatif

Cadre : Π héréditaire et "non-triviale"
Pb: Π-réparation par suppr. de sommets

Entrée: Graphe G, A, $B \subseteq V$ disjoints
Sortie : Existe-t-il un sous-graphe induit :

- qui contient A
- a la propriété Π recherchée
- est maximal par inclusion
- et évite B ?

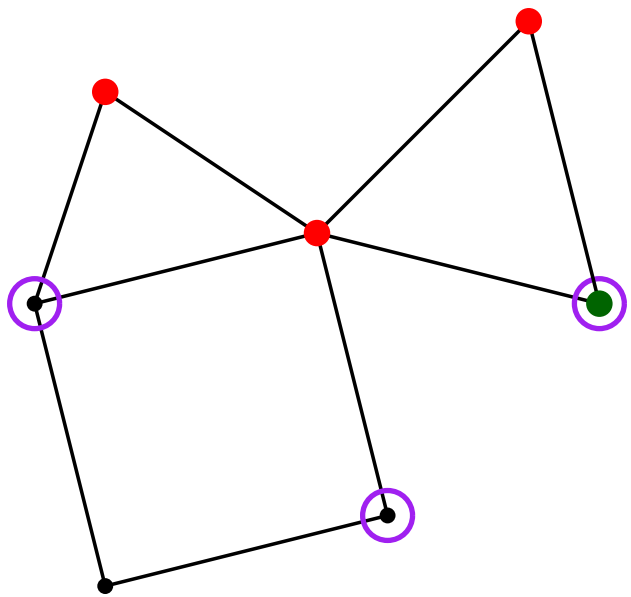


Problème d'extension : résultat négatif

Cadre : Π héréditaire et "non-triviale"
Pb: Π-réparation par suppr. de sommets

Entrée: Graphe G, A, $B \subseteq V$ disjoints
Sortie : Existe-t-il un sous-graphe induit :

- qui contient A
- a la propriété Π recherchée
- est maximal par inclusion
- et évite B ?



Théorème [Brosse, L., Limouzy, Mary, Pastor – 2020+]

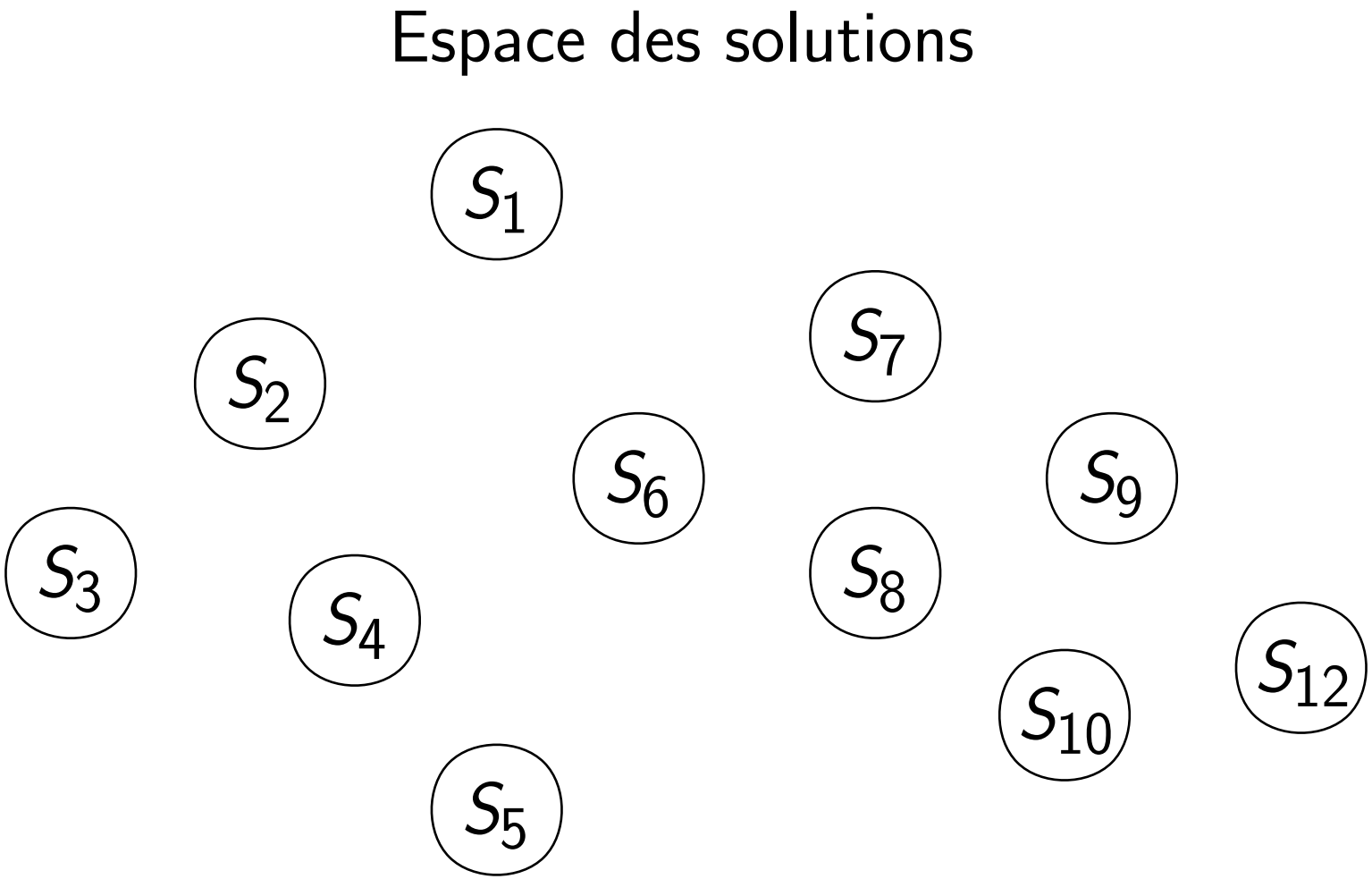
Le problème d'extension pour les Π-sous-graphes induits, pour toute propriété Π héréditaire non triviale, est NP-complet.

Trois méthodes classiques:

But : avoir un délai polynomial
+ espace poly pour 1 et 2 (et parfois 3)

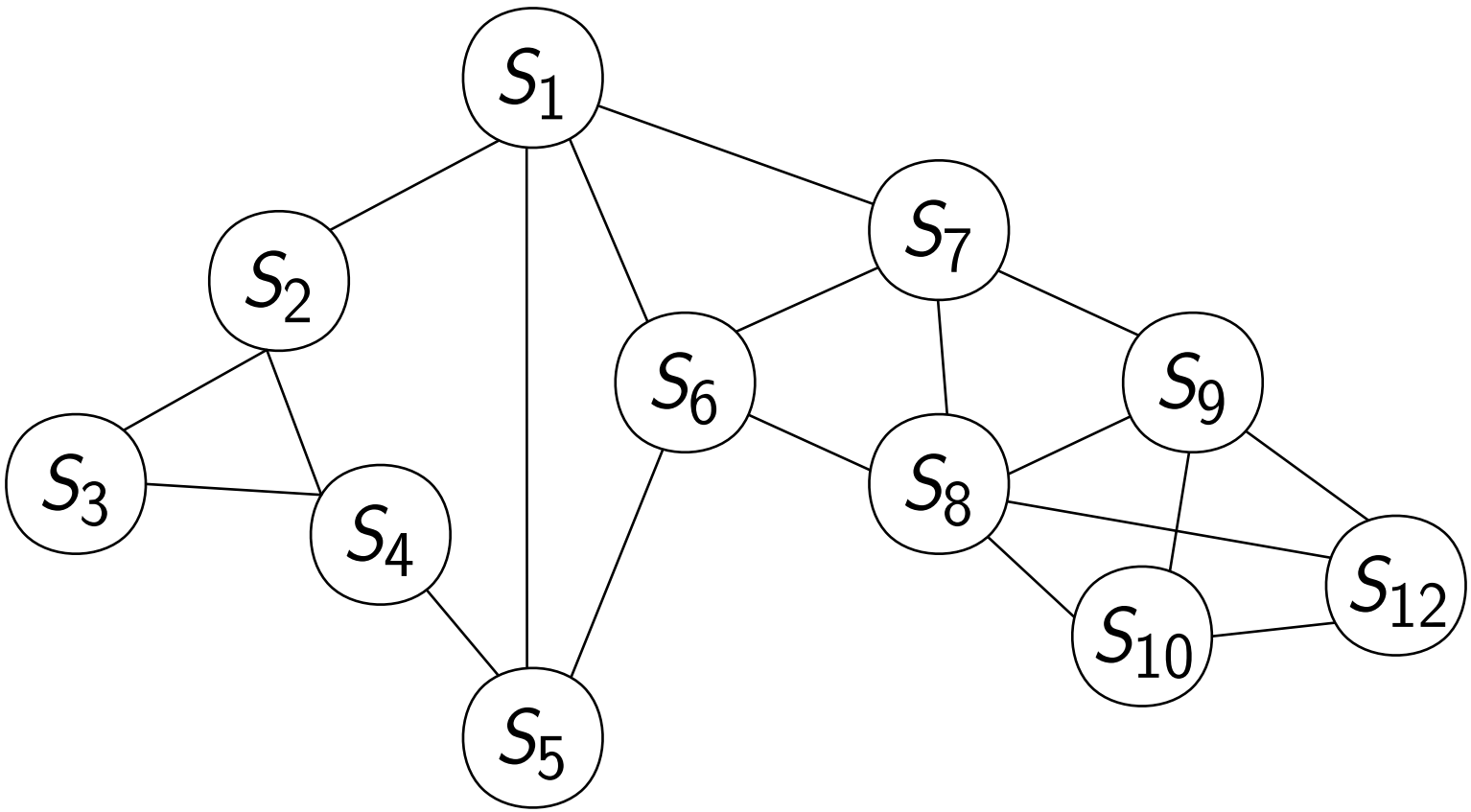
1. *Flashlight search* ou *Binary partition*
   [Read, Tarjan '75]
2. ***Reverse search***
   [Avis, Fukuda '96]
3. *Proximity Search*
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

Reverse search

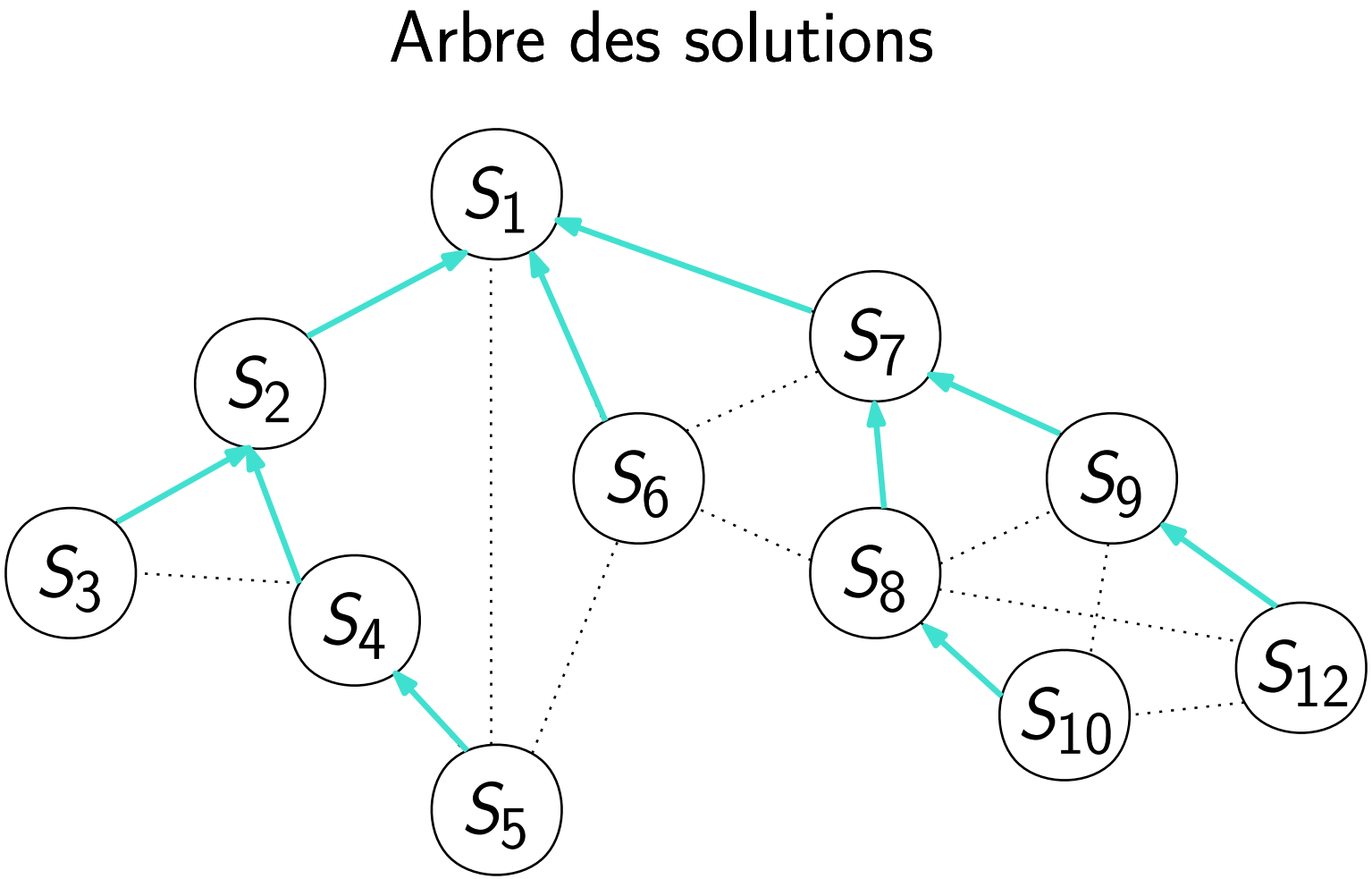


Reverse search

Graphe des solutions

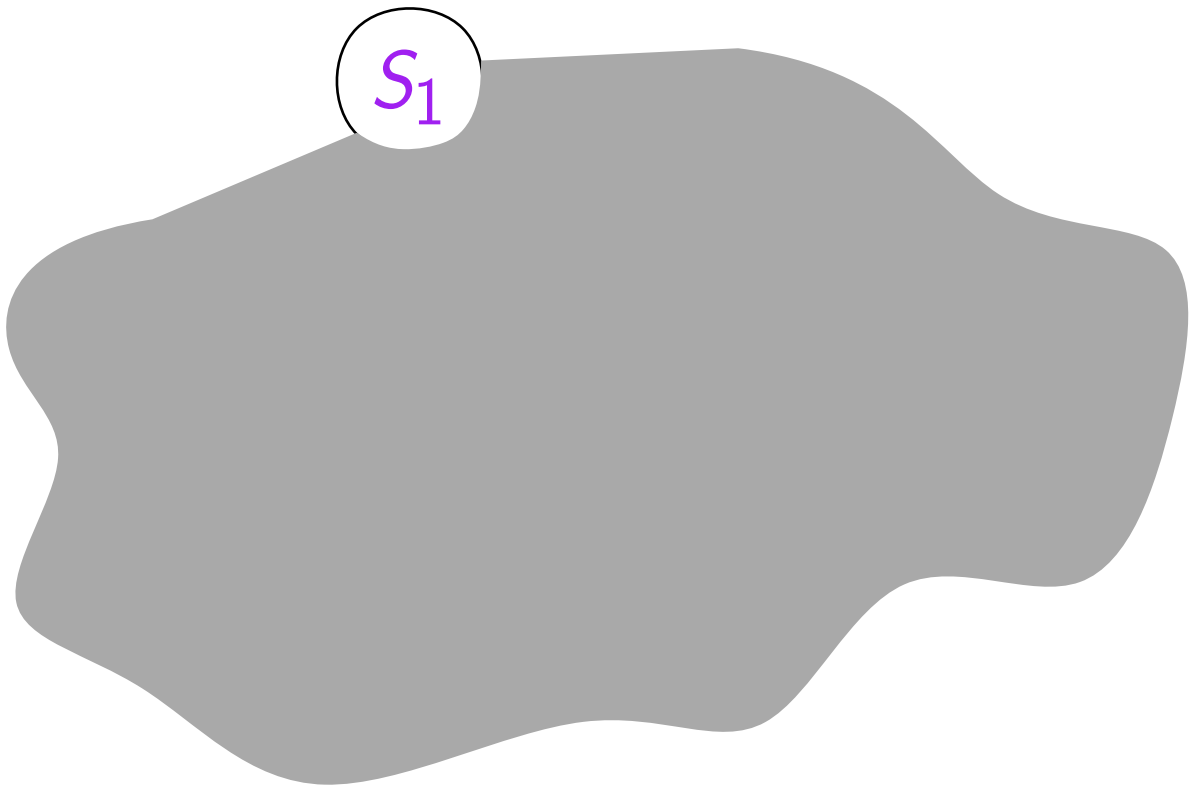


Reverse search

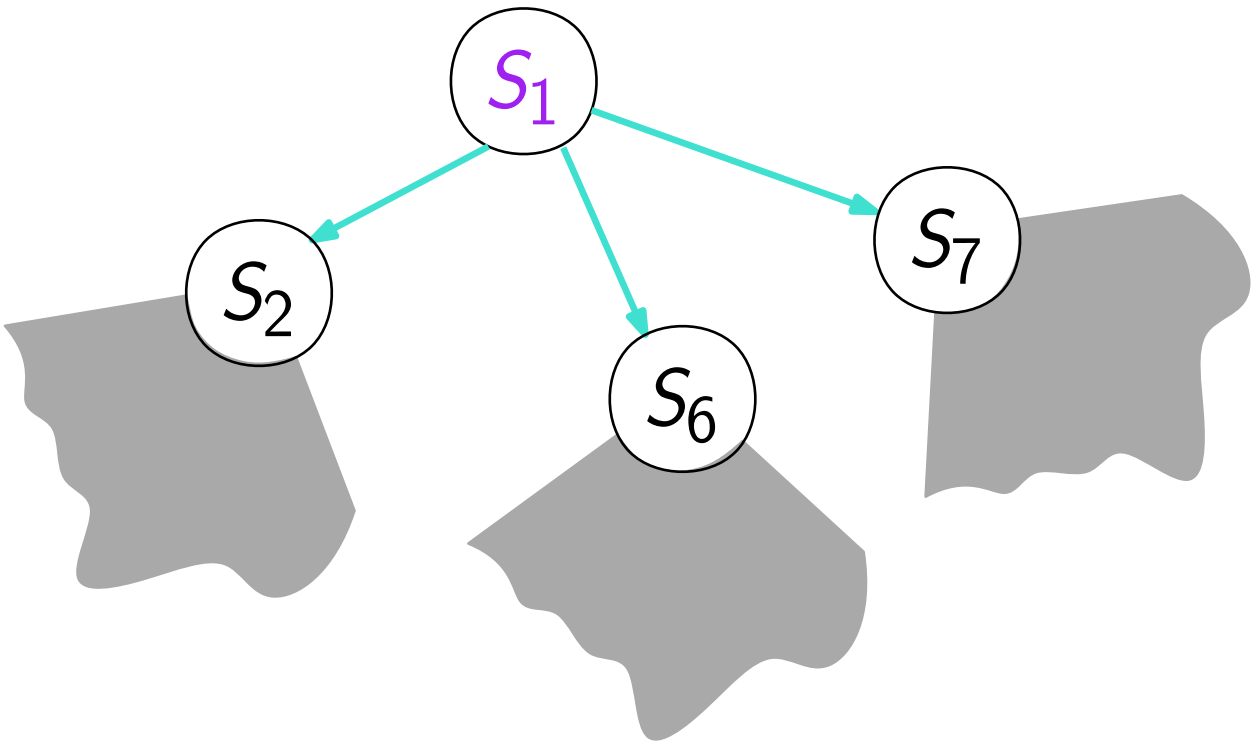


Reverse search

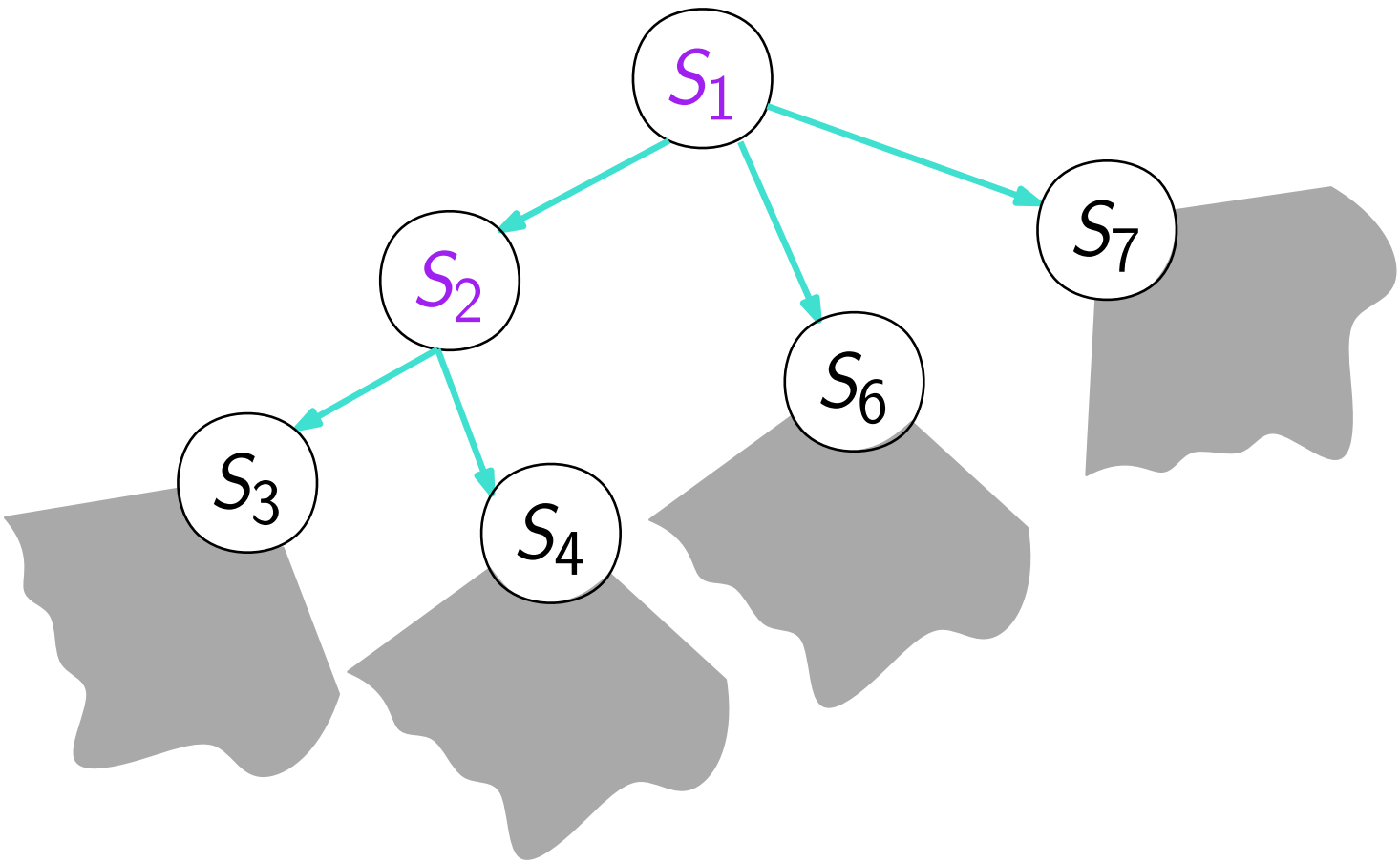
Arbre des solutions



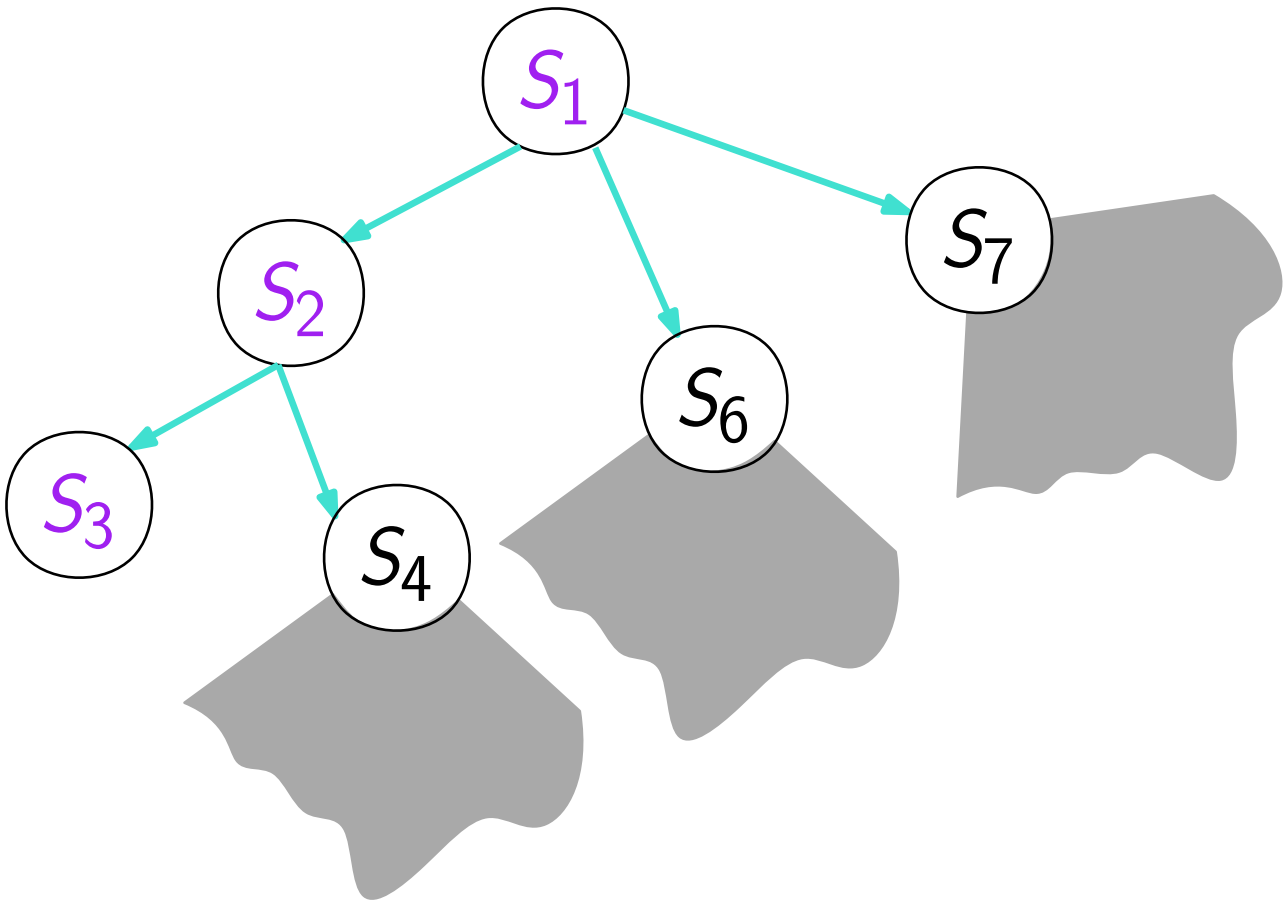
Reverse search



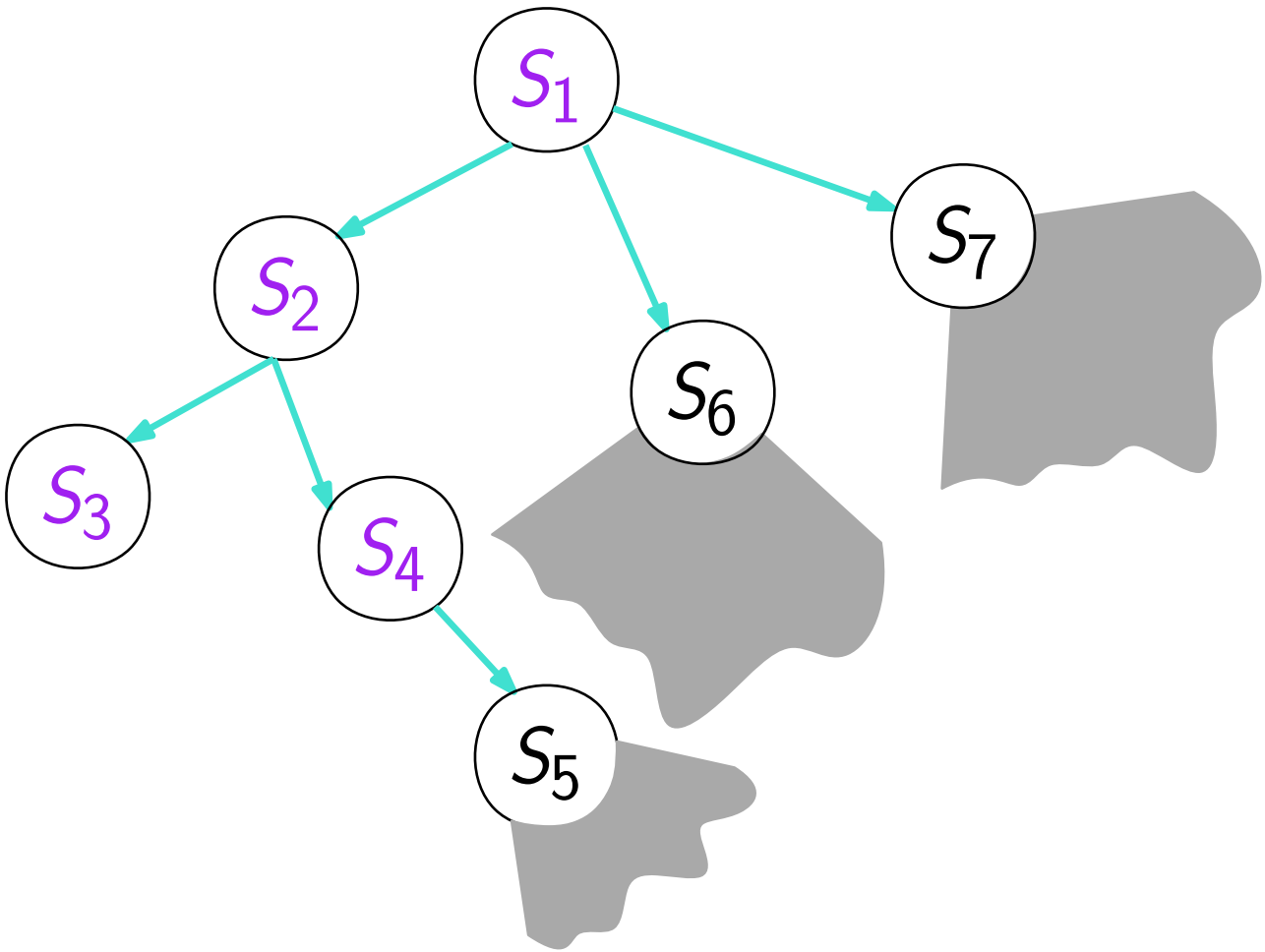
Reverse search



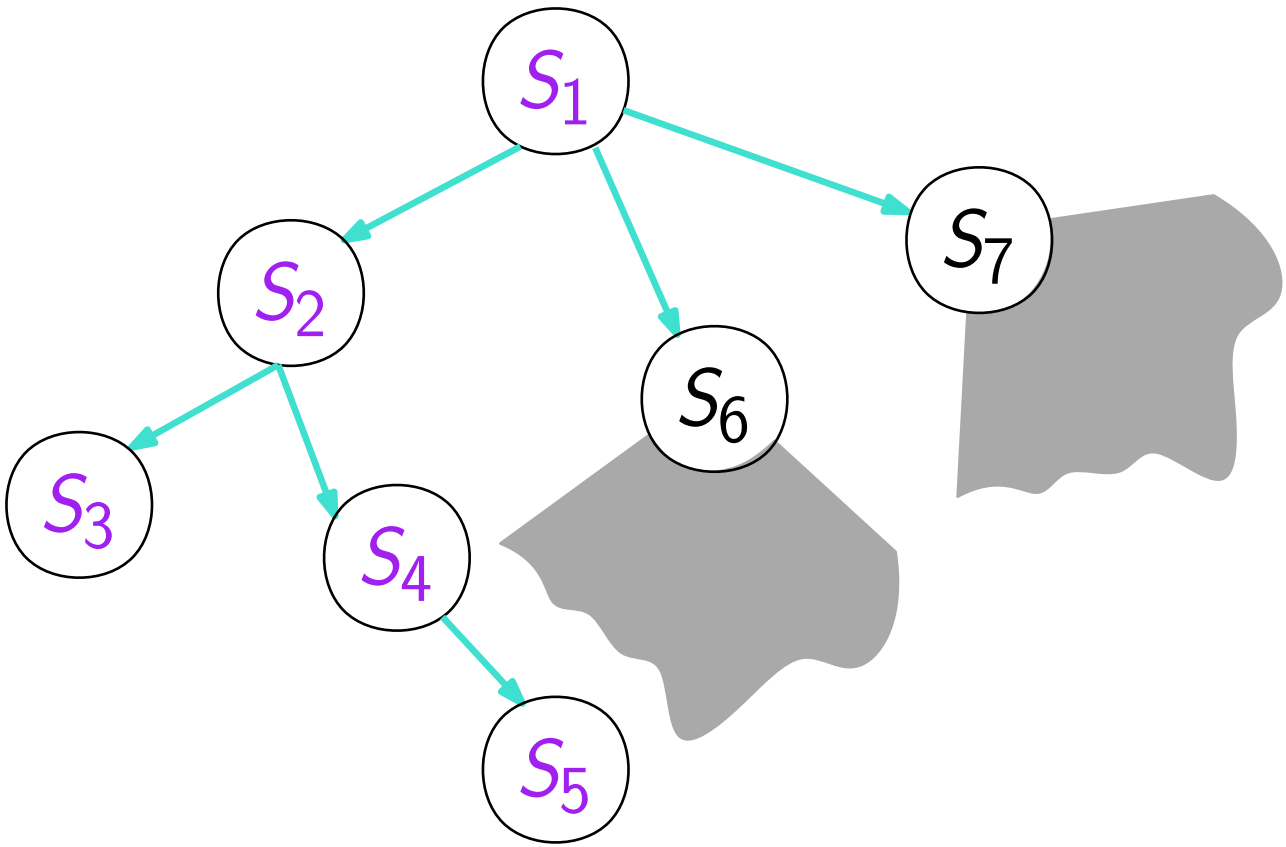
Reverse search



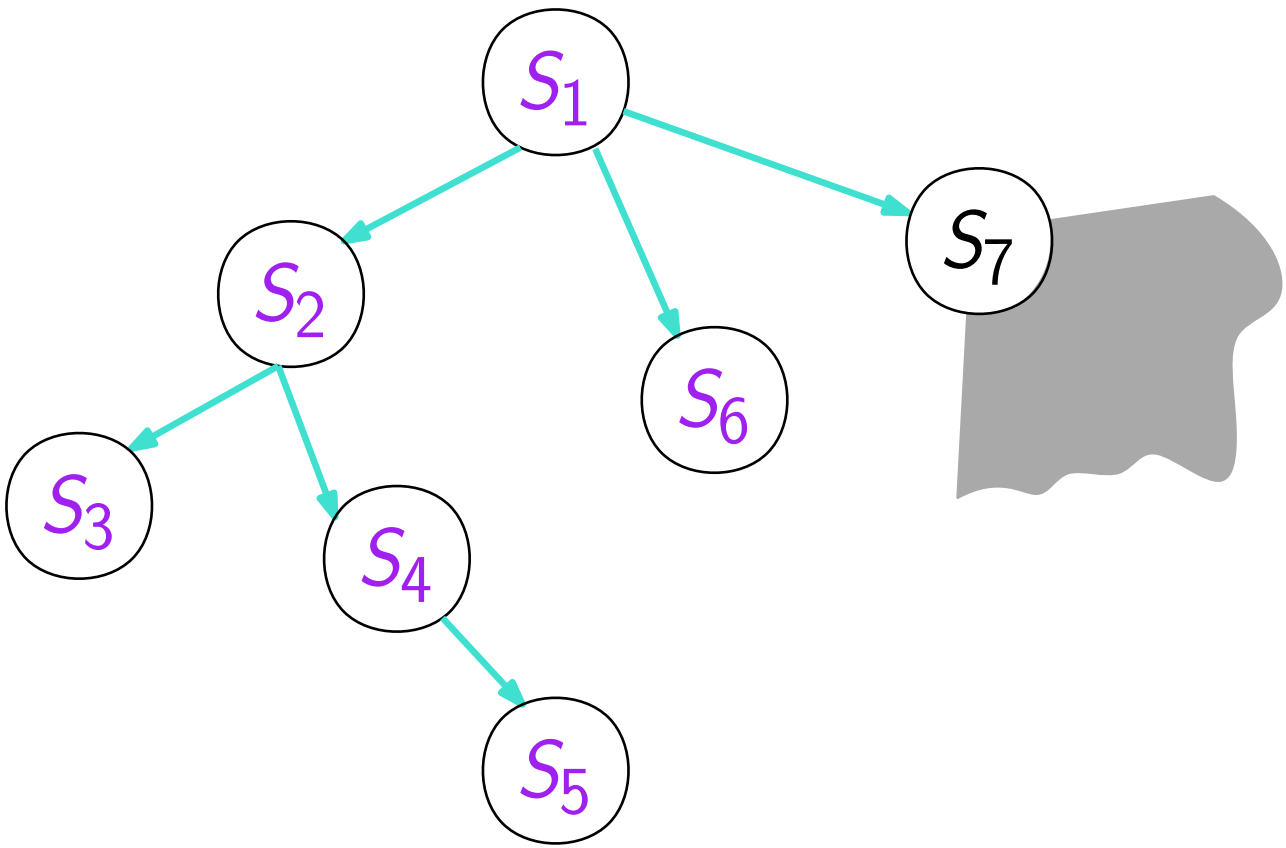
Reverse search



Reverse search



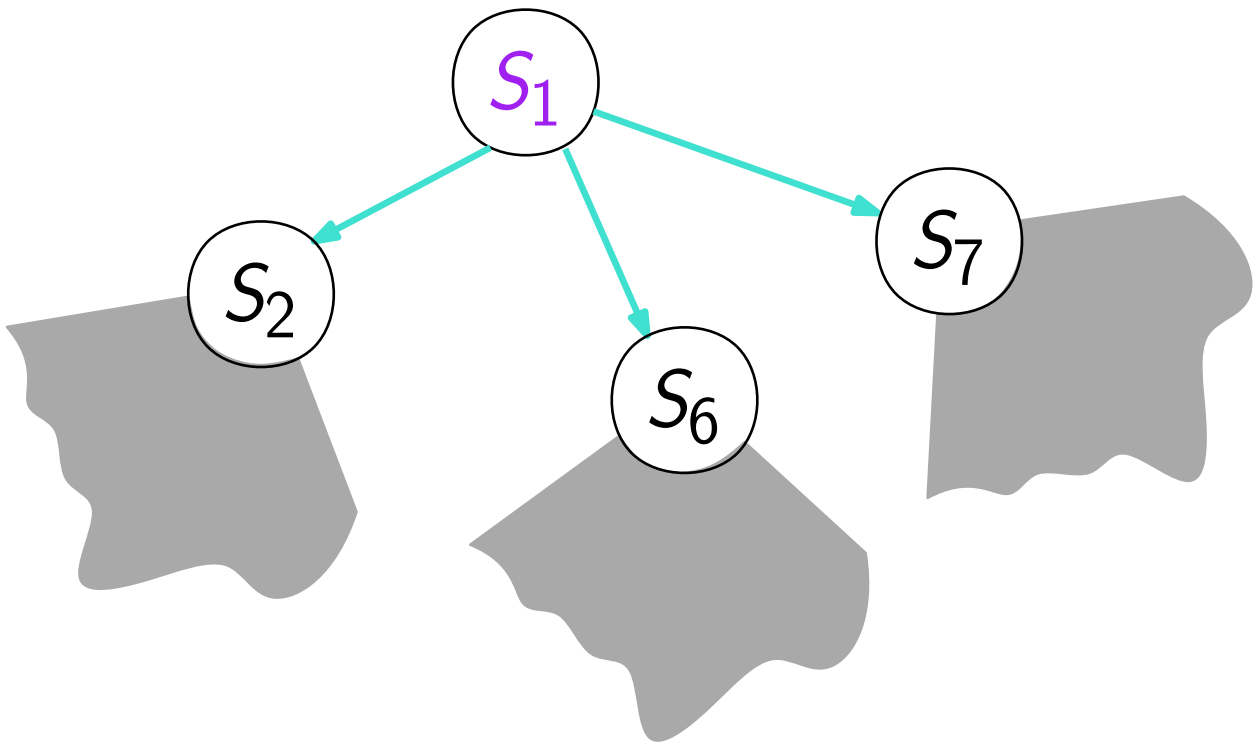
Reverse search



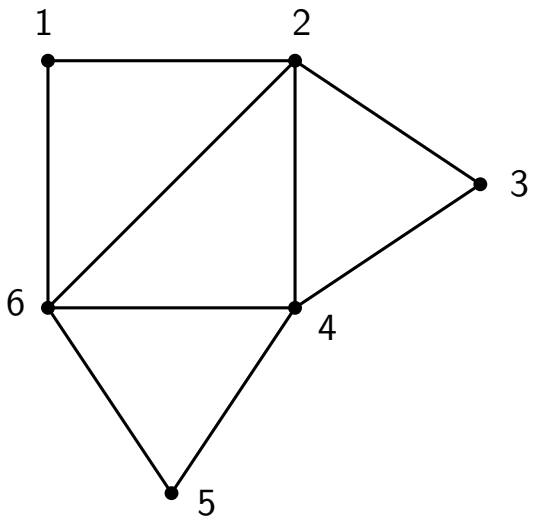
Reverse search

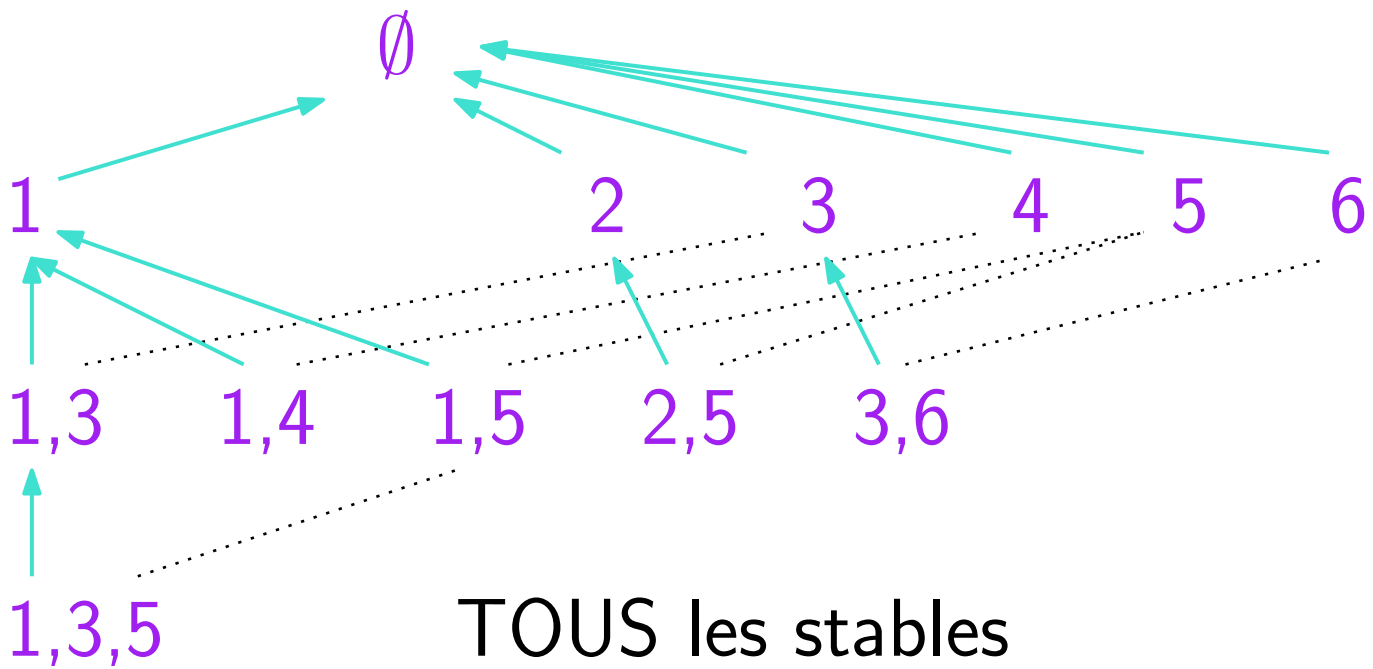
Pour faire marcher *Reverse search* en délai et espace polynomial :

Générer en temps et espace poly les *enfants* d'une solution (chaque solution doit n'avoir qu'***un seul*** père)



Reverse Search





TOUS les stables

Trois méthodes classiques:

But : avoir un délai polynomial
+ espace poly pour 1 et 2 (et parfois 3)

1. *Flashlight search* ou *Binary partition*
   [Read, Tarjan '75]
2. *Reverse search*
   [Avis, Fukuda '96]
3. ***Proximity Search***
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

Proximity Search

Pensé pour énumérer des solutions *maximales* (ou min.) par inclusion à un problème.

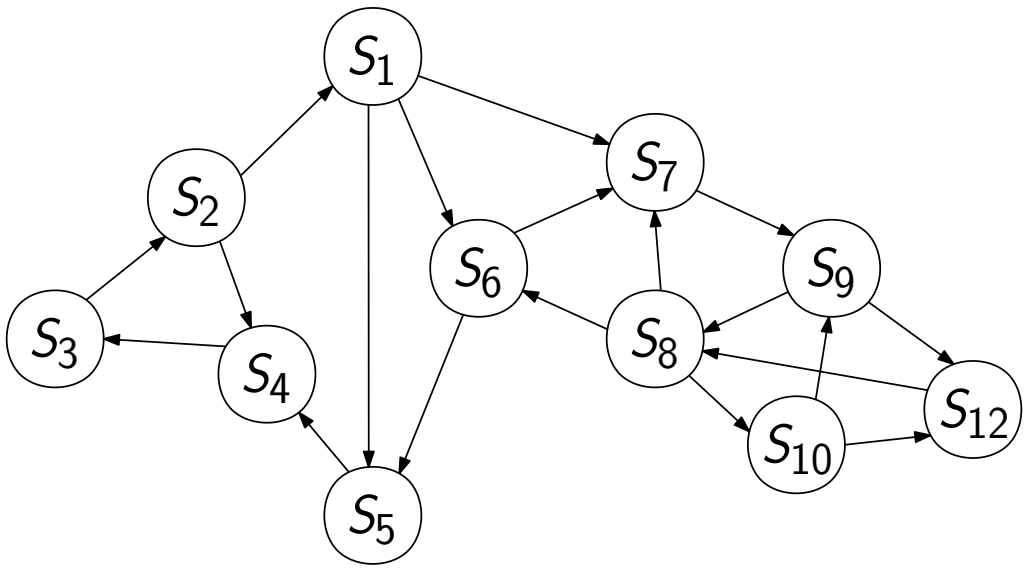
Proximity Search

Pensé pour énumérer des solutions *maximales* (ou min.) par inclusion à un problème.

Parcours en profondeur récursif assez classique avec conditions :

- pouvoir générer les **voisins** d'un sommets en temps poly $\rightarrow$ *degré poly*

Graphe des solutions



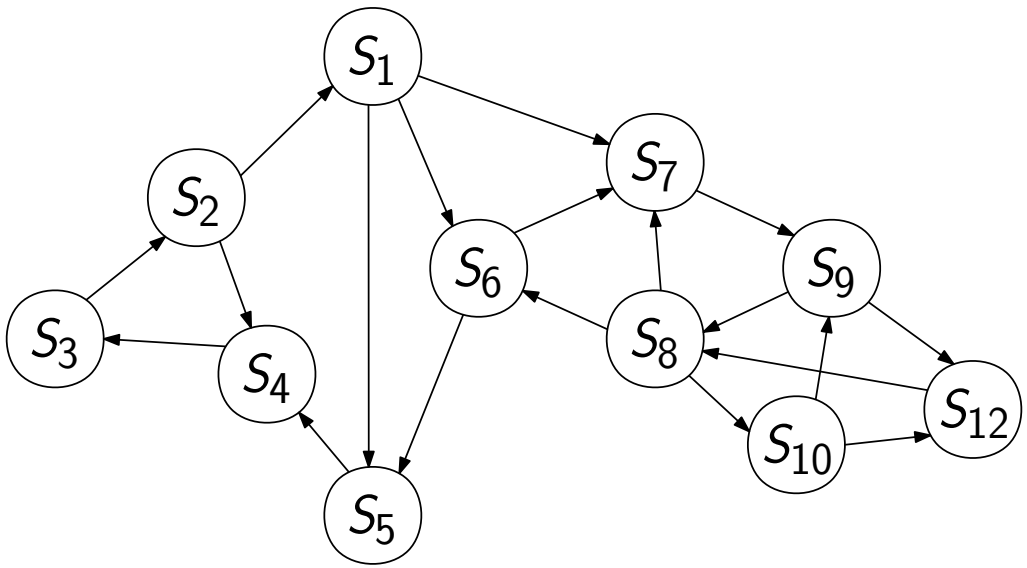
Proximity Search

Pensé pour énumérer des solutions *maximales* (ou min.) par inclusion à un problème.

Parcours en profondeur récursif assez classique avec conditions :

- pouvoir générer les **voisins** d'un sommets en temps poly
  $\rightarrow$ *degré poly*
- vérifier si un sommet a déjà été visité ou non
  $\rightarrow$ *espace exponentiel :-(*

Graphe des solutions



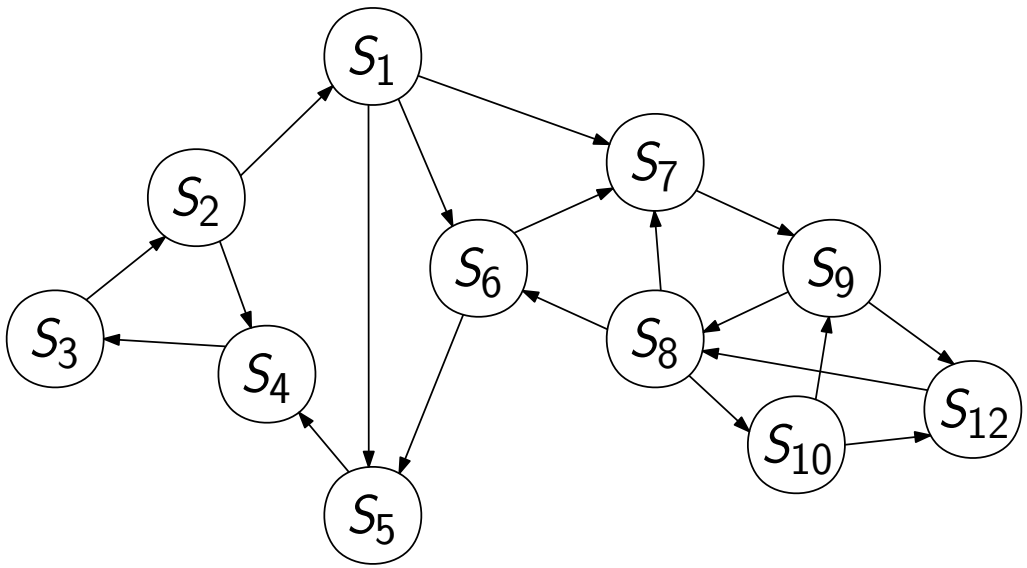
Proximity Search

Pensé pour énumérer des solutions *maximales* (ou min.) par inclusion à un problème.

Parcours en profondeur récursif assez classique avec conditions :

- pouvoir générer les **voisins** d'un sommets en temps poly
  $\rightarrow$ *degré poly*
- vérifier si un sommet a déjà été visité ou non
  $\rightarrow$ *espace exponentiel :-(*
- forte connexité du graphe à prouver via la notion de **proximité**

Graphe des solutions



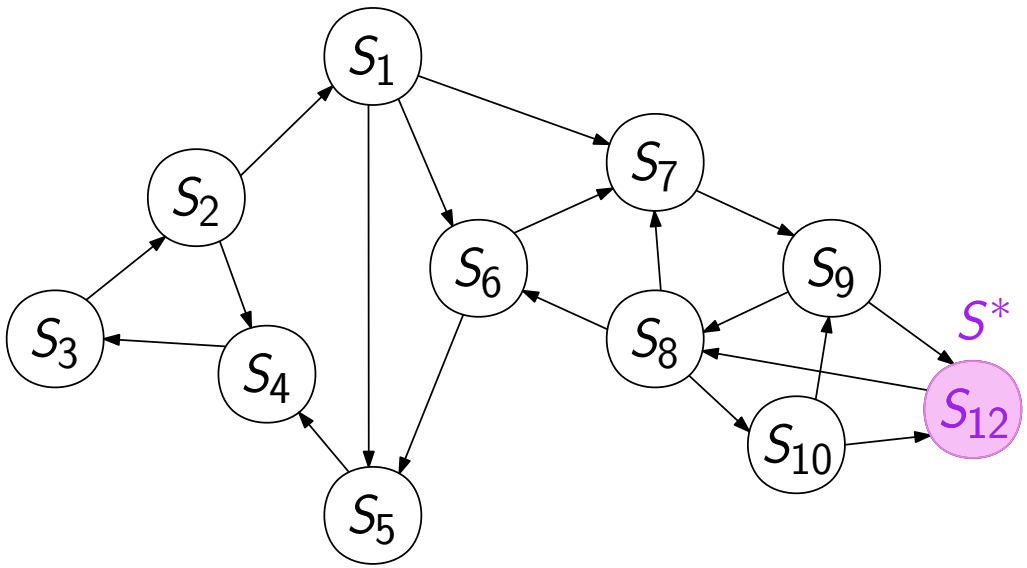
Proximity Search

Pensé pour énumérer des solutions *maximales* (ou min.) par inclusion à un problème.

Parcours en profondeur récursif assez classique avec conditions :

- pouvoir générer les **voisins** d'un sommets en temps poly
  $\rightarrow$ *degré poly*
- vérifier si un sommet a déjà été visité ou non
  $\rightarrow$ *espace exponentiel :-(*
- forte connexité du graphe à prouver via la notion de **proximité**

Graphe des solutions



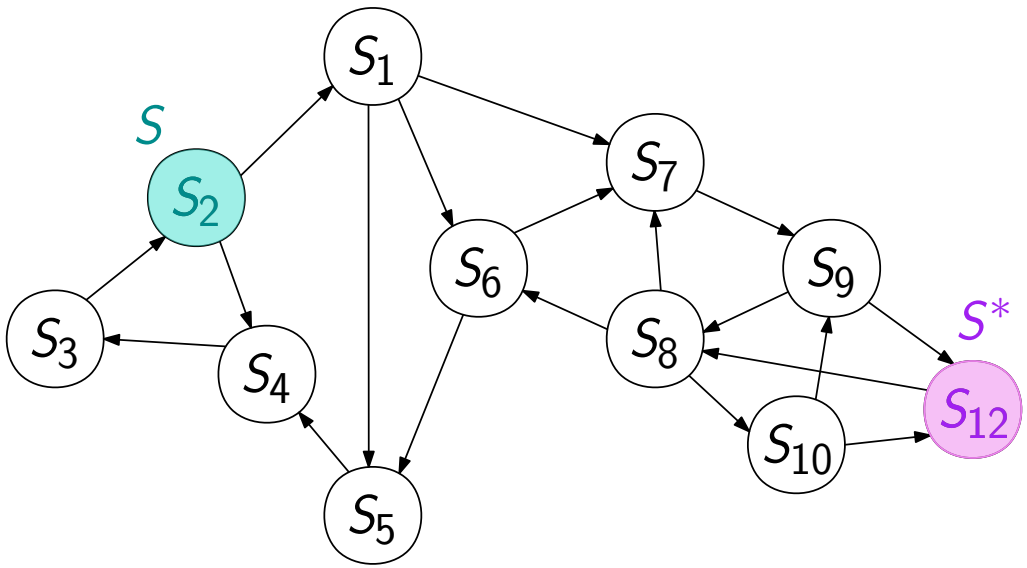
Proximity Search

Pensé pour énumérer des solutions *maximales* (ou min.) par inclusion à un problème.

Parcours en profondeur récursif assez classique avec conditions :

- pouvoir générer les **voisins** d'un sommets en temps poly
  $\rightarrow$ *degré poly*
- vérifier si un sommet a déjà été visité ou non
  $\rightarrow$ *espace exponentiel :-(*
- forte connexité du graphe à prouver via la notion de **proximité**

Graphe des solutions



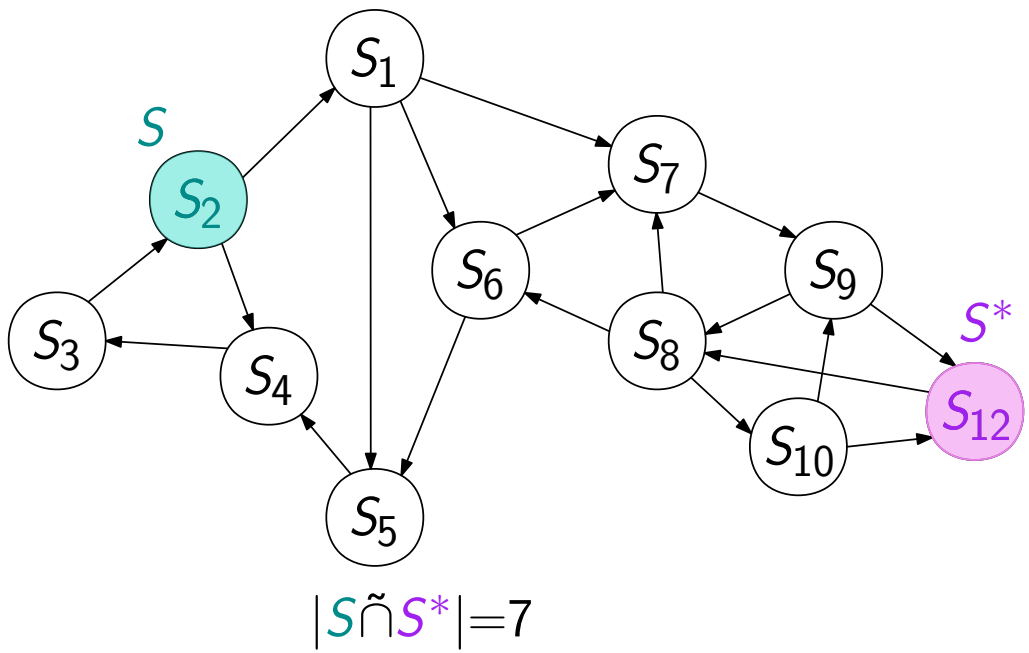
Proximity Search

Pensé pour énumérer des solutions *maximales* (ou min.) par inclusion à un problème.

Parcours en profondeur récursif assez classique avec conditions :

- pouvoir générer les **voisins** d'un sommets en temps poly
  $\rightarrow$ *degré poly*
- vérifier si un sommet a déjà été visité ou non
  $\rightarrow$ *espace exponentiel :-(*
- forte connexité du graphe à prouver via la notion de **proximité**

Graphe des solutions



$|S \tilde{\cap} S^*|=7$

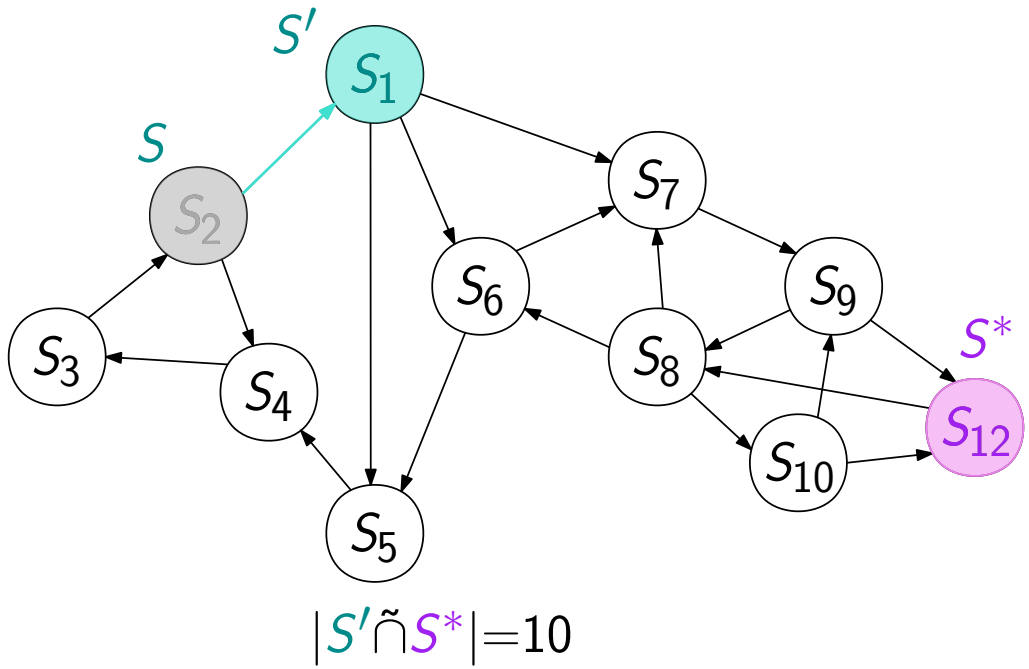
Proximity Search

Pensé pour énumérer des solutions *maximales* (ou min.) par inclusion à un problème.

Parcours en profondeur récursif assez classique avec conditions :

- pouvoir générer les **voisins** d'un sommets en temps poly
  $\rightarrow$ *degré poly*
- vérifier si un sommet a déjà été visité ou non
  $\rightarrow$ *espace exponentiel :-(*
- forte connexité du graphe à prouver via la notion de **proximité**

Graphe des solutions



$|S' \tilde{\cap} S^*|=10$

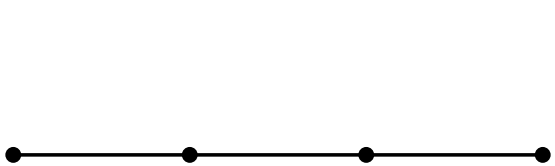
Trois méthodes classiques:

But : avoir un délai polynomial
+ espace poly pour 1 et 2 (et parfois 3)

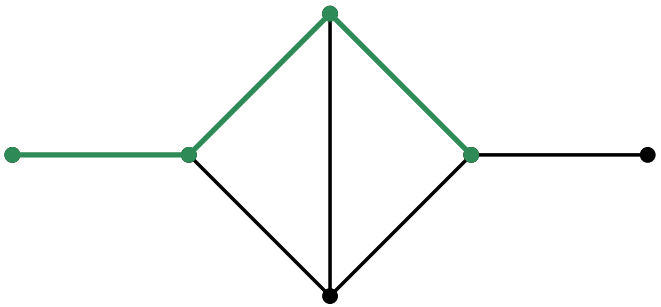
1. *Flashlight search* ou *Binary partition*
   [Read, Tarjan '75]
2. *Reverse search*
   [Avis, Fukuda '96]
3. *Proximity Search*
   [Conte, Uno '19]
   [Conte, Grossi, Marino, Uno, Versari, '21]

(Un petit quiz avant d'enchaîner ? `http://wooclap.com/JGA21`)

Cographes = P_4-free



P_4



$G \notin Cographs$

La classe des cographes est héréditaire et auto-complémentaire.

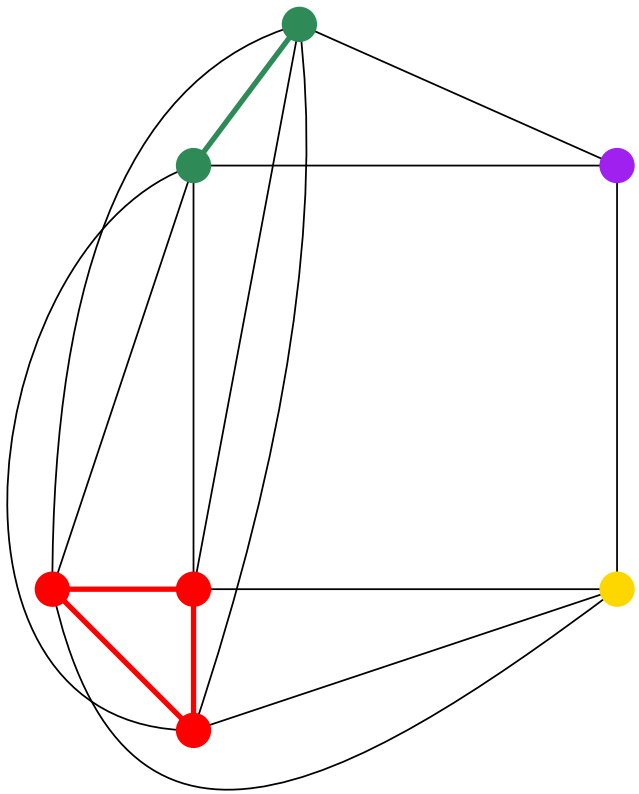
[Brosse, L., Limouzy, Mary, Pastor – 2020^+]

Via *Proximity Search*, il existe un algorithme pour :
Entrée: un graphe G
Sortie : énumération de tous les sous-cographes induits maximaux de G (réparations en cographe par suppression min. de sommets);
qui fonctionne en complexité

- délai-polynomial ;
- espace exponentiel.

Cographes & jumeaux

Les cographes ont un ordre de construction par vrai ou faux jumeau.



Cographes & jumeaux

Les cographes ont un ordre de construction par vrai ou faux jumeau.

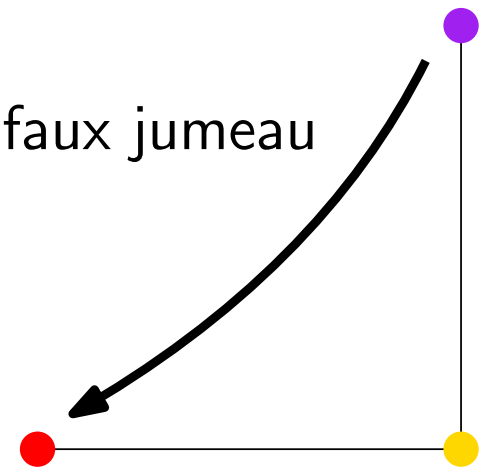
Cographes & jumeaux

Les cographes ont un ordre de construction par vrai ou faux jumeau.

vrai jumeau

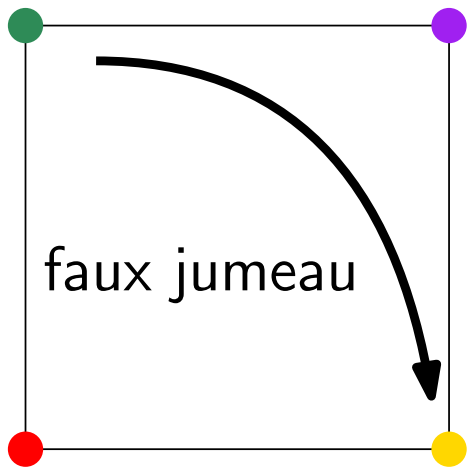
Cographes & jumeaux

Les cographes ont un ordre de construction par vrai ou faux jumeau.



Cographes & jumeaux

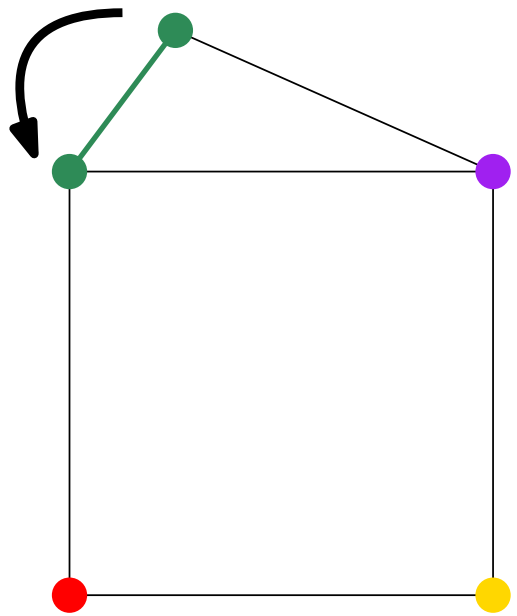
Les cographes ont un ordre de construction par vrai ou faux jumeau.



Cographes & jumeaux

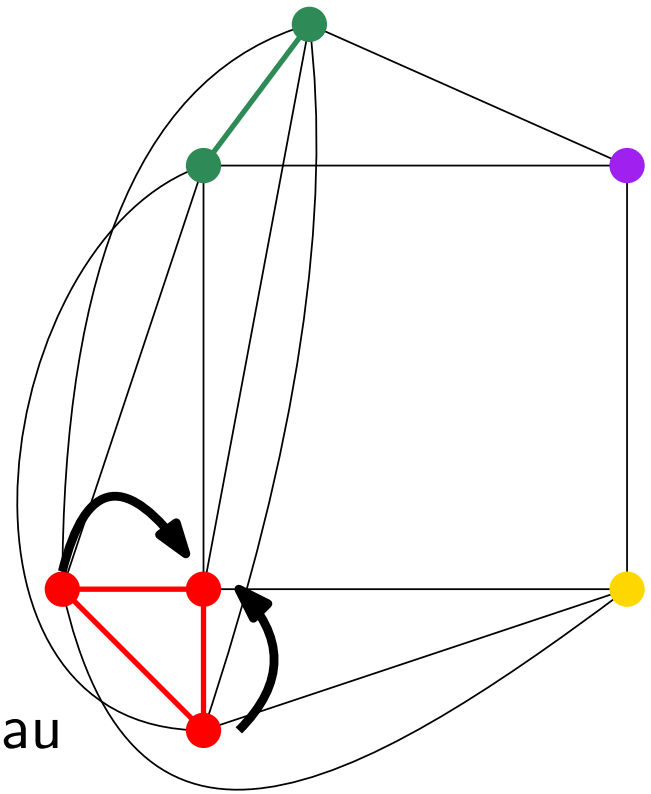
Les cographes ont un ordre de construction par vrai ou faux jumeau.

vrai jumeau



Cographes & jumeaux

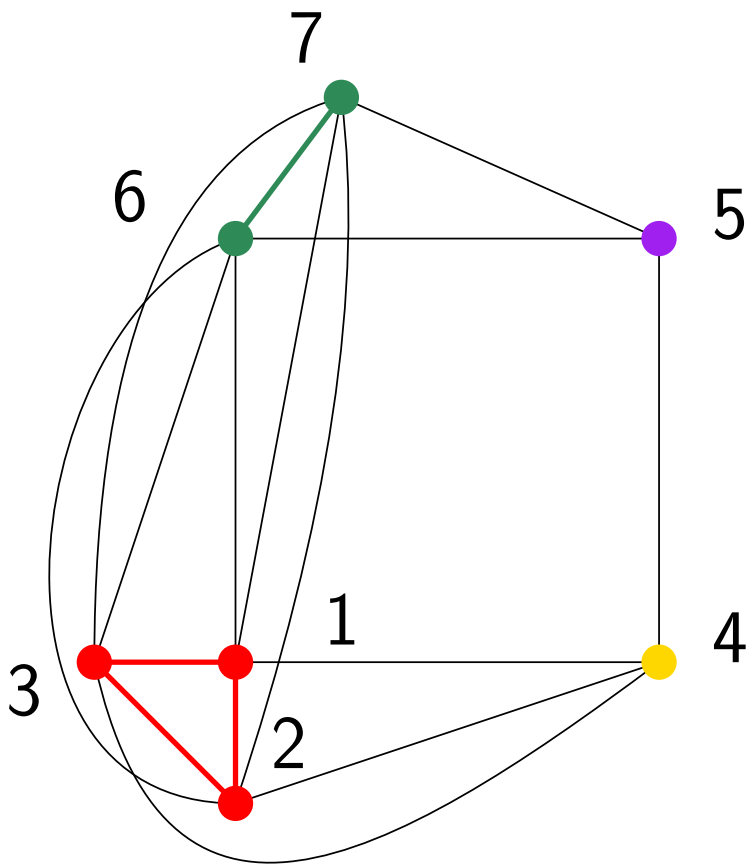
Les cographes ont un ordre de construction par vrai ou faux jumeau.



vrai jumeau

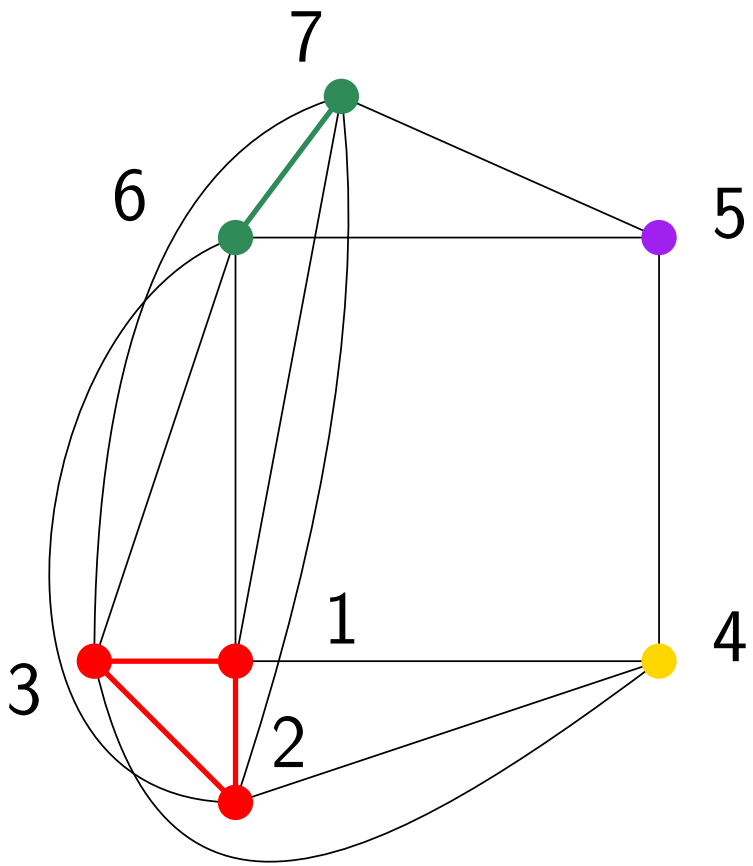
Cographes & jumeaux

Les cographes ont un ordre de construction par vrai ou faux jumeau.



Cographes & jumeaux

Les cographes ont un ordre de construction par vrai ou faux jumeau.

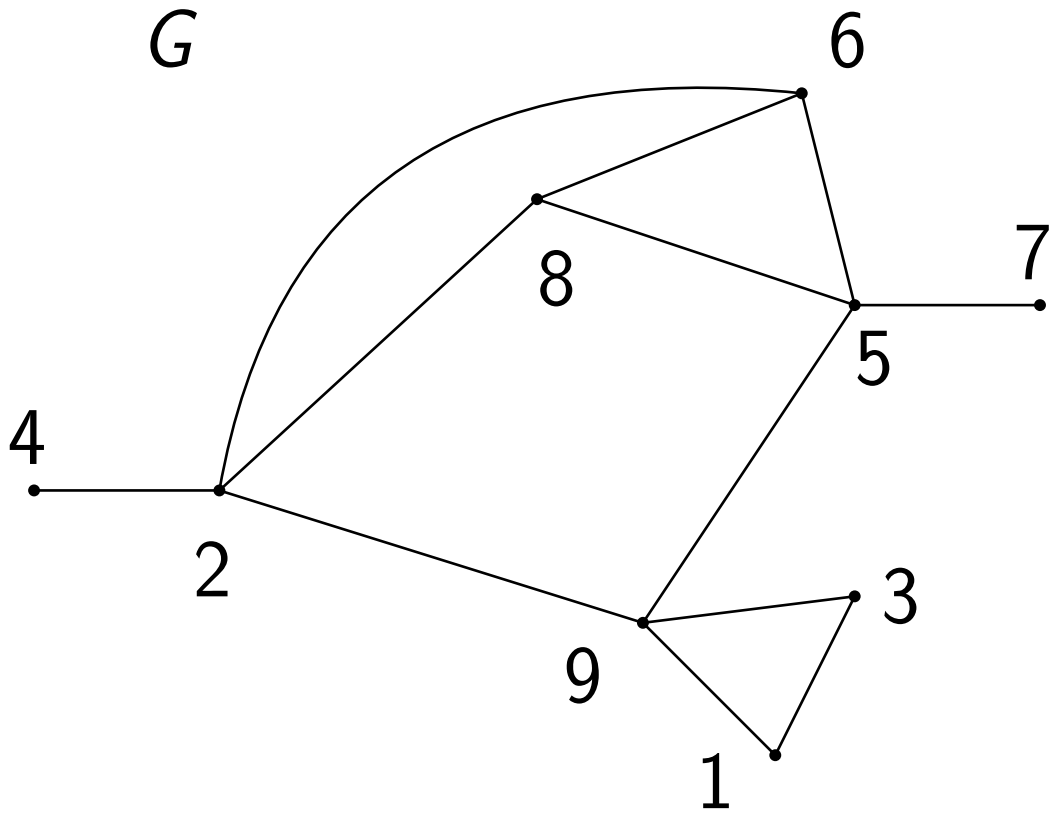


1 4 5 6 2 3 7

Ordre canonique de construction par jumeau

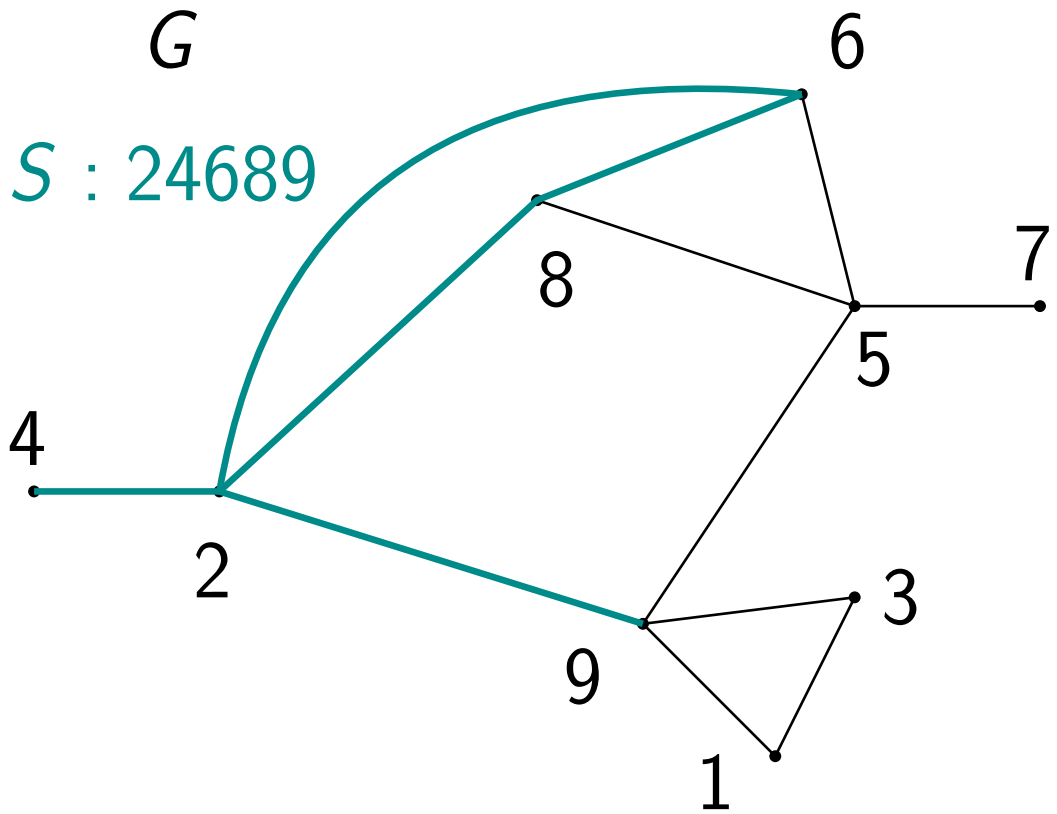
Proximité entre deux sous-cographes

But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



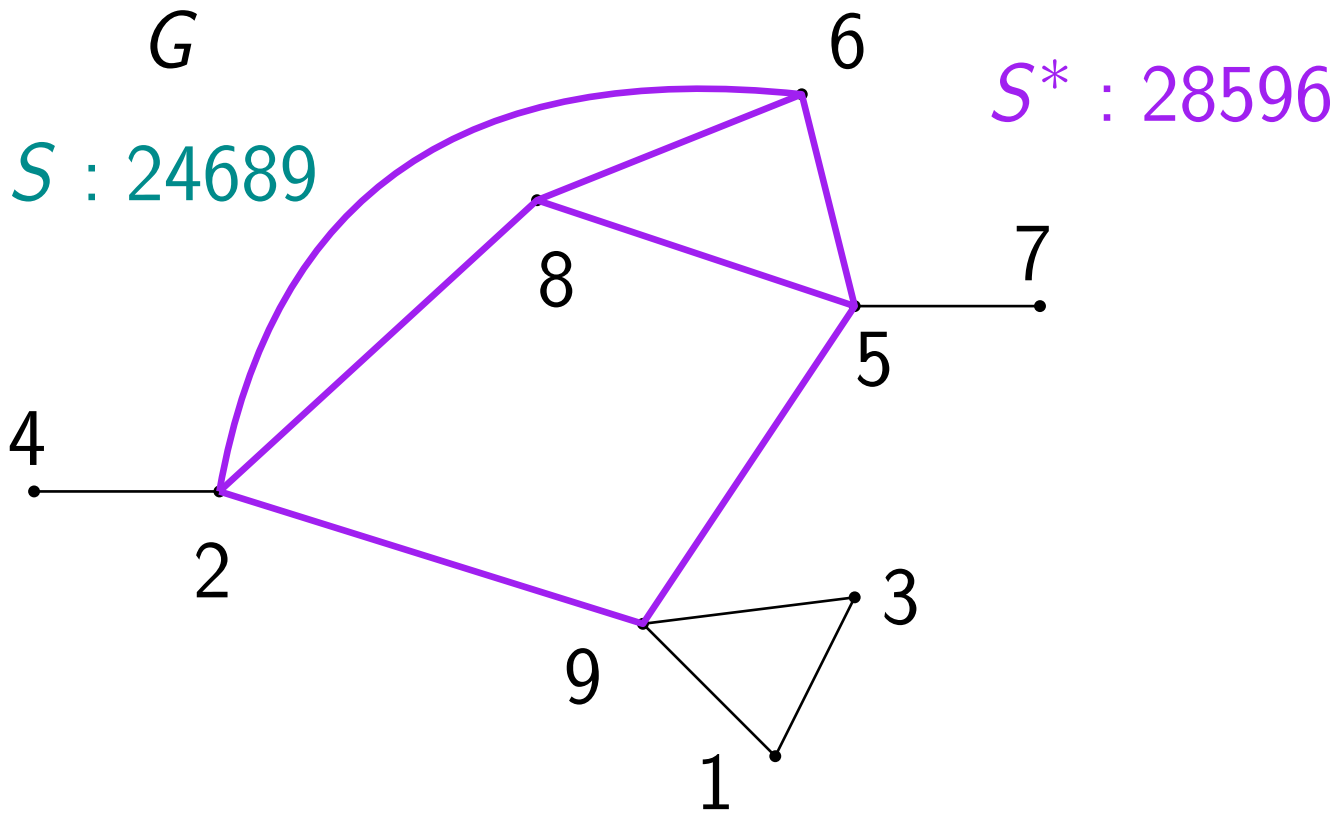
Proximité entre deux sous-cographes

But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



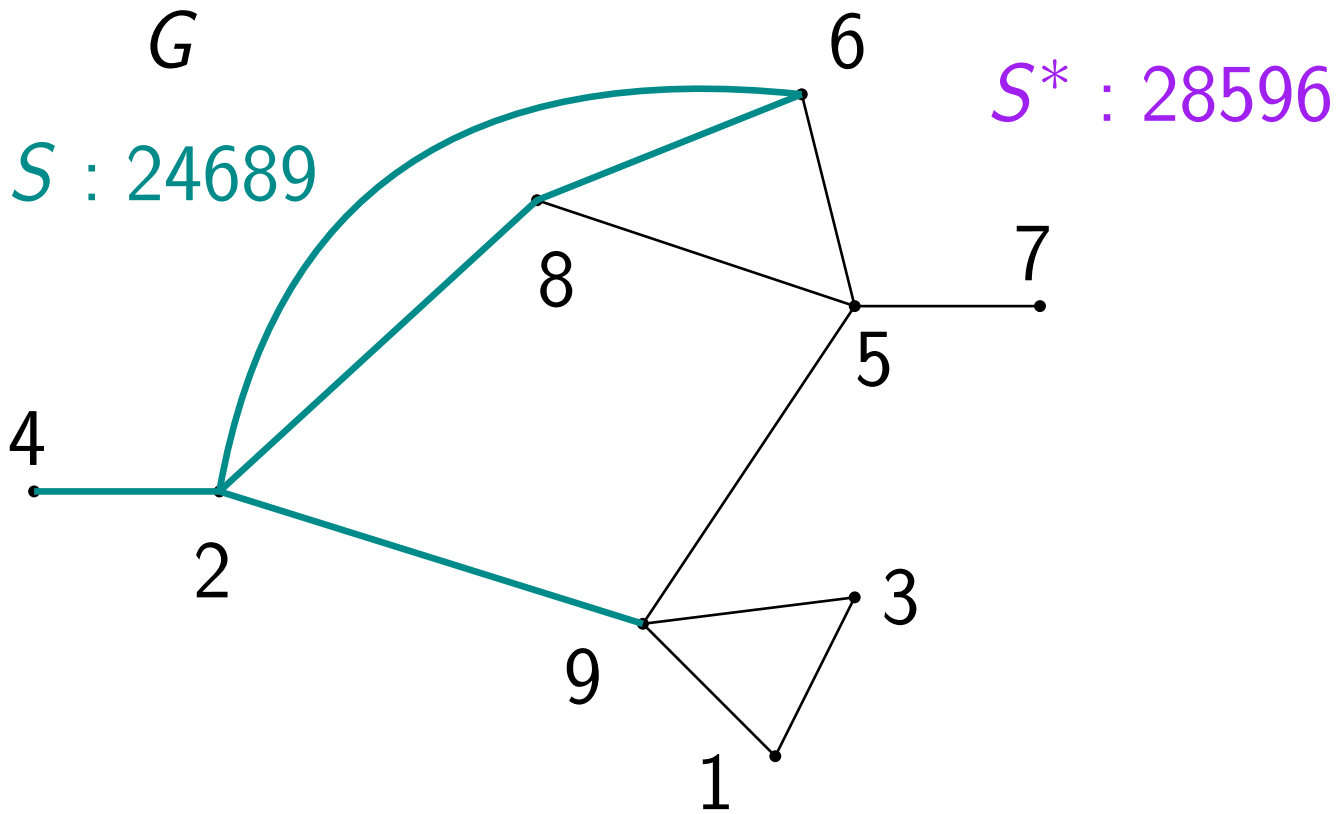
Proximité entre deux sous-cographes

But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



Proximité entre deux sous-cographes

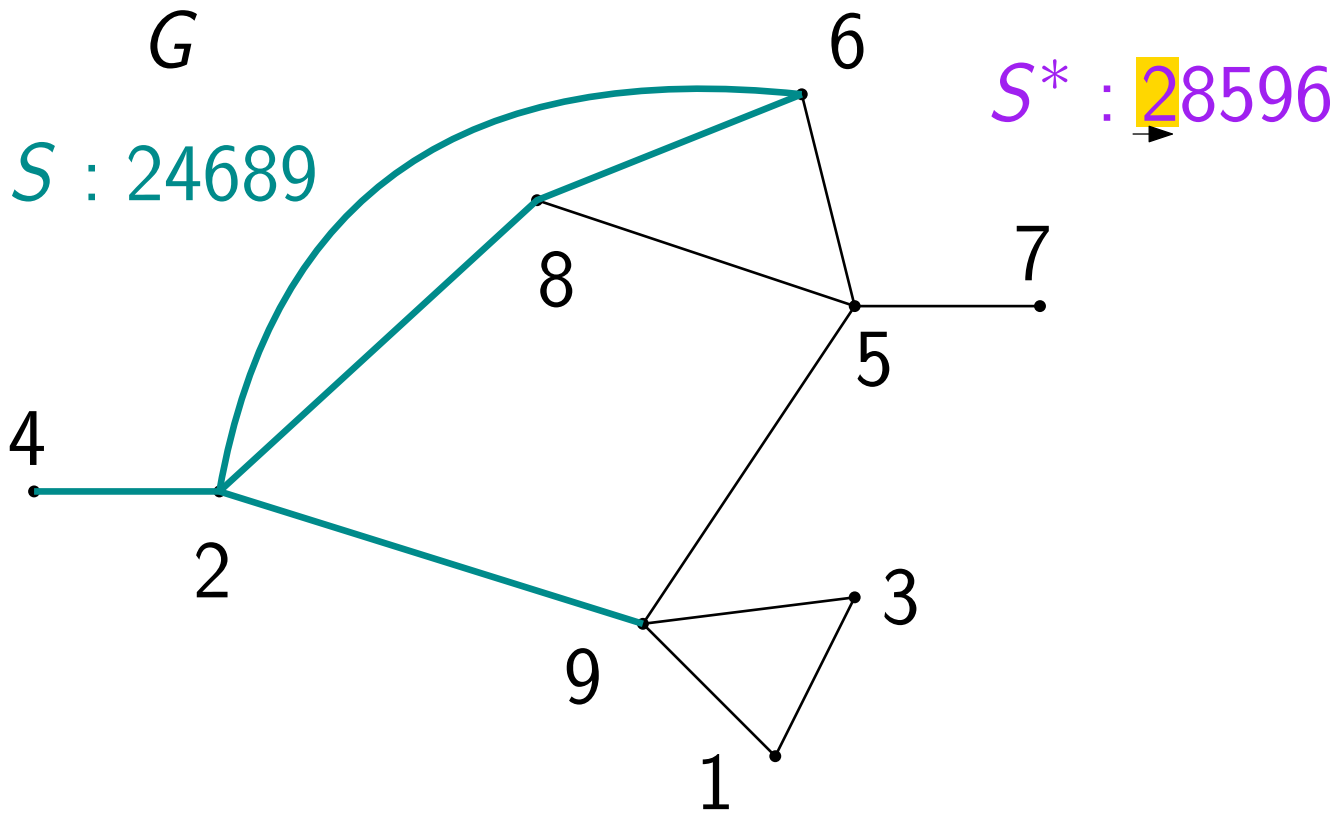
But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



$S \tilde{\cap} S^* = \{ \quad \}$

Proximité entre deux sous-cographes

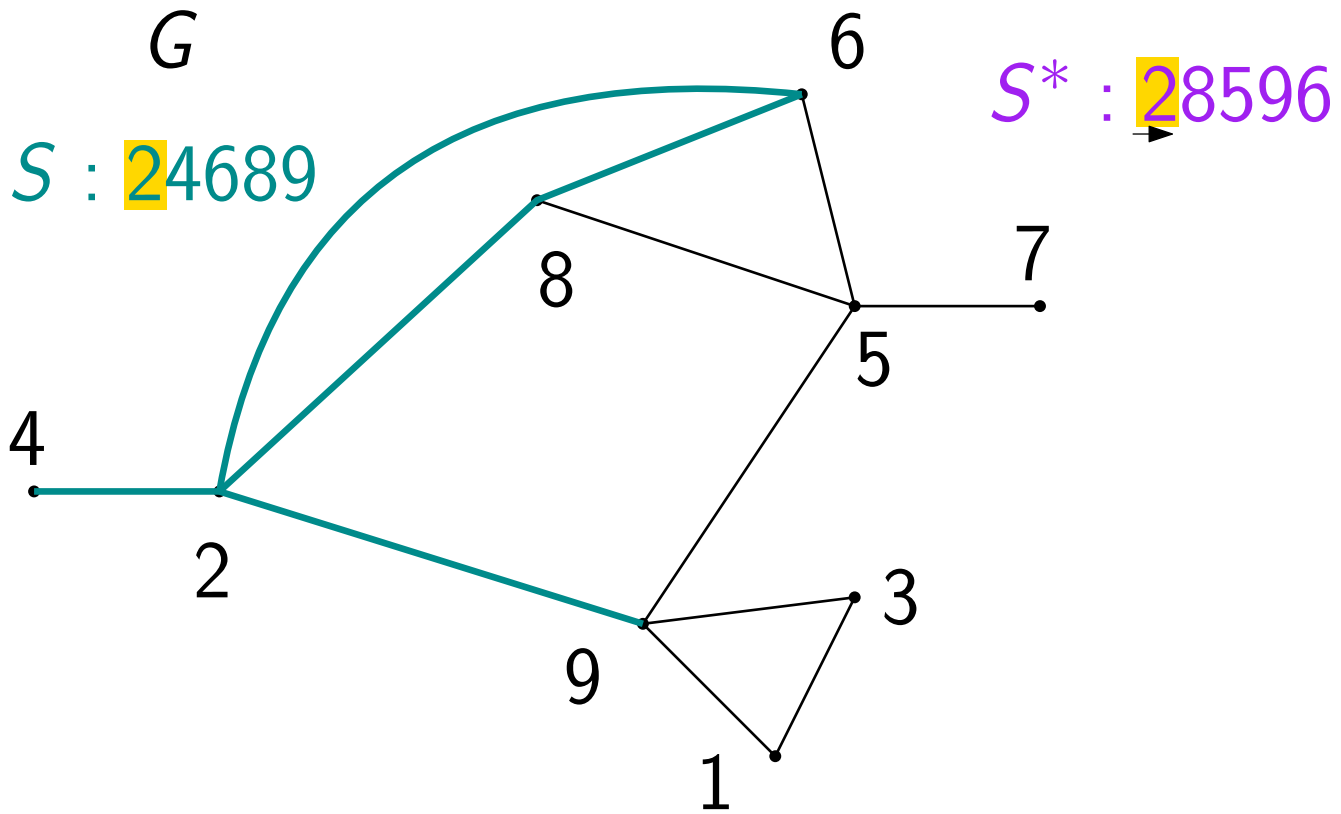
But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



$S \tilde{\cap} S^* = \{\quad\quad\quad\quad\}$

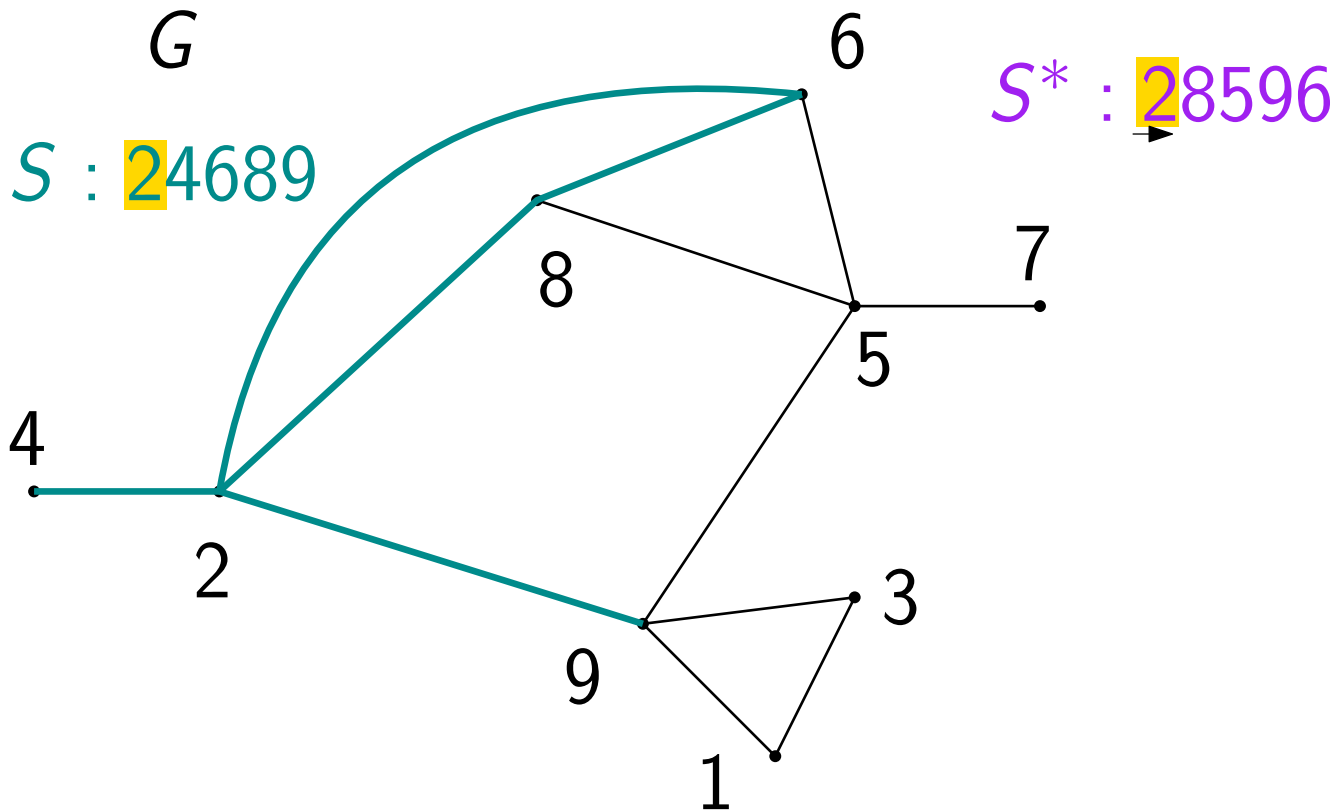
Proximité entre deux sous-cographes

But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



Proximité entre deux sous-cographes

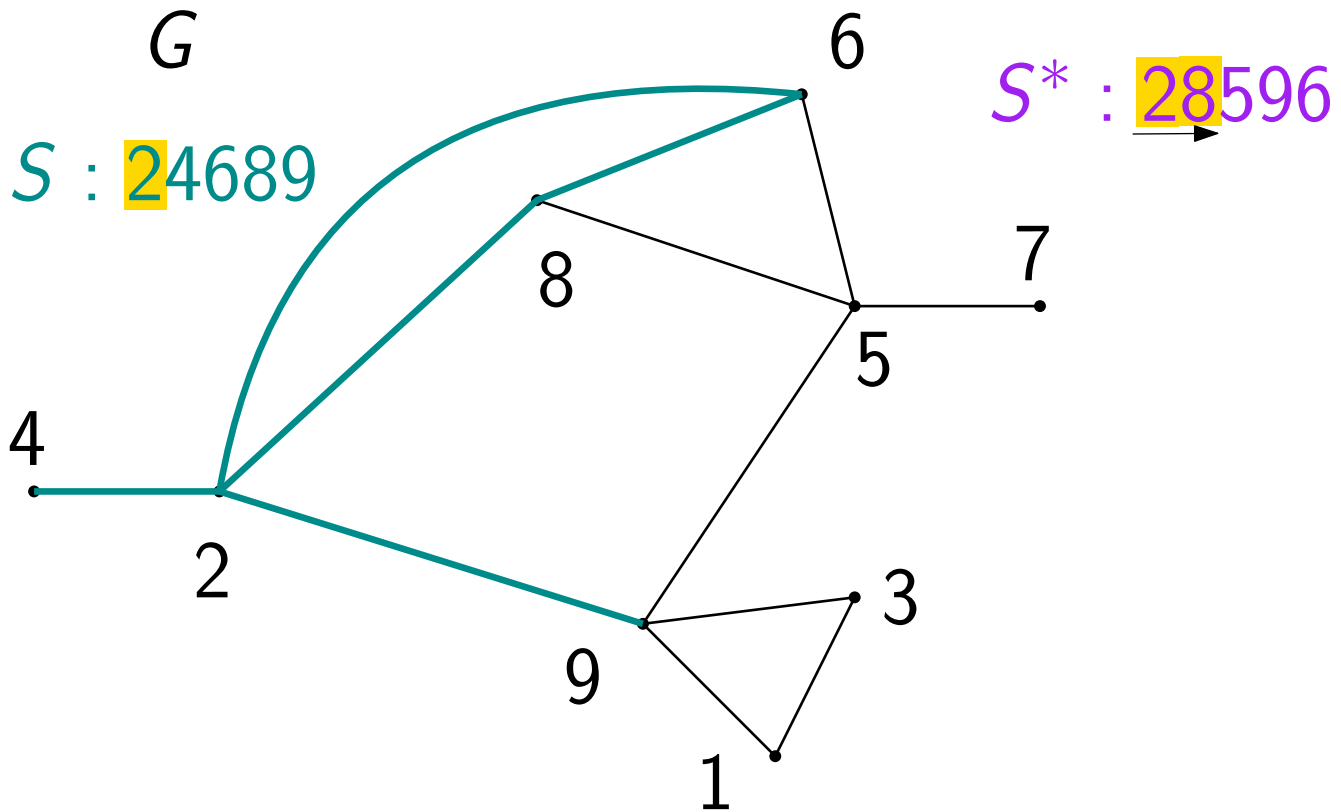
But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



$S \tilde{\cap} S^* = \{ 2 \quad \}$

Proximité entre deux sous-cographes

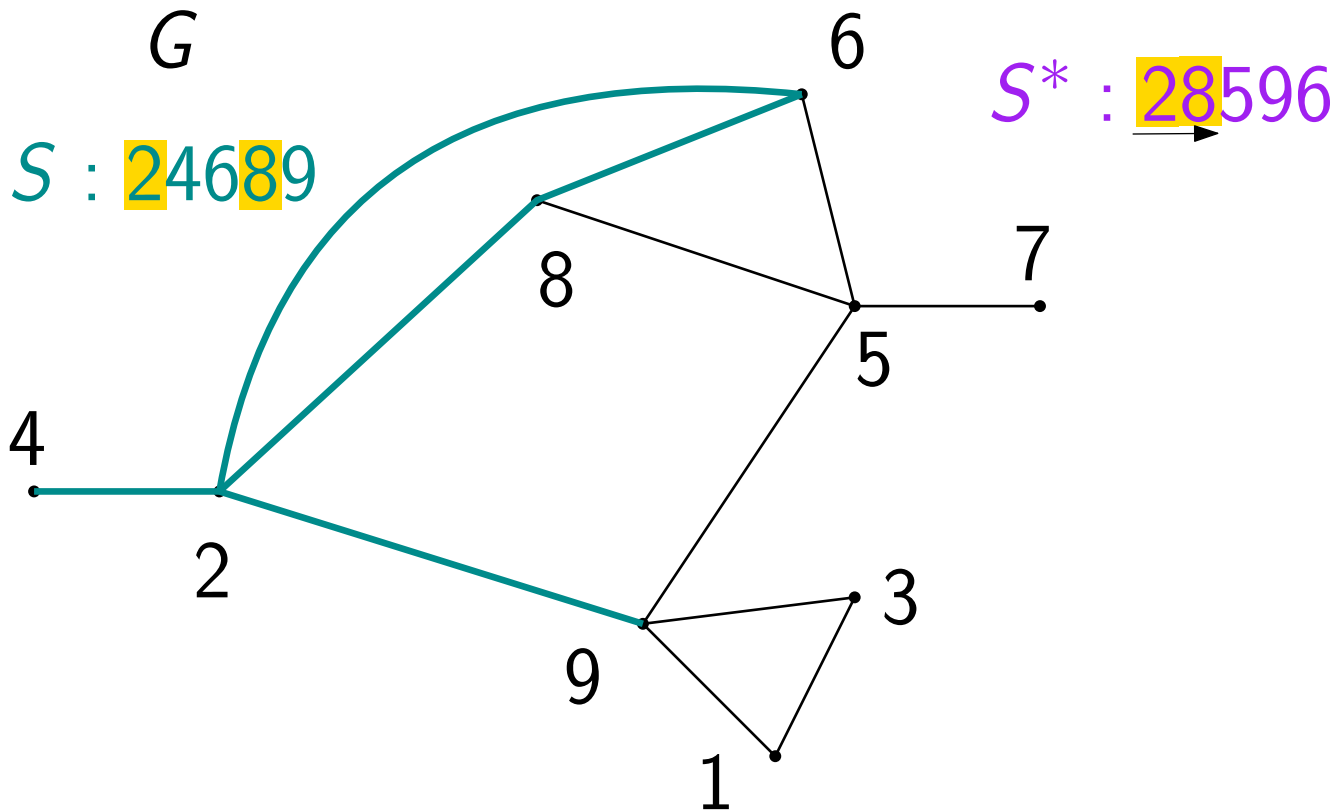
But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



$S \tilde{\cap} S^* = \{ 2 \quad \}$

Proximité entre deux sous-cographes

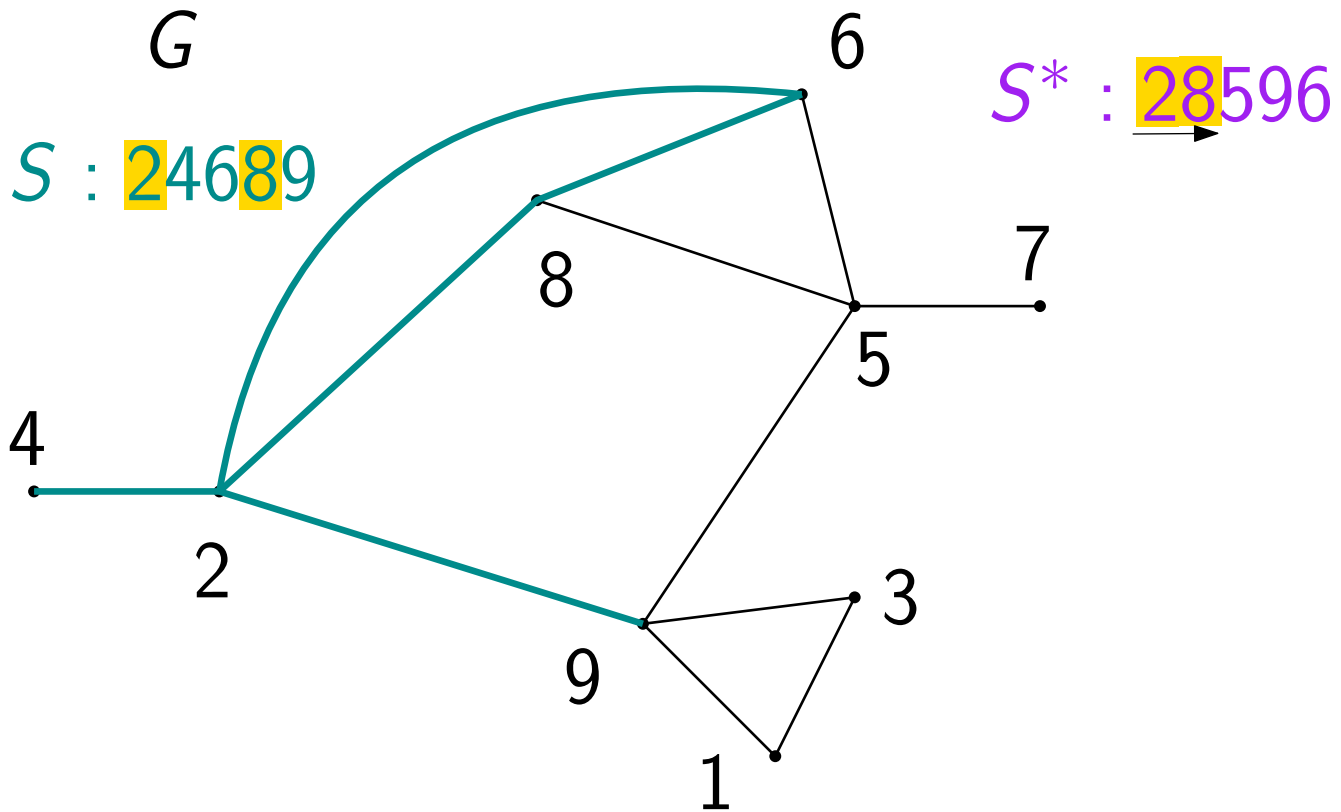
But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



$S \tilde{\cap} S^* = \{ 2 \quad \}$

Proximité entre deux sous-cographes

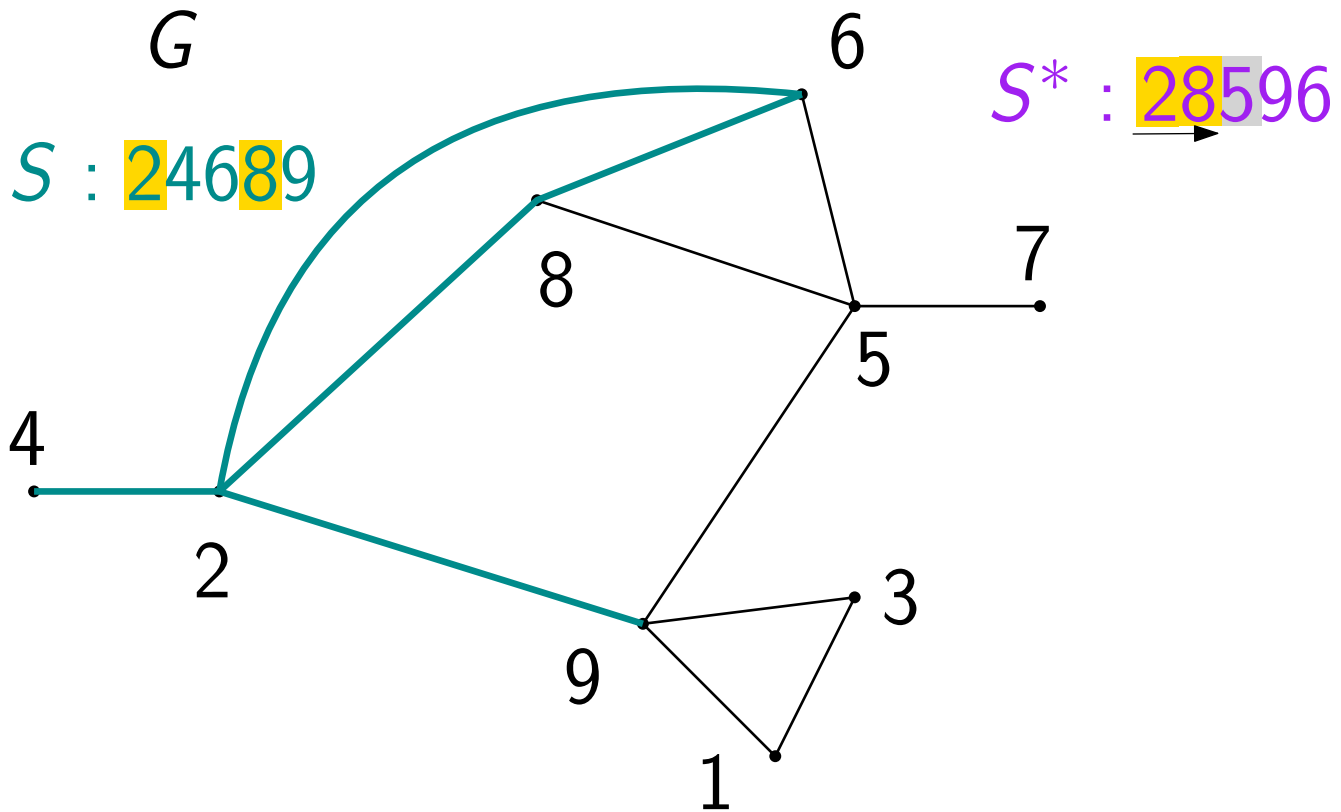
But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



$S \tilde{\cap} S^* = \{ 2, 8 \}$

Proximité entre deux sous-cographes

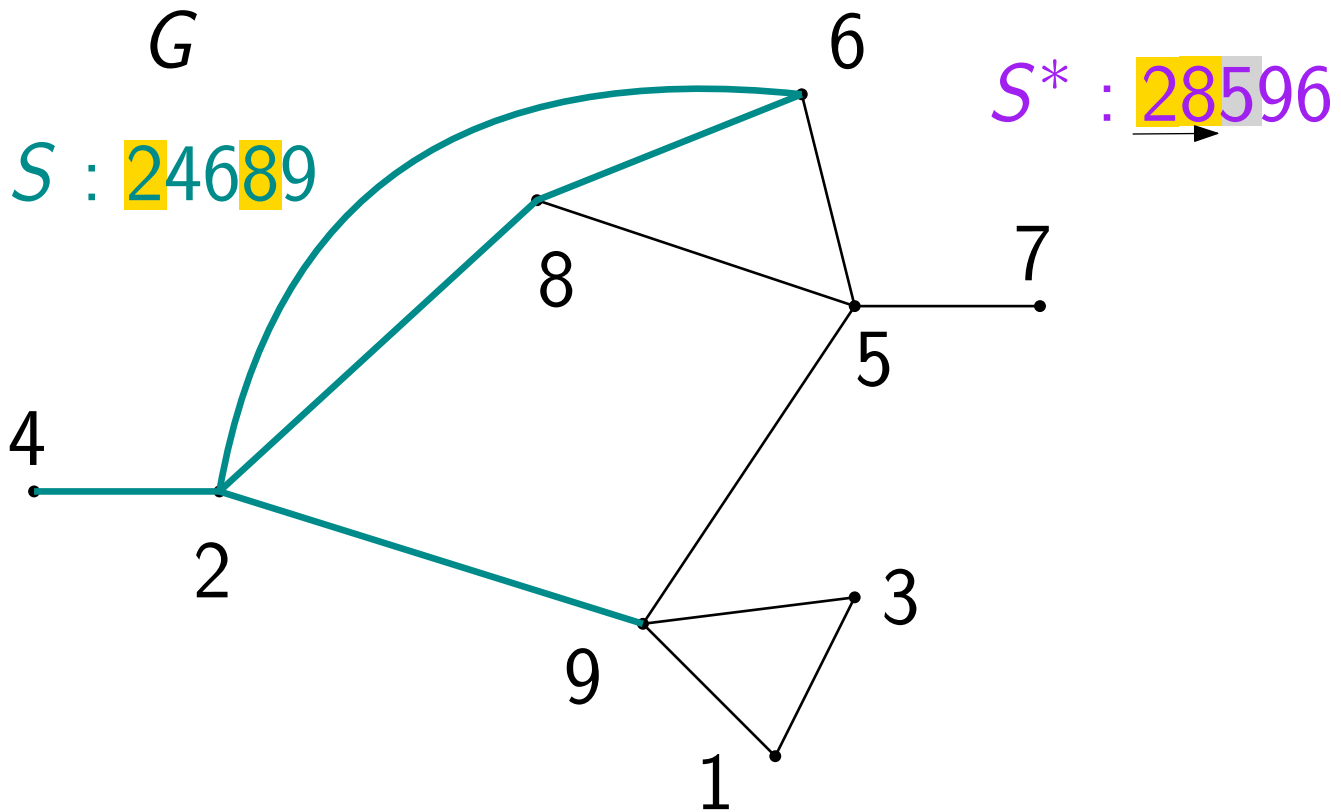
But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



$S \tilde{\cap} S^* = \{ 2, 8 \}$

Proximité entre deux sous-cographes

But : énumérer les sous-graphes induits maximaux de G qui sont des cographes avec *Proximity Search*



$S \tilde{\cap} S^* = \{ 2, 8 \} \rightarrow |\text{prox}|\ 2$

Voisins dans le graphe des solutions

S : 24689 $\quad$ S^* : 28596

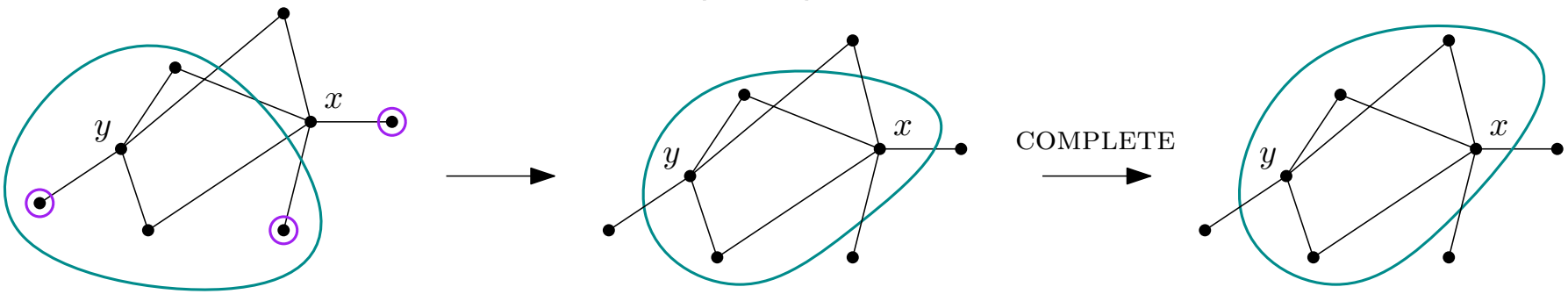
S': en conservant 2 et 8, faire rentrer 5 "de force" comme (faux) jumeau de 2

Voisins dans le graphe des solutions

S : 24689 $\quad$ S^* : 28596

S': en conservant 2 et 8, faire rentrer 5 "de force" comme (faux) jumeau de 2

Faire entrer x de force comme (faux) jumeau de y:



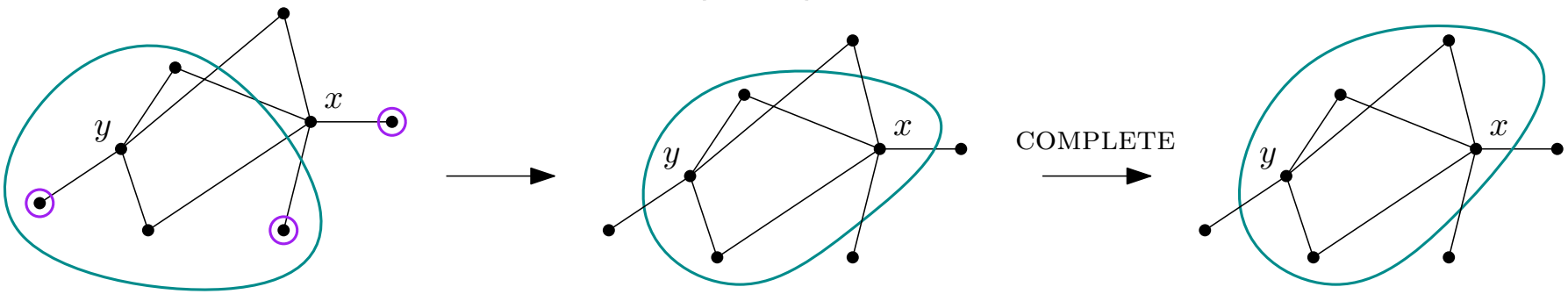
S $\qquad$ $S \setminus (N_G(x) \triangle N_S(y)) \cup \{x, y\}$ $\qquad$ S'_{xy}

Voisins dans le graphe des solutions

S : 24689 $\quad\quad$ S^* : 28596

S': en conservant 2 et 8, faire rentrer 5 "de force" comme (faux) jumeau de 2

Faire entrer x de force comme (faux) jumeau de y:



S $\quad\quad$ $S \setminus (N_G(x) \triangle N_S(y)) \cup \{x, y\}$ $\quad\quad$ S'_{xy}

$$Voisins(S) = \bigcup_{y \in S, x \notin S} S'_{xy}$$

Réparation en cographes : bilan

On a tous les ingrédients du Proximity search :

1. Proximité entre deux solutions S et S^*
2. $\mathit{Voisins}(S)$ calculable en temps poly
3. Lemme montrant qu'on peut toujours trouver un voisin augmentant la proximité avec une solution cible, grâce à l'ordre de construction par jumeau

$\rightarrow$ Énumération des réparations en cographe par suppression min. de sommets en délai polynomial

Résultats

[Brosse, L., Limouzy, Mary, Pastor – 2020+]

Prop. Π: being a....	Sous-graphes induits max.	Min. déletions	Min complétions
split graph	Délai & esp. poly [Cao'20$^+$]	Délai & esp. poly	Délai & esp. poly *via* autocompl.
cograph	délai poly. *via Proxi. Search*	délai poly. *via Proxi. Search*	délai poly. *via* autocompl.
P_3-free graph	Délai & esp. poly [Cao'20$^+$]	délai poly. *via Proxi. Search*	poly. delay trivial
threshold graph	Délai & esp. poly [Cao'20$^+$]	délai poly. *via Proxi. Search*	poly. delay *via* autocompl.
trivially perfect graph	Délai & esp. poly [Cao'20$^+$]	délai poly. *via Proxi. Search*	?

La plupart des résultats de [Cao20$^+$] sont via un autre méthode, le *restricted problem*.

Perspectives

Dégager des résultats généraux à partir de l'étude de plusieurs "petites" classes :

Perspectives

Dégager des résultats généraux à partir de l'étude de plusieurs "petites" classes :

→ Enumérer les triangluations minimales d'un graphe ?

Perspectives

Dégager des résultats généraux à partir de l'étude de plusieurs "petites" classes :

→ Enumérer les triangluations minimales d'un graphe ?

Théorème [Brosse, Limouzy, Mary, 2021$^+$]

Il existe un algorithme en délai polynomial et espace polynomial pour énumérer les complétions cordales minimales d'un graphe G donné en entrée.

→ *Proximity Search* raffiné (notamment pour avoir l'espace polynomial)

Perspectives

Dégager des résultats généraux à partir de l'étude de plusieurs "petites" classes :

→ Enumérer les triangluations minimales d'un graphe ?

Théorème [Brosse, Limouzy, Mary, 2021$^+$]

Il existe un algorithme en délai polynomial et espace polynomial pour énumérer les complétions cordales minimales d'un graphe G donné en entrée.

→ *Proximity Search* raffiné (notamment pour avoir l'espace polynomial)

→ Enumérer les réparations minimales d'un graphe en graphe d'intervalle ?

Ecole d'été School on Graph Theory

SGT' ~~20~~ ~~21~~ 22 Murol, Puy-De-Dôme

→ 5-10 juin 2022

Merci de votre attention !