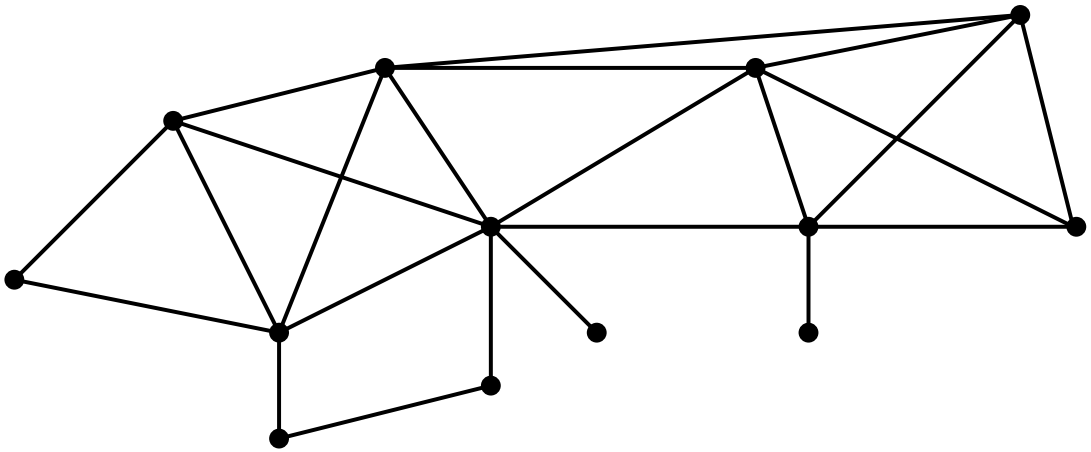
Graphes: structure et algorithmes

Aurélie LAGOUTTE

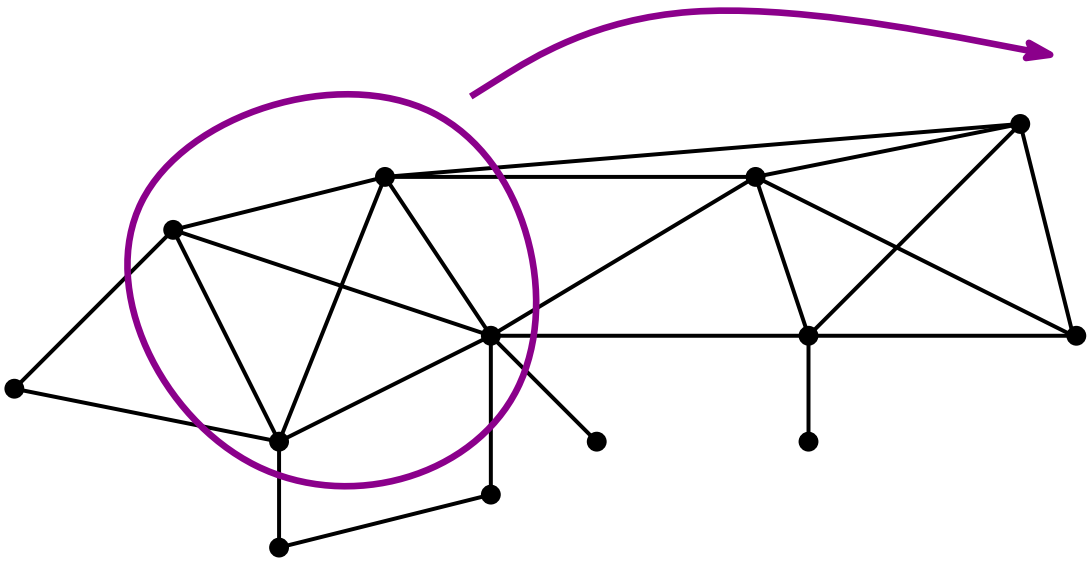
LIMOS, Université Clermont Auvergne

- 2012-2015 Thèse au LIP, ENS de Lyon, dans l'équipe MC2, sous la direction de Stéphan Thomassé

Problèmes d'optimisation dans les graphes



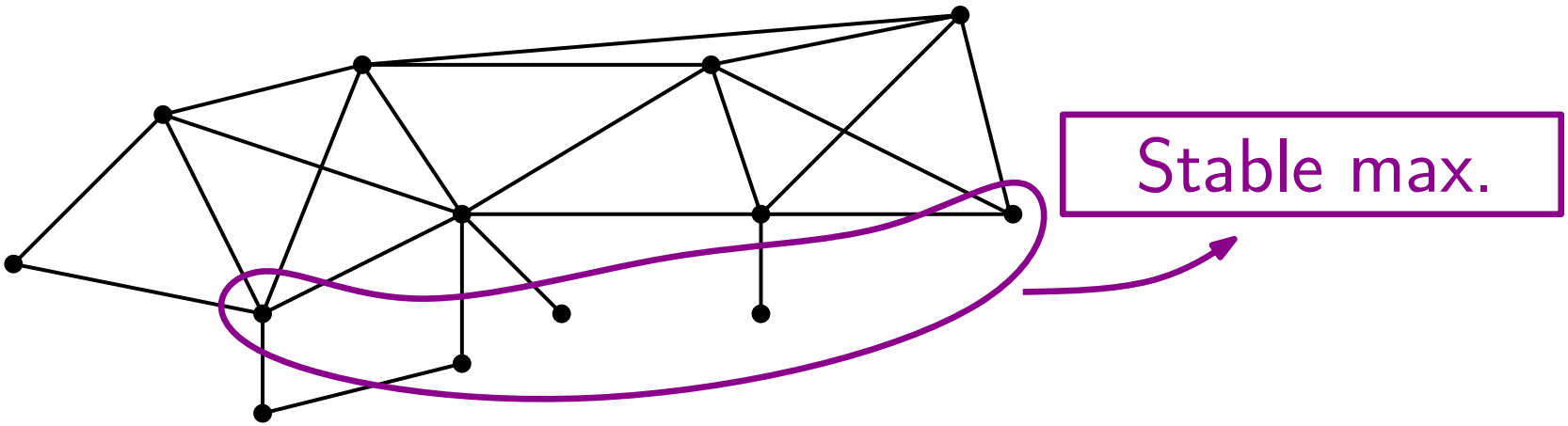
Problèmes d'optimisation dans les graphes



Clique max.

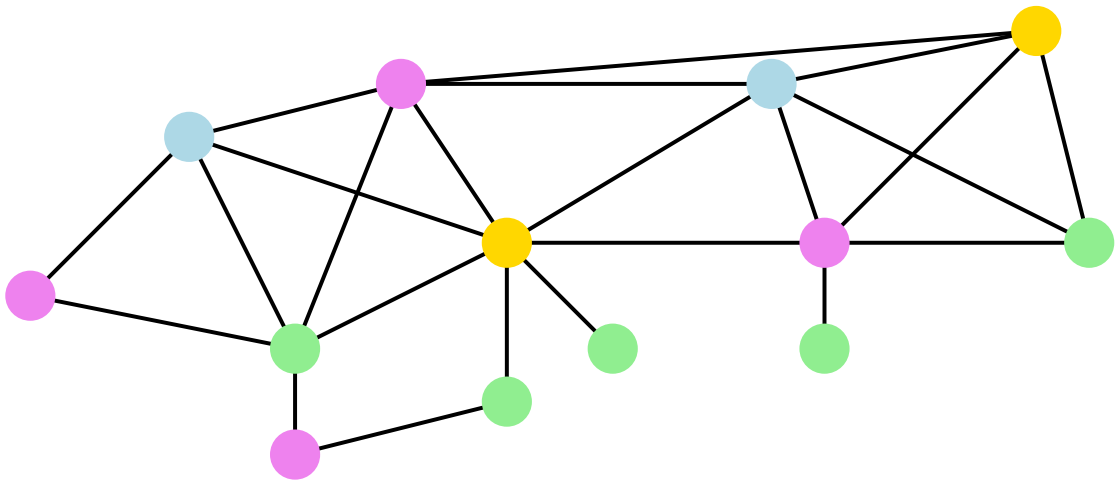
- ⊕ Modélisation de nombreux problèmes
  Ex : Zone de conflit maximal
- ⊖ NP-complet en général

Problèmes d'optimisation dans les graphes



- ⊕ Modélisation de nombreux problèmes
  Ex : Assister à des événements non chevauchants
- ⊖ NP-complet en général

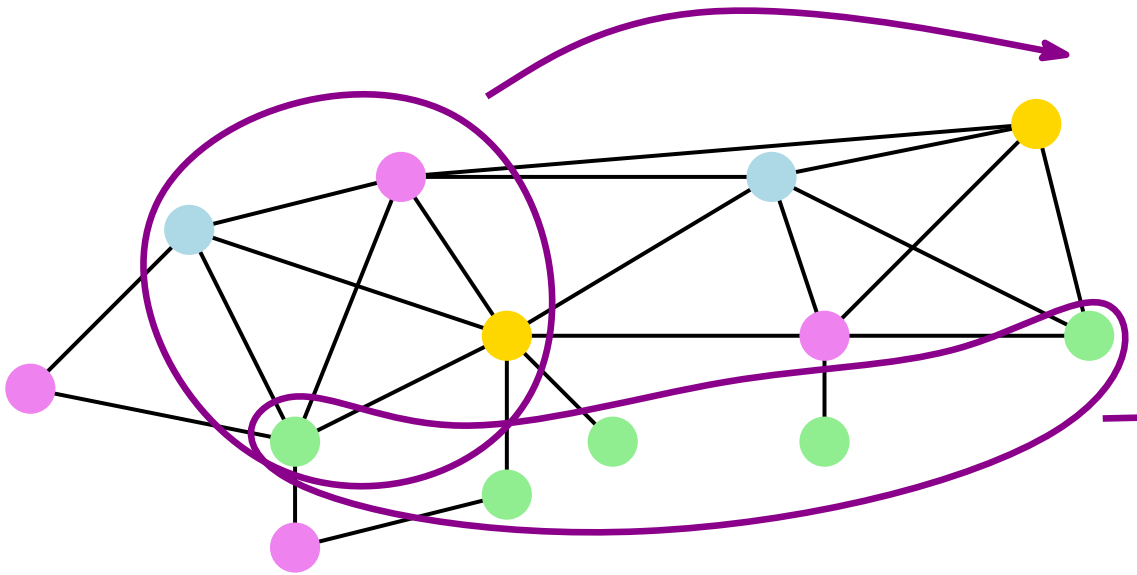
Problèmes d'optimisation dans les graphes



Coloration opt.

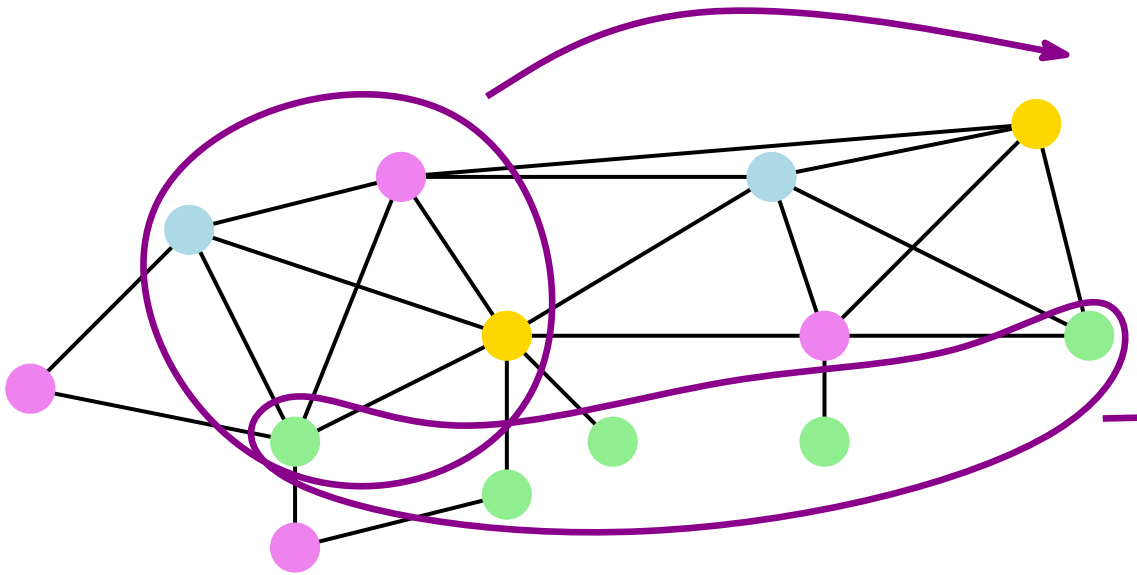
- ⊕ Modélisation de nombreux problèmes
  Ex : Allocation de ressources
- ⊖ NP-complet en général

Problèmes d'optimisation dans les graphes



- ⊕ Modélisation de nombreux problèmes
- ⊖ NP-complet en général

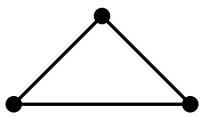
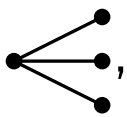
Problèmes d'optimisation dans les graphes



⊕ Modélisation de nombreux problèmes

⊖ NP-complet en général

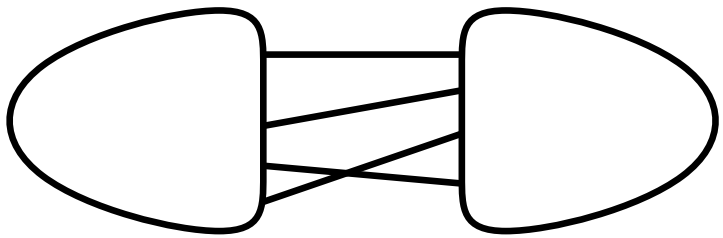
→ Et dans les graphes "structurés" ?

Par exemple, les graphes **ne contenant pas**  ou  ,

les **graphes parfaits**...

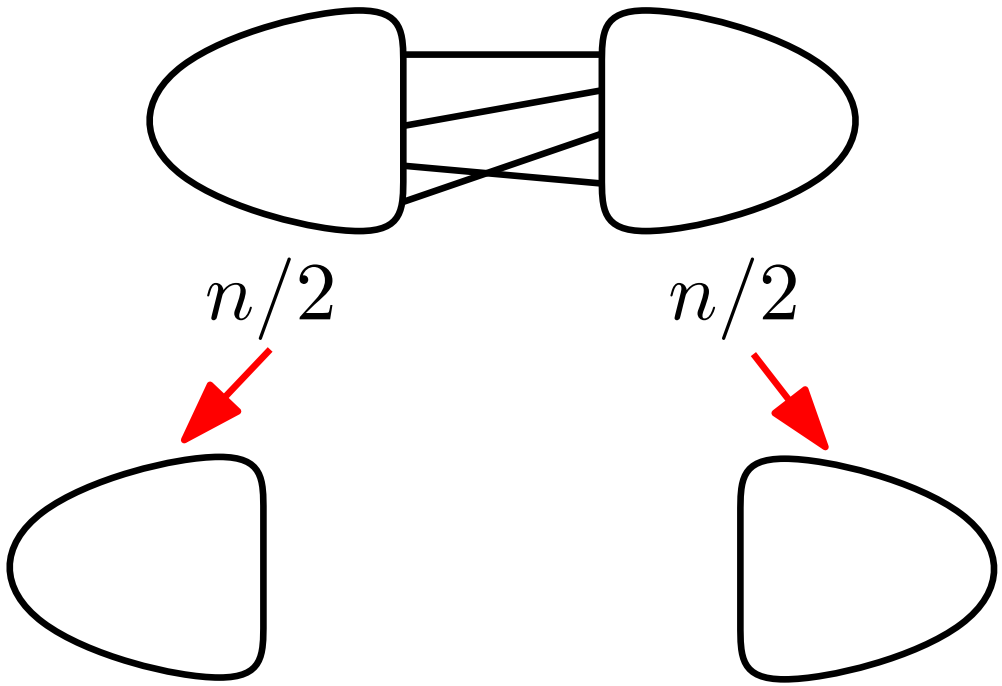
Décompositions

Approche "Diviser pour régner"



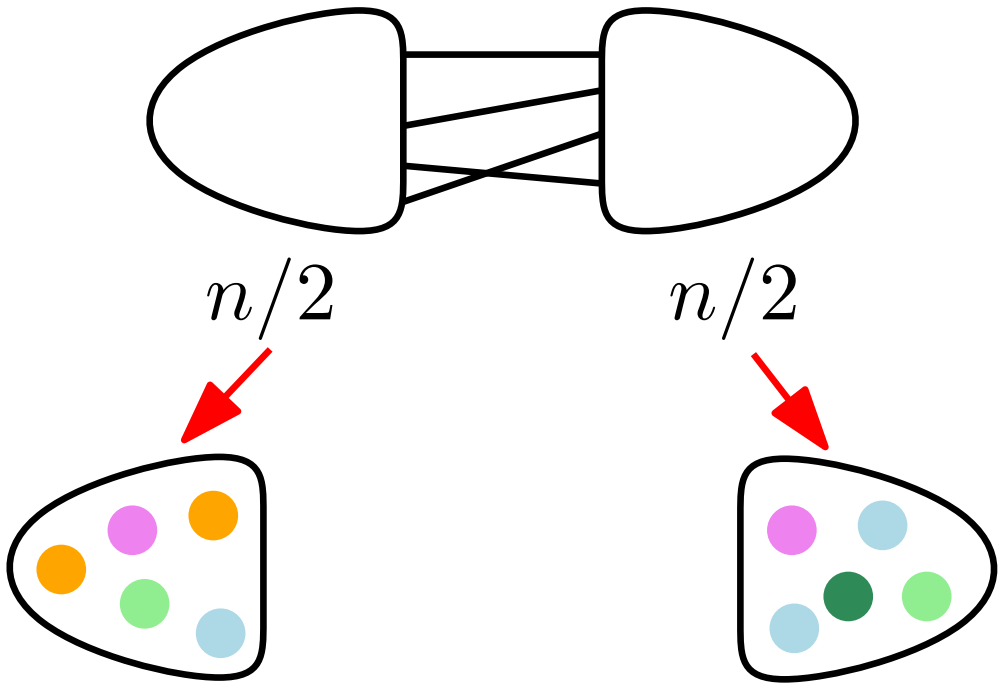
Décompositions

Approche "Diviser pour régner"



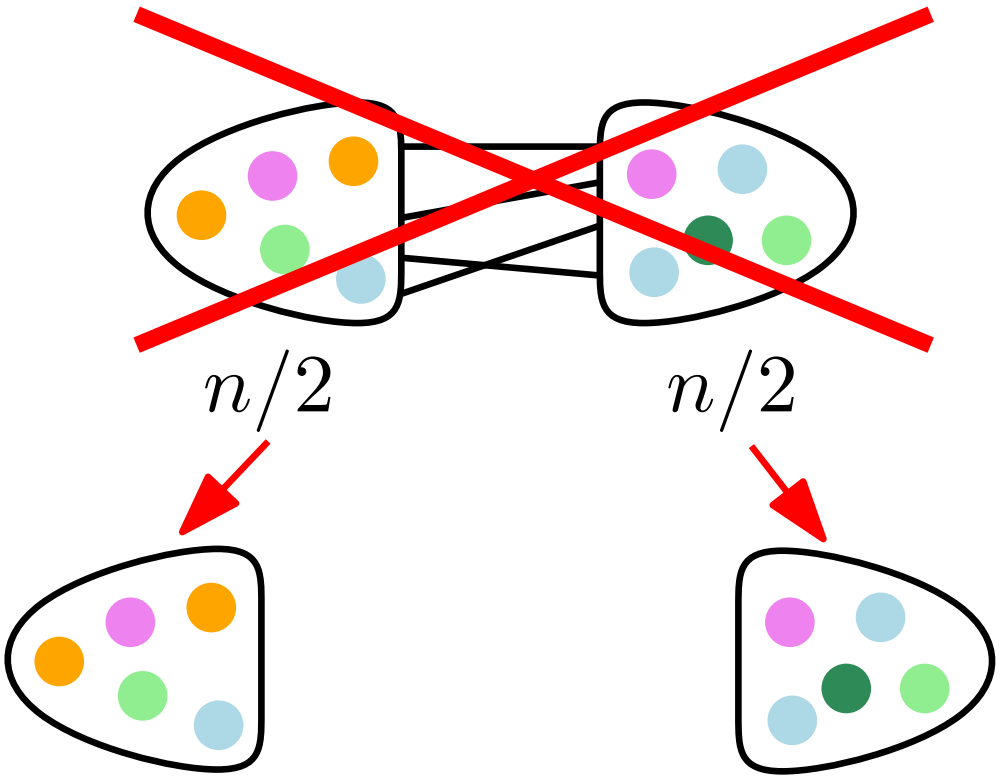
Décompositions

Approche "Diviser pour régner"



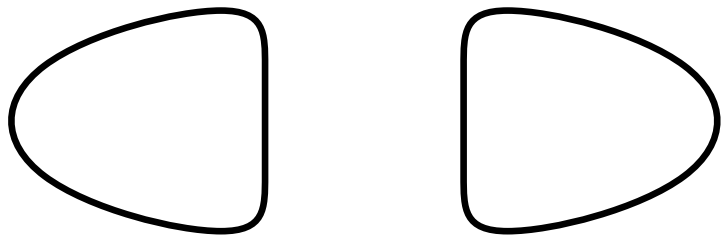
Décompositions

Approche "Diviser pour régner"



Décompositions

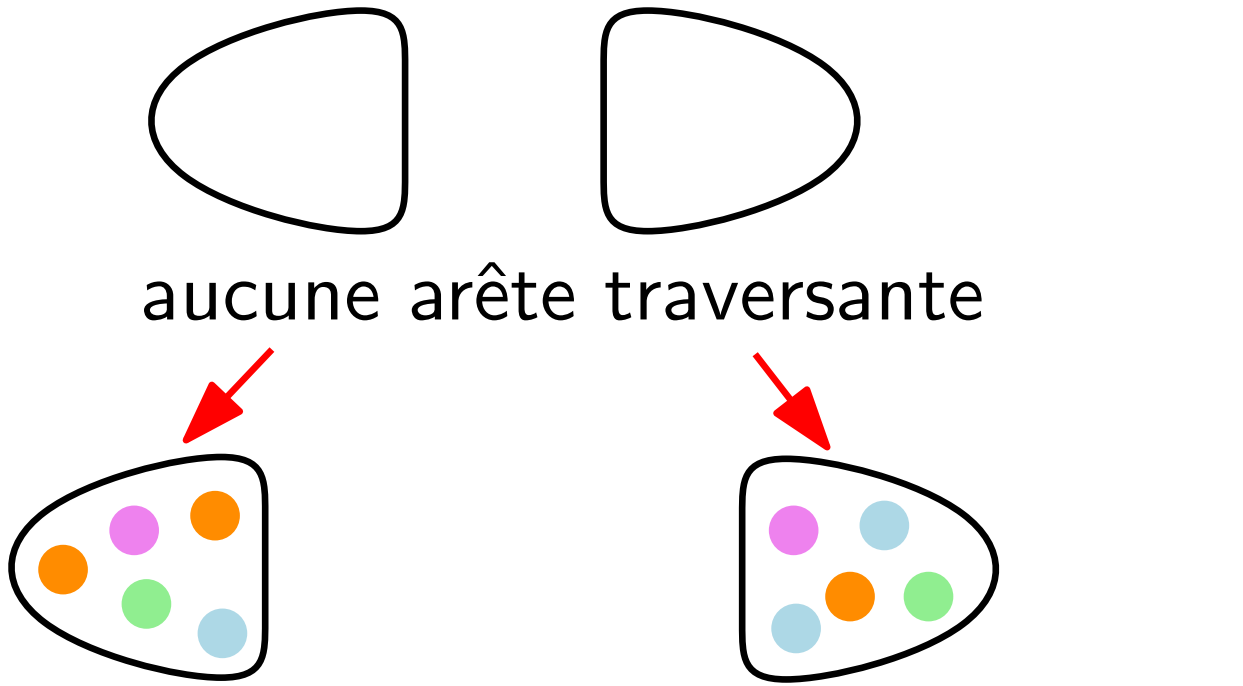
Approche "Diviser pour régner"



aucune arête traversante

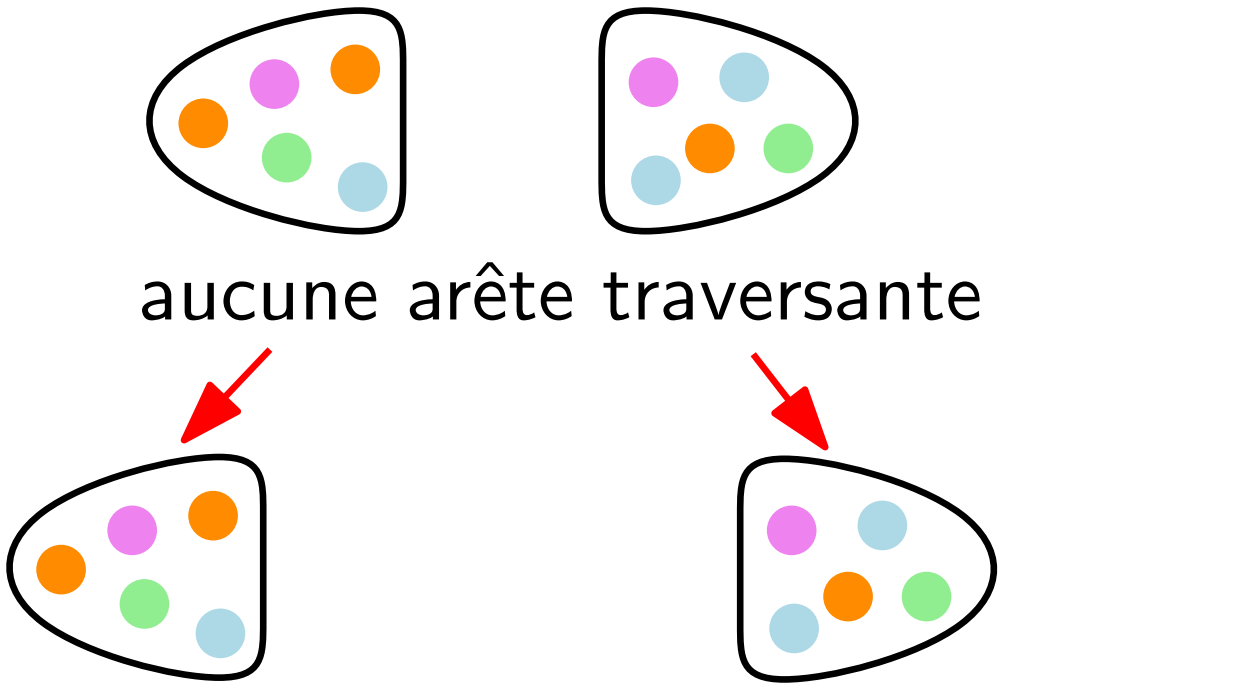
Décompositions

Approche "Diviser pour régner"



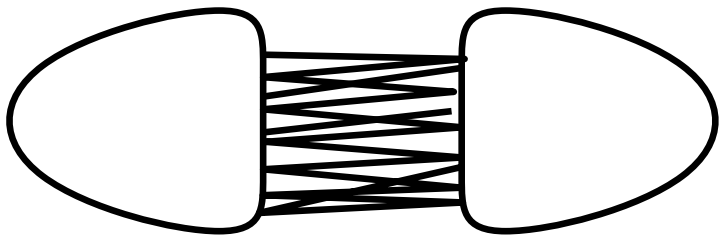
Décompositions

Approche "Diviser pour régner"



Décompositions

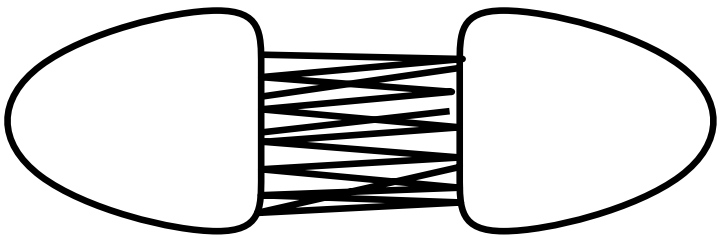
Approche "Diviser pour régner"



toutes les arêtes traversantes possibles

Décompositions

Approche "Diviser pour régner"

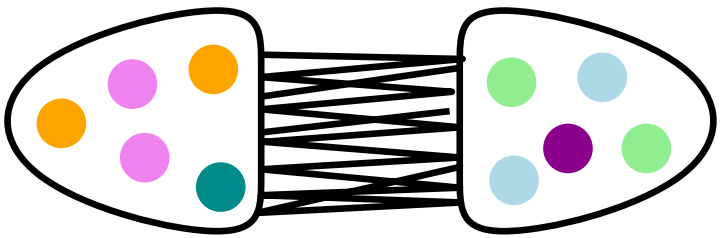


toutes les arêtes traversantes possibles

utiliser des ensembles de couleurs disjoints

Décompositions

Approche "Diviser pour régner"

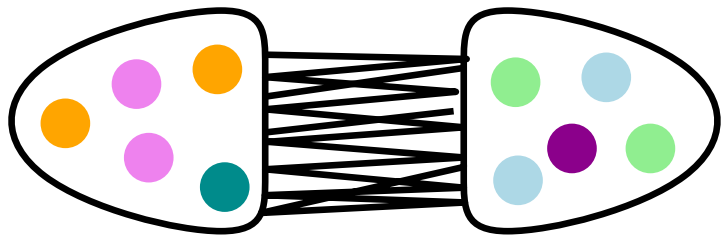


toutes les arêtes traversantes possibles

utiliser des ensembles de couleurs disjoints

Décompositions

Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes



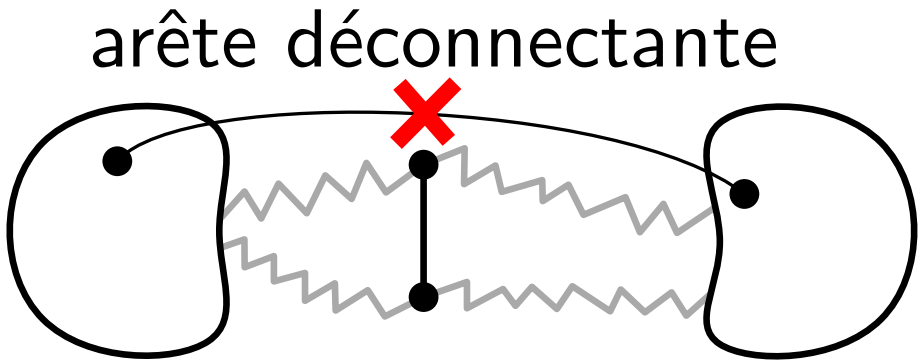
toutes les arêtes traversantes possibles

utiliser des ensembles de couleurs disjoints

Décompositions

Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes

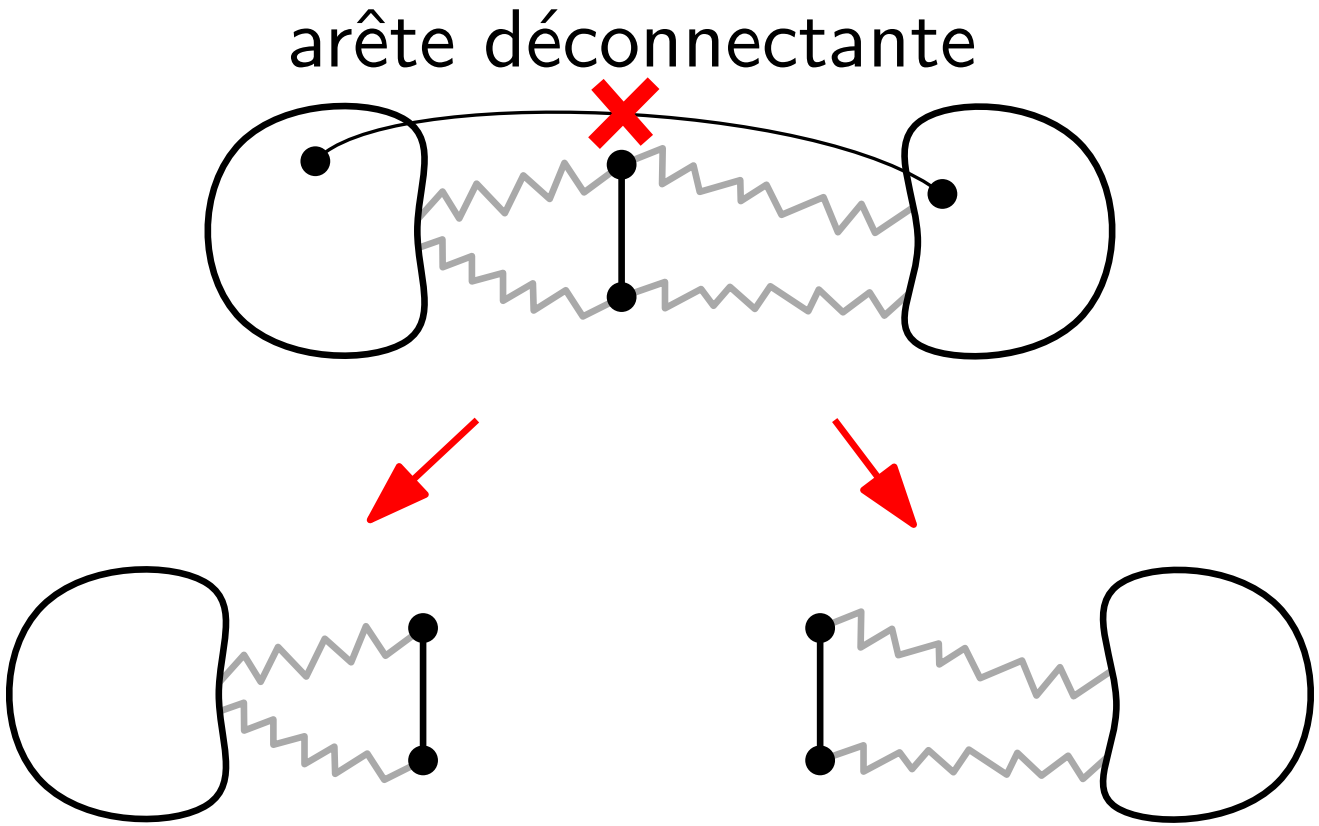
Compromis : s'autoriser à moins "diviser"



Décompositions

Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes

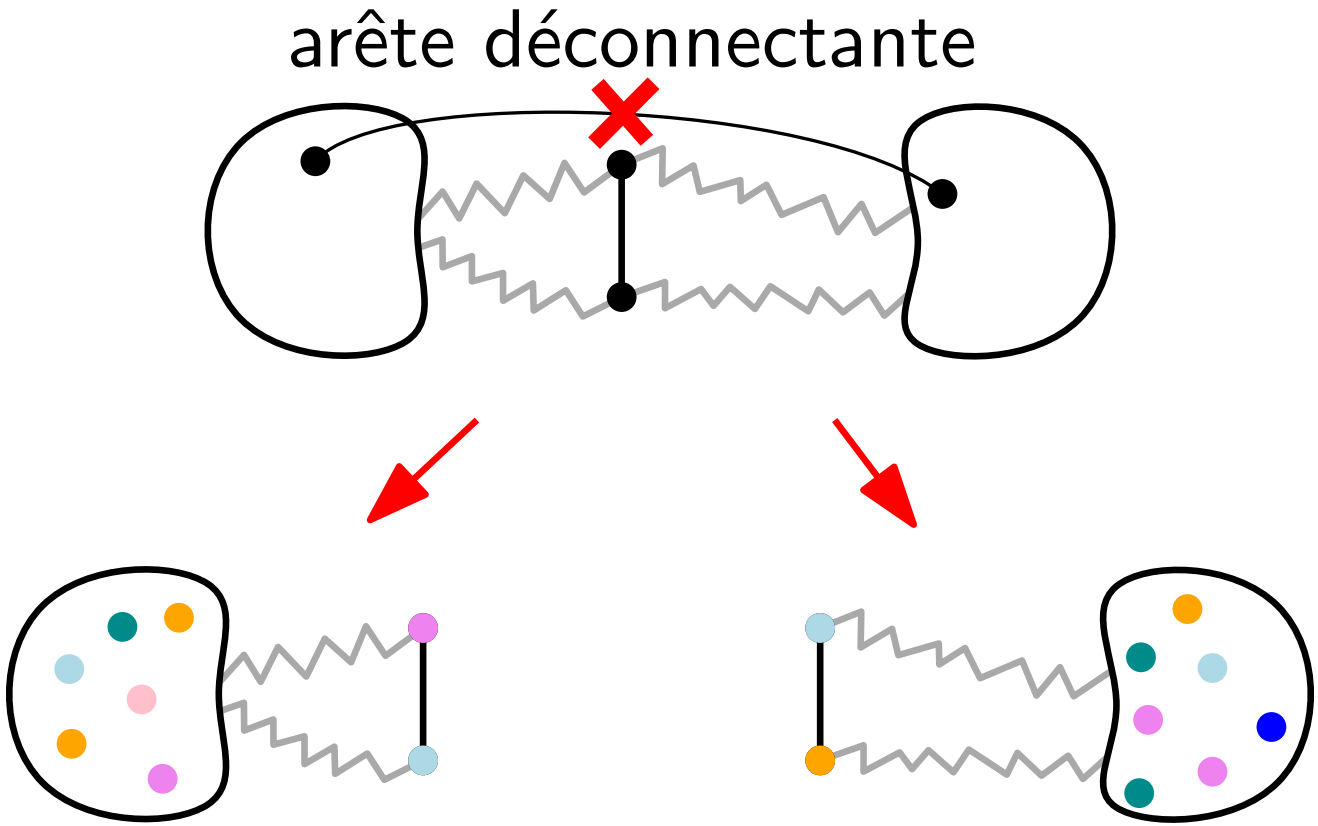
Compromis : s'autoriser à moins "diviser"



Décompositions

Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes

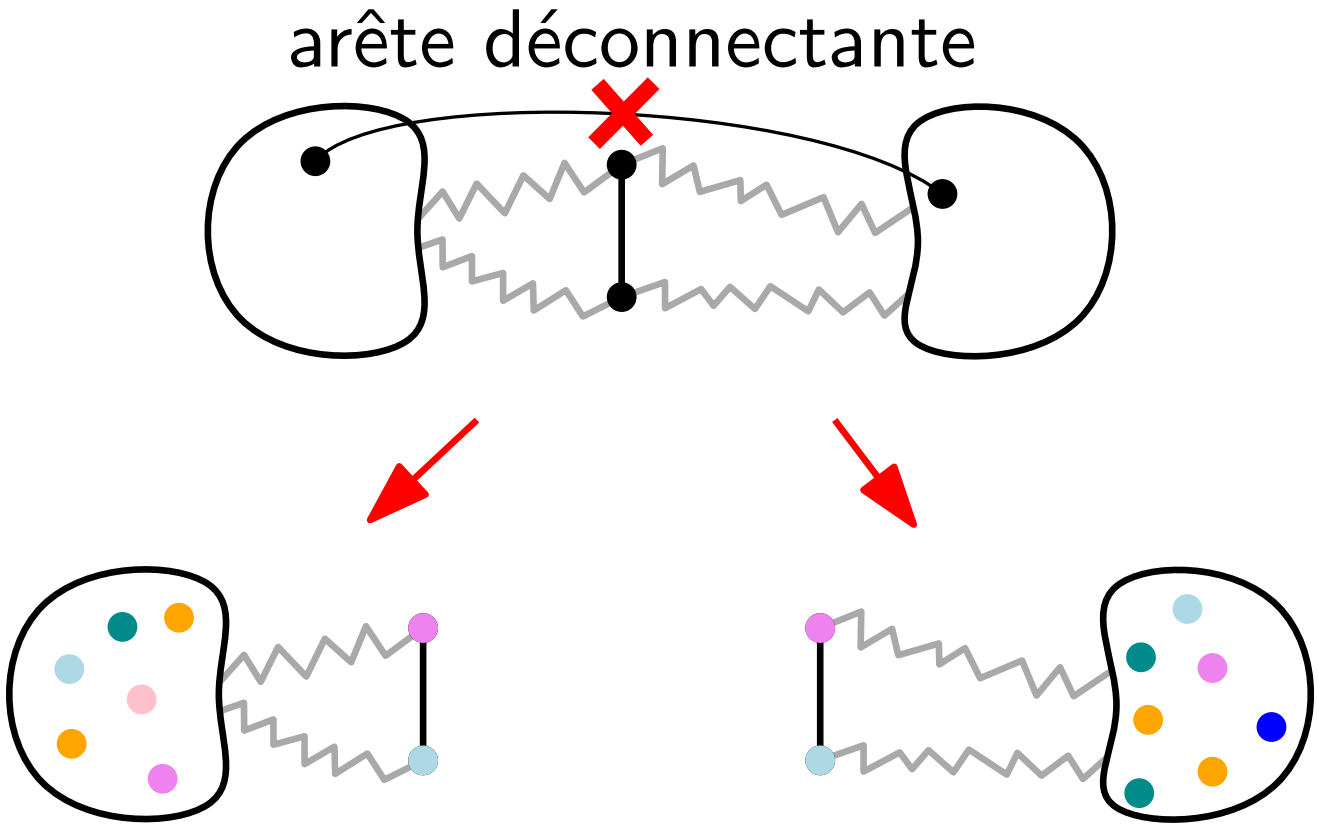
Compromis : s'autoriser à moins "diviser"



Décompositions

Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes

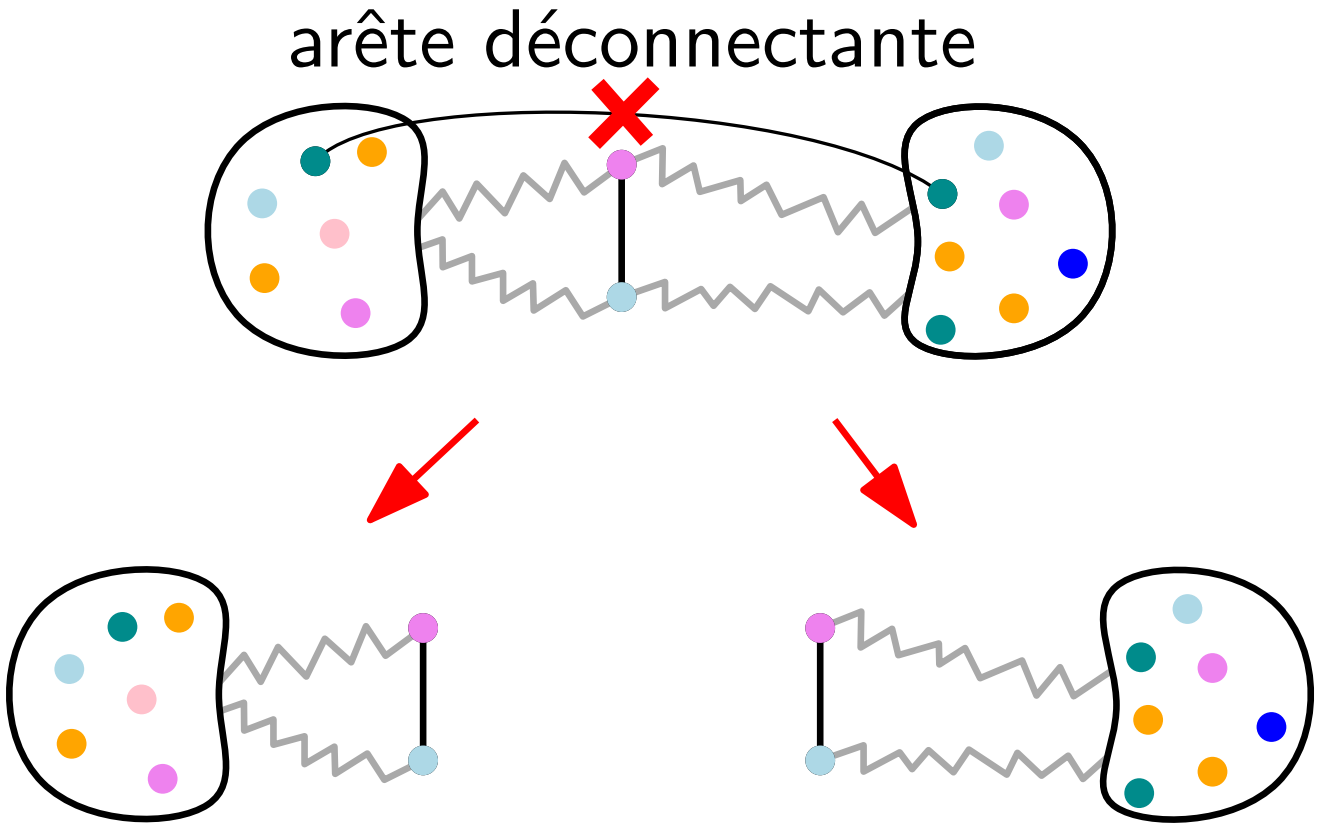
Compromis : s'autoriser à moins "diviser"



Décompositions

Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes

Compromis : s'autoriser à moins "diviser"



Méthodes de décompositions

Décompositions: à quoi ça sert?

Méthodes de décompositions

Décompositions: à quoi ça sert?

Faire des algos

Du type "diviser pour régner"

Méthodes de décompositions

Décompositions: à quoi ça sert?

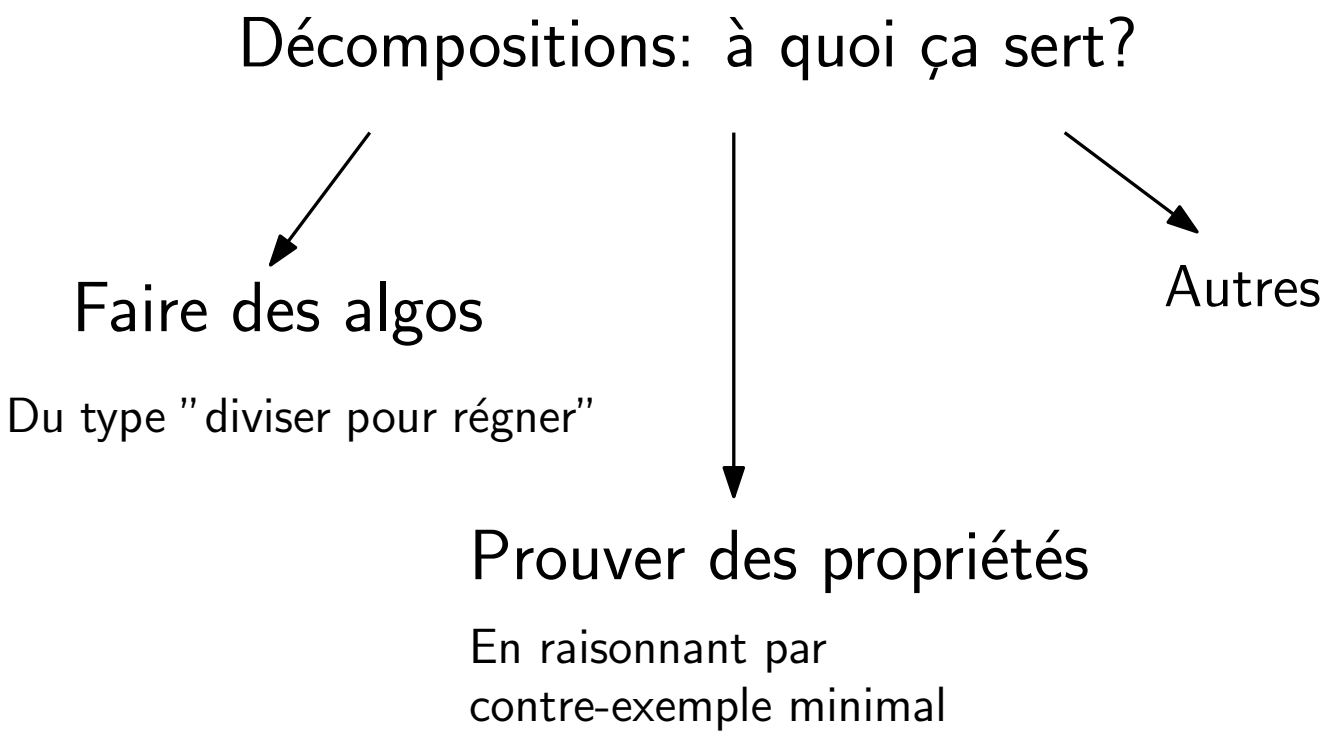
Faire des algos

Du type "diviser pour régner"

Prouver des propriétés

En raisonnant par
contre-exemple minimal

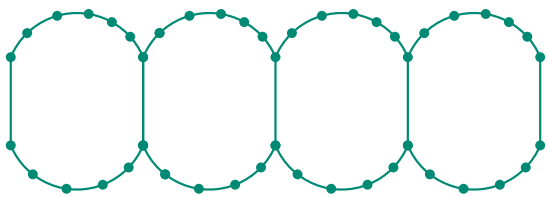
Méthodes de décompositions



Méthodes de décompositions

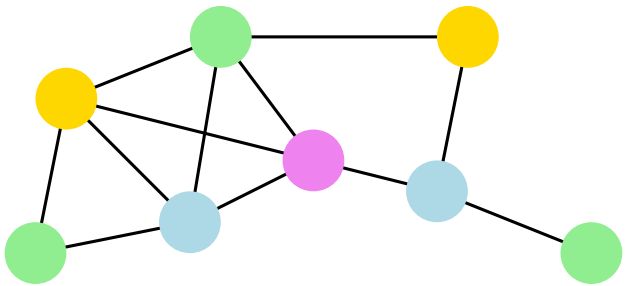
Combinaison de 3 ingrédients :

1. Choisir un problème : algo à écrire, propriété à montrer, ...
   Exemple : algorithme pour la coloration
2. Identifier les décompositions qui se comportent "bien" par rapport à ce problème-là
   Exemple : arête-déconnectante
3. Se restreindre à une classe structurée dans laquelle on peut (presque) toujours trouver une décomposition listée à l'item 2.

Exemple : 

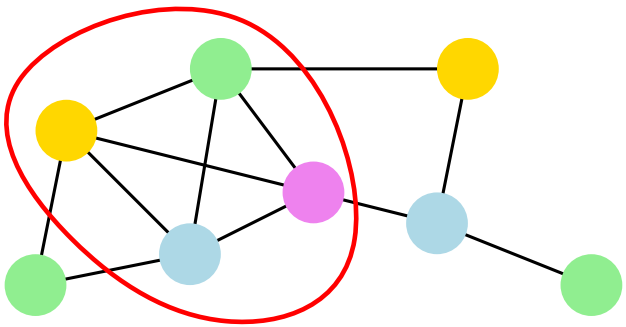
Graphes parfaits

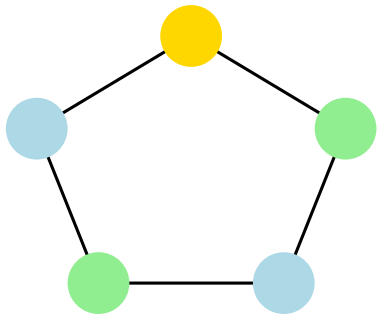
Certifier que beaucoup de couleurs sont nécessaires ?



Graphes parfaits

Certifier que beaucoup de couleurs sont nécessaires ?



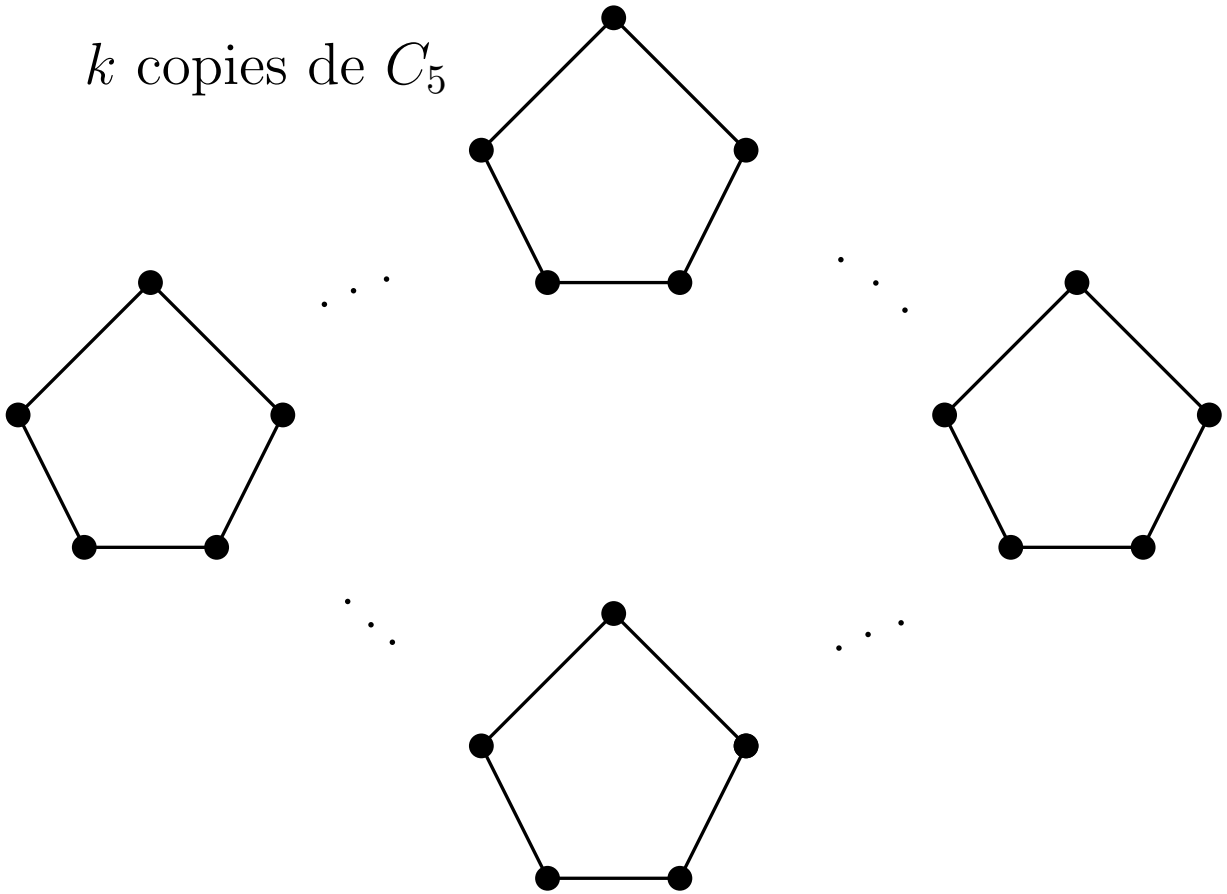


nb. couleurs $\geq$ max. clique nb. couleurs $>$ max. clique

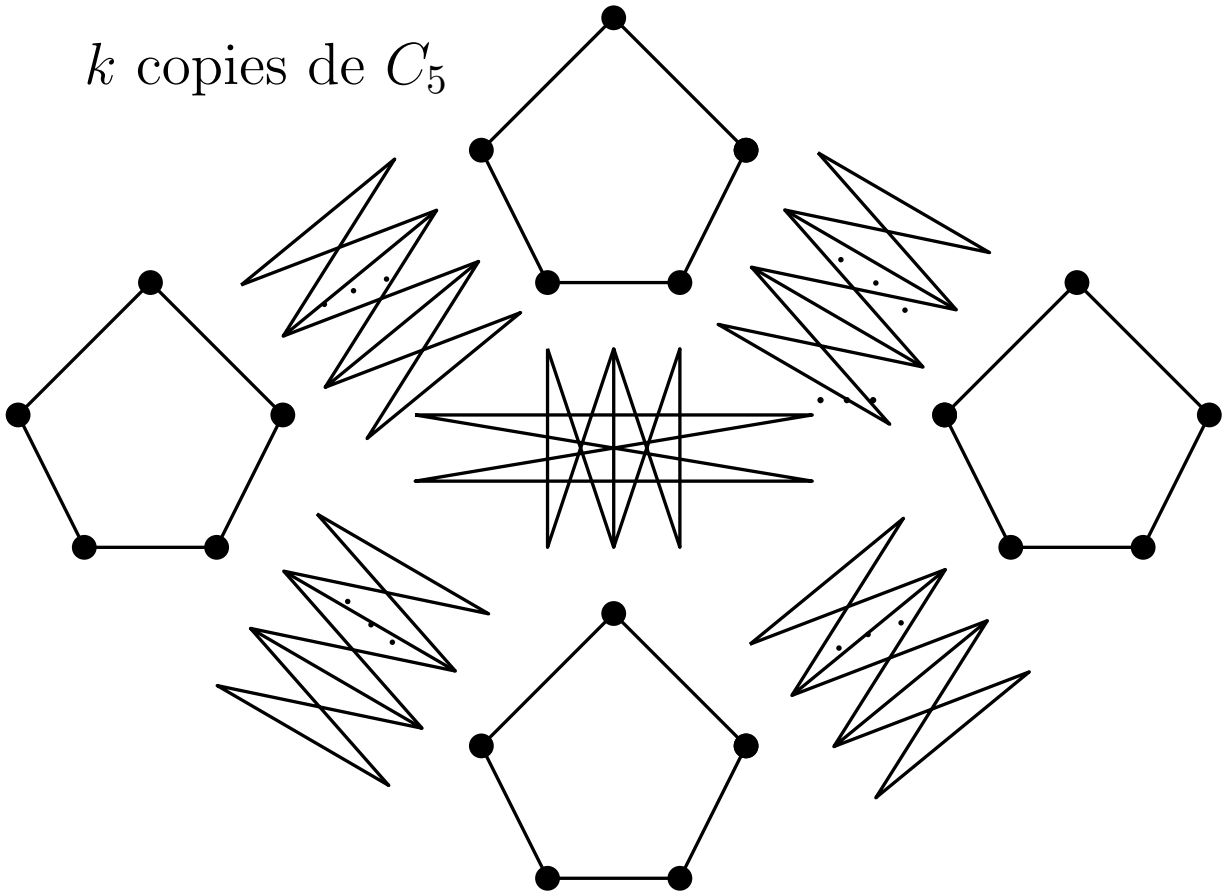
Si nb. couleurs $=$ max. clique $\Rightarrow$ le graphe est *parfait*.
(partout dans le graphe)

On peut même avoir un écart aussi grand que l'on veut !

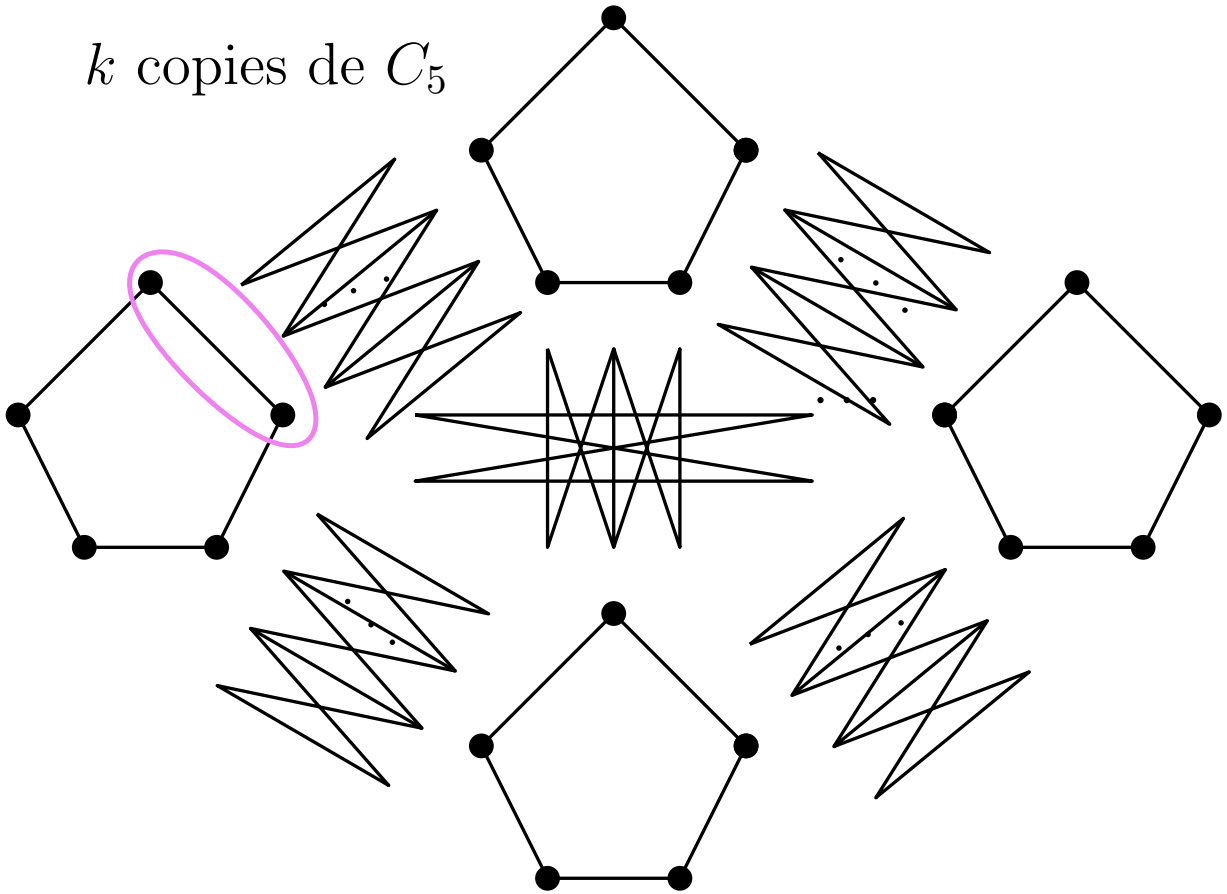
On peut même avoir un écart aussi grand que l'on veut !



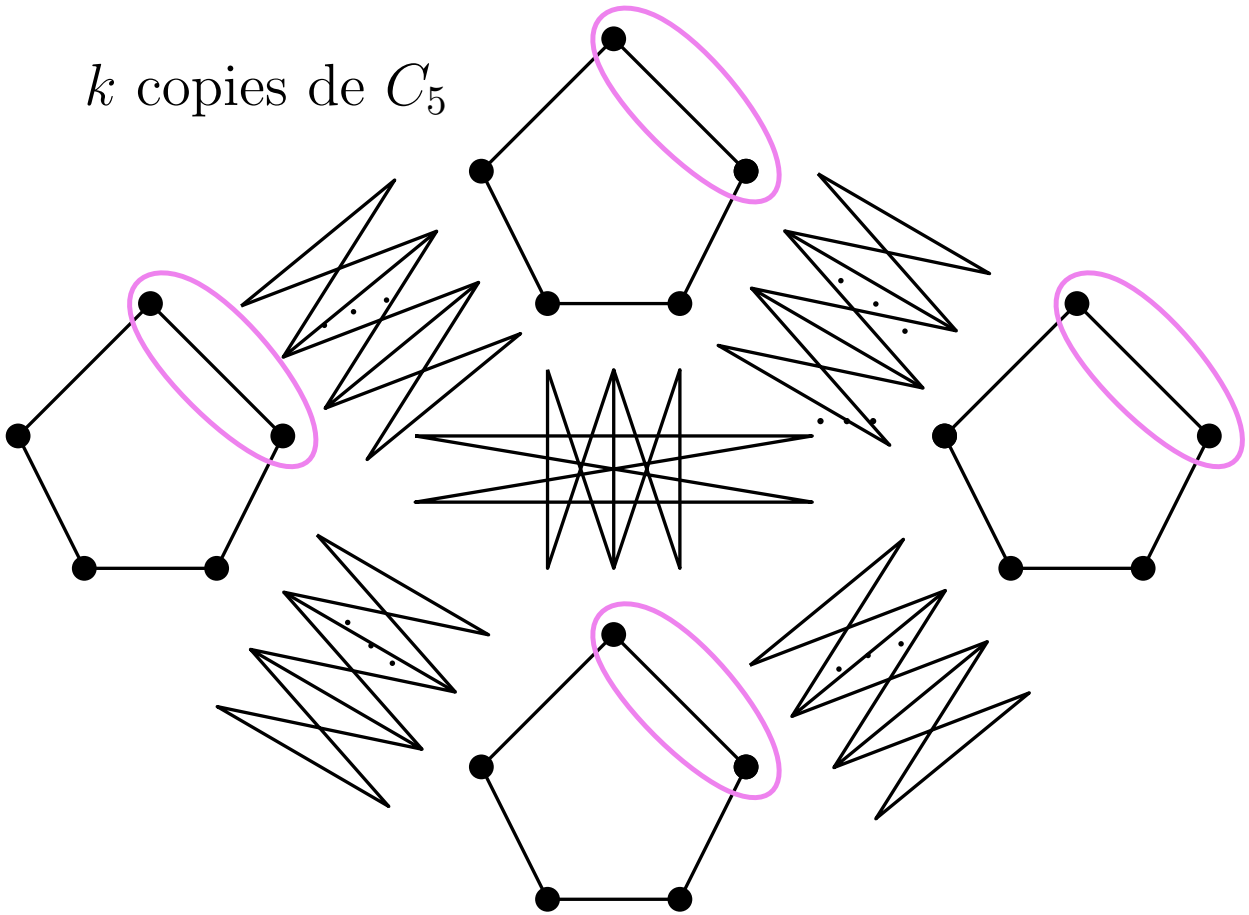
On peut même avoir un écart aussi grand que l'on veut !



On peut même avoir un écart aussi grand que l'on veut !

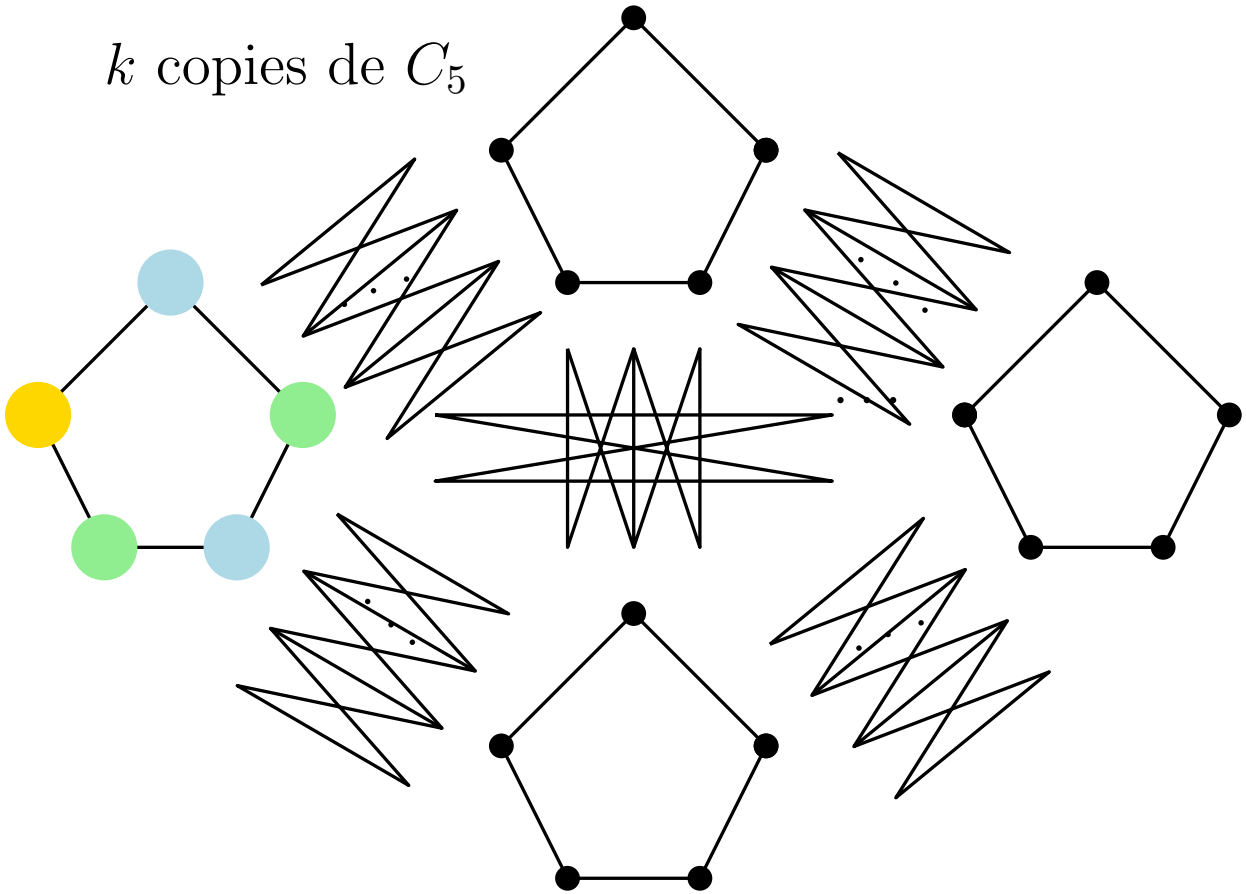


On peut même avoir un écart aussi grand que l'on veut !



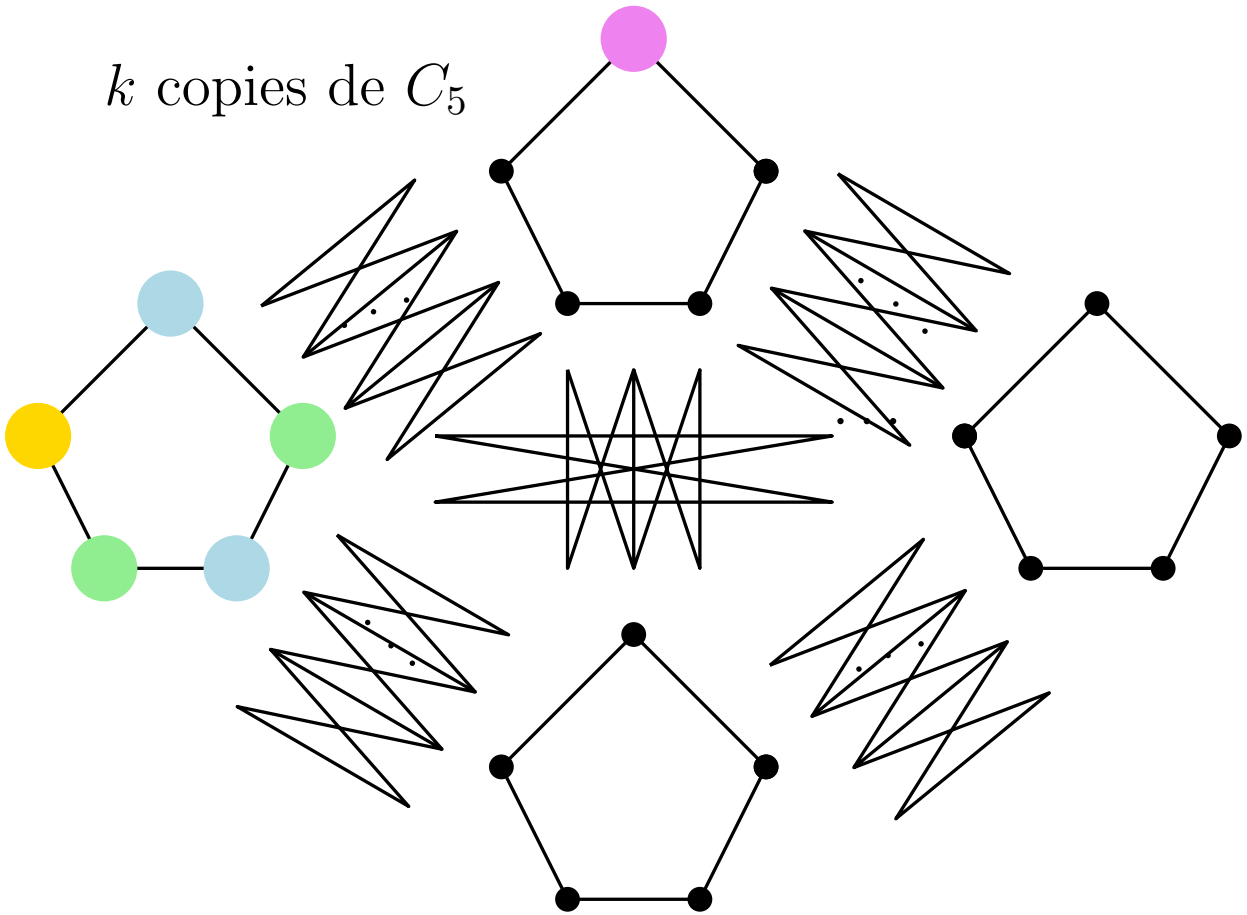
Taille max de clique : $\omega(G) = 2k$

On peut même avoir un écart aussi grand que l'on veut !



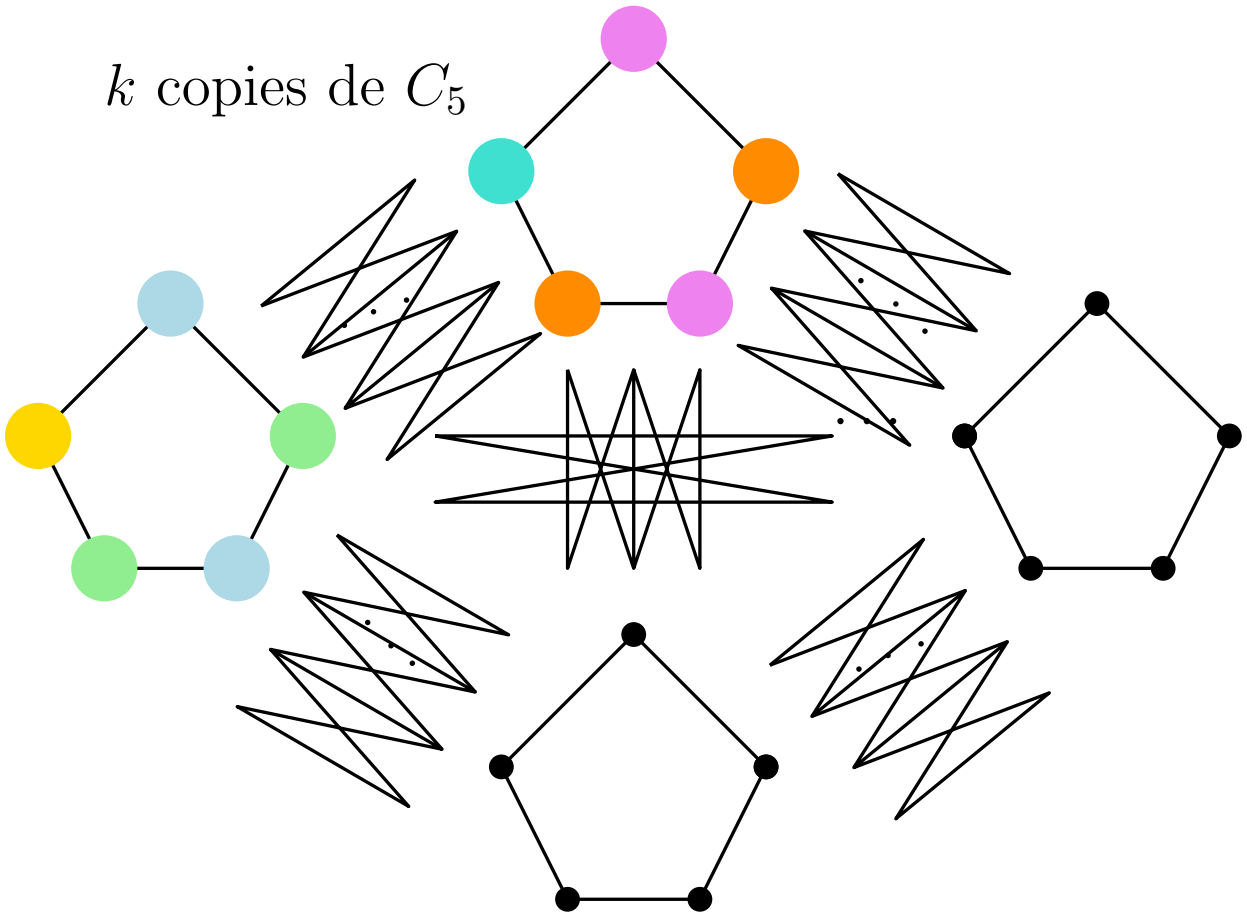
Taille max de clique : $\omega(G) = 2k$

On peut même avoir un écart aussi grand que l'on veut !



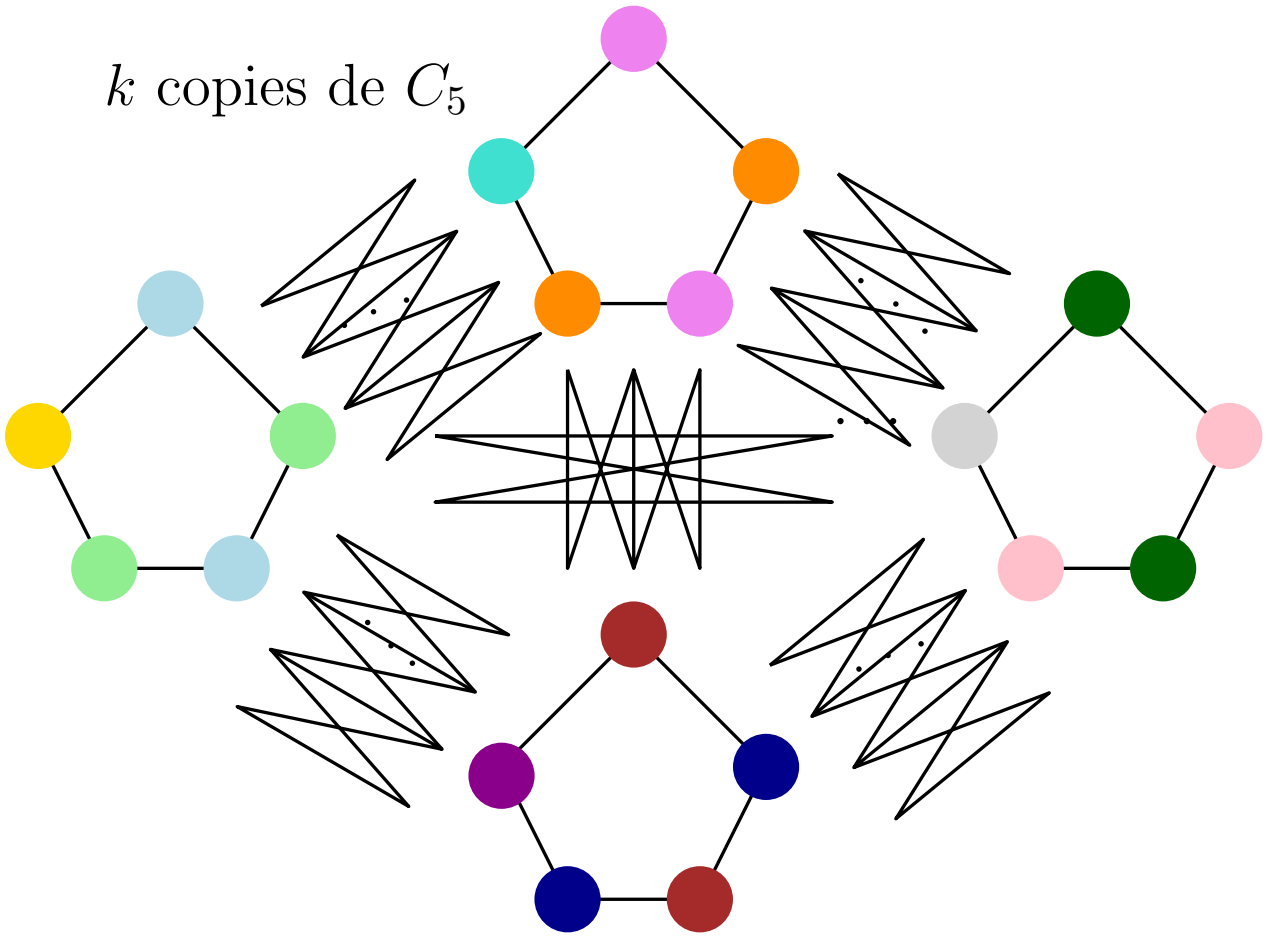
Taille max de clique : $\omega(G) = 2k$

On peut même avoir un écart aussi grand que l'on veut !



Taille max de clique : $\omega(G) = 2k$

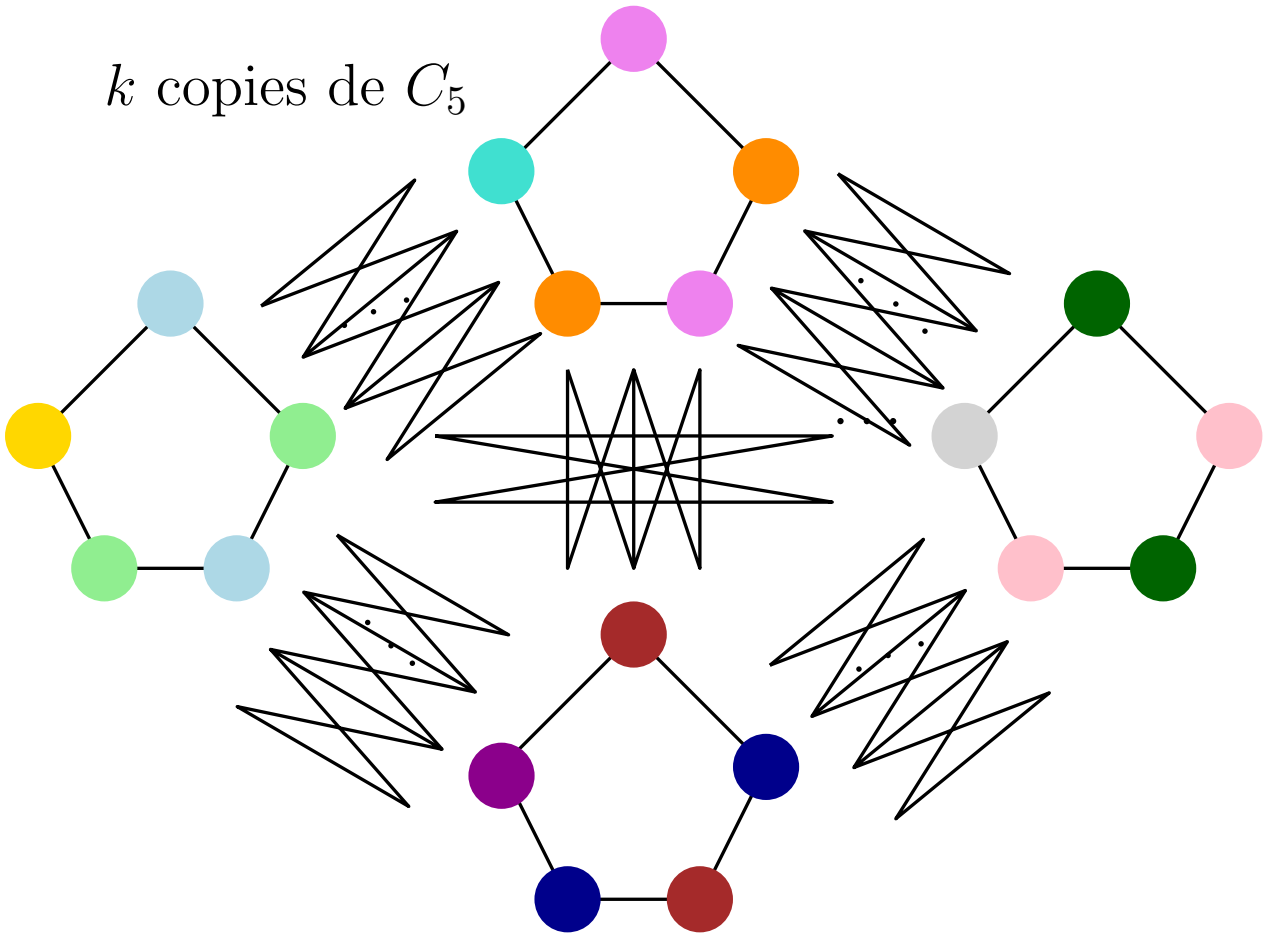
On peut même avoir un écart aussi grand que l'on veut !



Taille max de clique : $\omega(G) = 2k$

Nb de couleurs : $\chi(G) = 3k$

On peut même avoir un écart aussi grand que l'on veut !



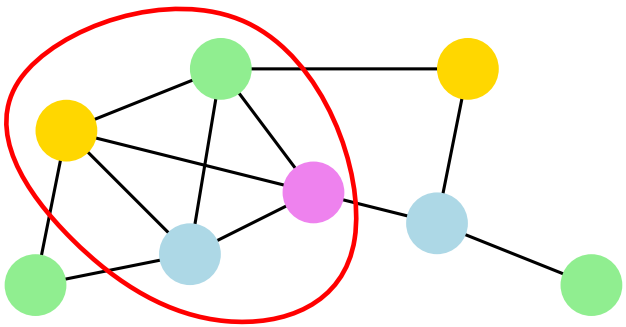
Taille max de clique : $\omega(G) = 2k$

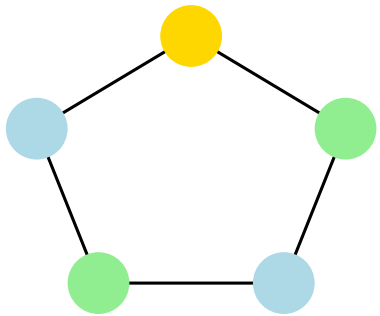
Nb de couleurs : $\chi(G) = 3k$

$\chi(G) - \omega(G) = k$

Graphes parfaits

Certifier que beaucoup de couleurs sont nécessaires ?



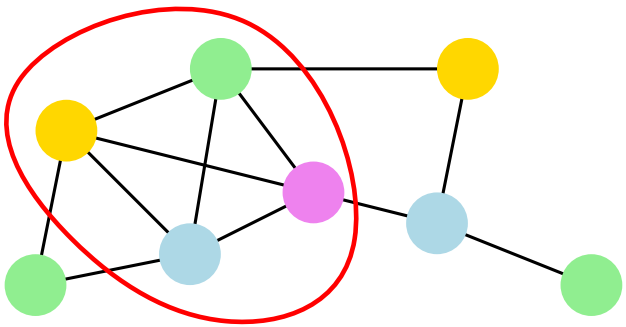


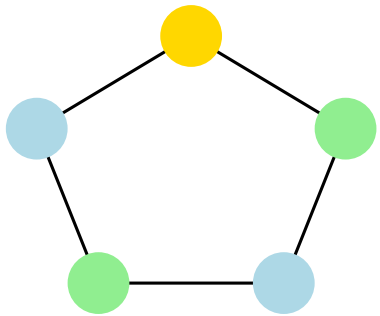
nb. couleurs $\geq$ max. clique nb. couleurs $>$ max. clique

Si nb. couleurs $=$ max. clique $\Rightarrow$ le graphe est *parfait*.
(partout dans le graphe)

Graphes parfaits

Certifier que beaucoup de couleurs sont nécessaires ?

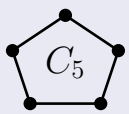




nb. couleurs $\geq$ max. clique nb. couleurs $>$ max. clique

Si nb. couleurs = max. clique $\Rightarrow$ le graphe est *parfait*.
(partout dans le graphe)

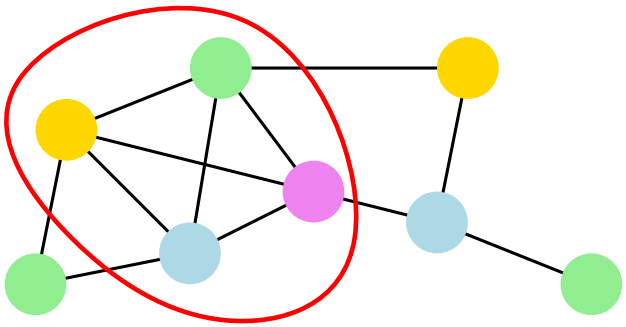
Question :

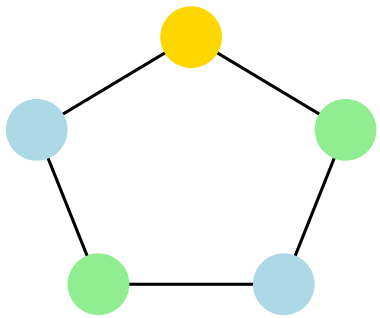
Si G ne contient pas 

, alors G est parfait ?

Graphes parfaits

Certifier que beaucoup de couleurs sont nécessaires ?

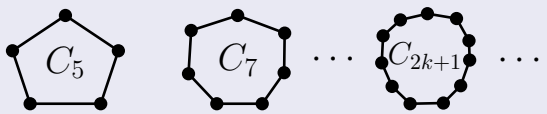




nb. couleurs $\geq$ max. clique nb. couleurs $>$ max. clique

Si nb. couleurs = max. clique $\Rightarrow$ le graphe est *parfait*.
(partout dans le graphe)

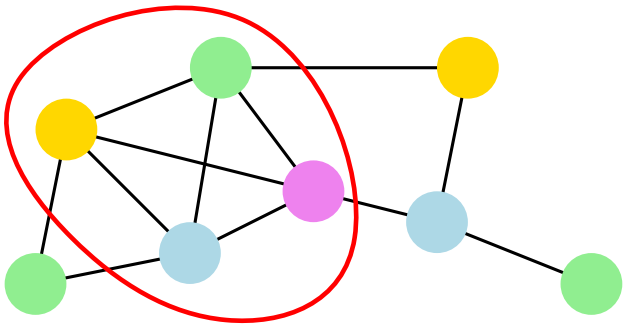
Question :

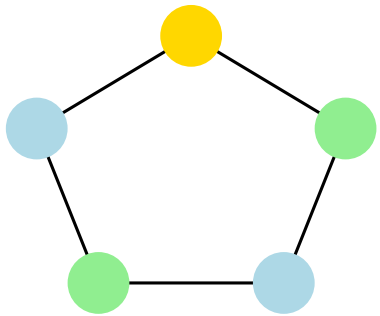
Si G ne contient pas 

, alors G est parfait ?

Graphes parfaits

Certifier que beaucoup de couleurs sont nécessaires ?

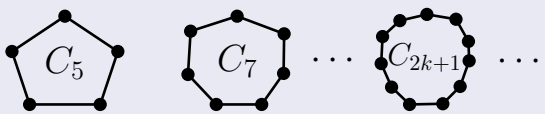




nb. couleurs $\geq$ max. clique nb. couleurs $>$ max. clique

Si nb. couleurs $=$ max. clique $\Rightarrow$ le graphe est *parfait*.
(partout dans le graphe)

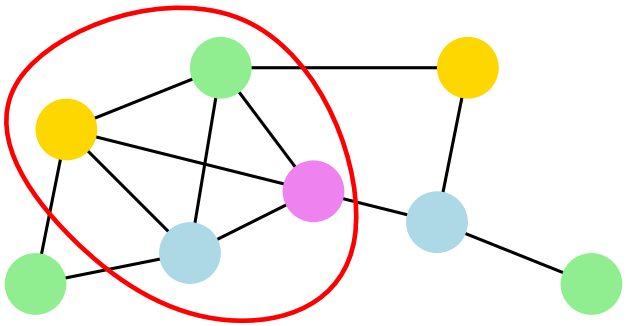
Conjecture forte des graphes parfaits (1960)

Si ni G ni $\overline{G}$ ne contient 

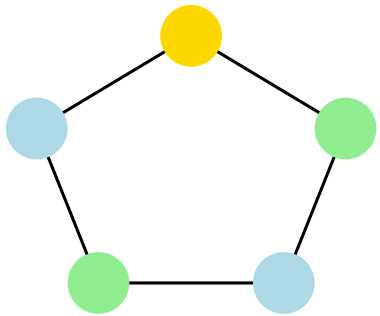
, alors G est parfait.

Graphes parfaits

Certifier que beaucoup de couleurs sont nécessaires ?



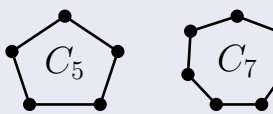
nb. couleurs $\geq$ max. clique



nb. couleurs $>$ max. clique

Si nb. couleurs = max. clique $\Rightarrow$ le graphe est *parfait*.
(partout dans le graphe)

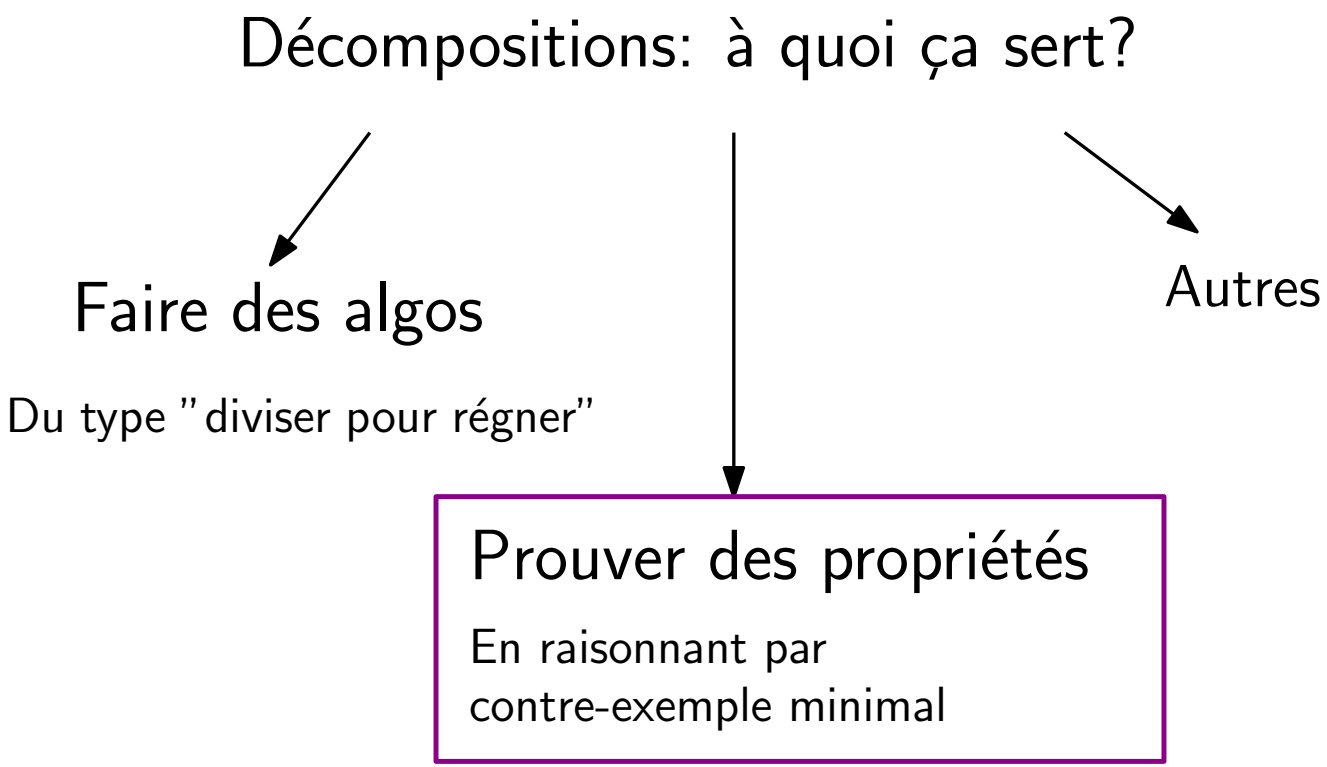
~~Conjecture~~Théorème fort des graphes parfaits (1960$\rightarrow$ 2002)

Si ni G ni $\overline{G}$ ne contient 

, alors G est parfait.

[Chudnovsky, Robertson, Seymour, Thomas]

Méthodes de décompositions



Méthodes de décompositions

Exemple 1 - But : prouver que tout graphe sans trou ni anti-trou impair est parfait.

Combinaison de 3 ingrédients :

1. Choisir un problème : algo à écrire, propriété à montrer, ...
   Exemple : propriété d'être un graphe parfait
2. Identifier les décompositions qui se comportent "bien" par rapport à ce problème-là
   Exemple : clique déconnectante, (2-join, BSP, ...)
3. Se restreindre à une classe structurée dans laquelle on peut (presque) toujours trouver une décomposition listée à l'item 2.
   Exemple : graphes qui n'ont pas de trous ni d'anti-trous impairs

1^e étape : Prouver un théorème de décomposition.

Théorème de décomposition [CRST'02]

Soit G un graphe sans trou ni antitrou impair. Alors

- soit G est *facilement parfait* (biparti, ...),
- soit G admet une décomposition parmi la liste suivante : [...],
  $\rightarrow$ G peut être coupé *de manière spéciale* en plusieurs blocs.

$\rightarrow$ Partie la plus complexe de la preuve

1^e étape : Prouver un théorème de décomposition.

Théorème de décomposition [CRST'02]

Soit G un graphe sans trou ni antitrou impair. Alors

- soit G est *facilement parfait* (biparti, ...),
- soit G admet une décomposition parmi la liste suivante : [...],
  $\rightarrow$ G peut être coupé *de manière spéciale* en plusieurs blocs.

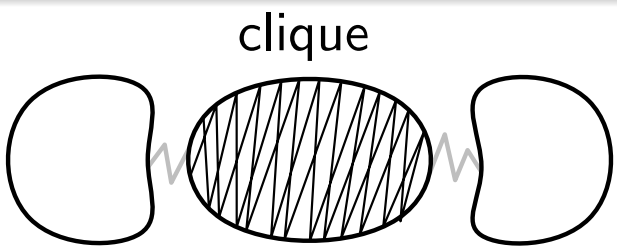
$\rightarrow$ Partie la plus complexe de la preuve

Pour simplifier, supposons :

Théorème fictif de décomposition

Soit G un graphe sans trou ni antitrou impair. Alors

- soit G est *facilement parfait* (biparti, ...),
- soit G admet une clique-déconnectante



Théorème fictif de décomposition

Soit G un graphe sans trou ni antitrou impair. Alors

- soit G est *facilement parfait*,
- soit G peut être coupé par une clique-déconnectante

Théorème fictif de décomposition

Soit G un graphe sans trou ni antitrou impair. Alors

- soit G est *facilement parfait*,
- soit G peut être coupé par une clique-déconnectante

But : tout graphe sans trou ni antitrou impair est parfait.

Théorème fictif de décomposition

Soit G un graphe sans trou ni antitrou impair. Alors

- soit G est *facilement parfait*,
- soit G peut être coupé par une clique-déconnectante

But : tout graphe sans trou ni antitrou impair est parfait.

Soit G un contre-exemple minimal.

Théorème fictif de décomposition

Soit G un graphe sans trou ni antitrou impair. Alors

- soit G est *facilement parfait*,
- soit G peut être coupé par une clique-déconnectante

But : tout graphe sans trou ni antitrou impair est parfait.

Soit G un contre-exemple minimal.
On applique le théorème de décomposition :

Théorème fictif de décomposition

Soit G un graphe sans trou ni antitrou impair. Alors

- soit G est *facilement parfait*,
- soit G peut être coupé par une clique-déconnectante

But : tout graphe sans trou ni antitrou impair est parfait.

Soit G un contre-exemple minimal.
On applique le théorème de décomposition :

- Soit G est *facilement parfait*

Théorème fictif de décomposition

Soit G un graphe sans trou ni antitrou impair. Alors

- soit G est *facilement parfait*,
- soit G peut être coupé par une clique-déconnectante

But : tout graphe sans trou ni antitrou impair est parfait.

Soit G un contre-exemple minimal.
On applique le théorème de décomposition :

- Soit G est *facilement parfait* $\rightarrow$ Une contradiction.

Théorème fictif de décomposition

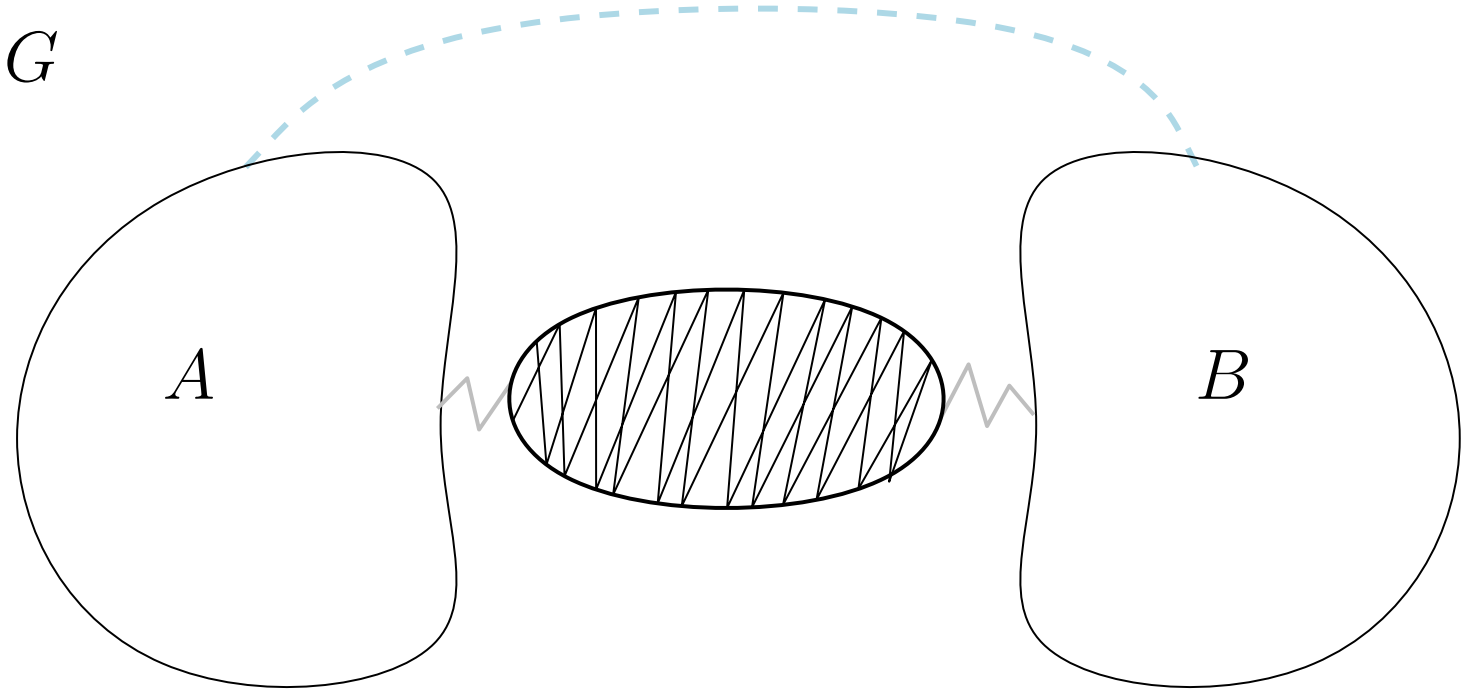
Soit G un graphe sans trou ni antitrou impair. Alors

- soit G est *facilement parfait*,
- soit G peut être coupé par une clique-déconnectante

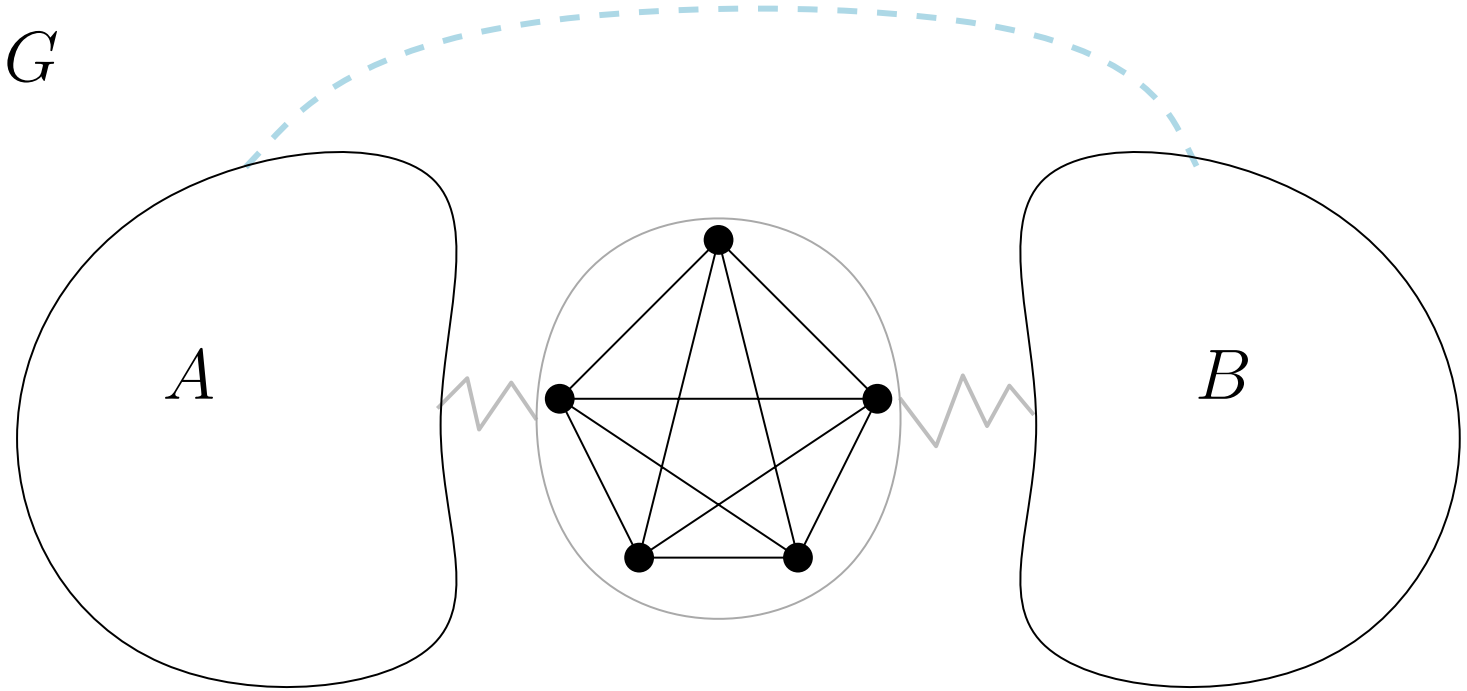
But : tout graphe sans trou ni antitrou impair est parfait.

Soit G un contre-exemple minimal.
On applique le théorème de décomposition :

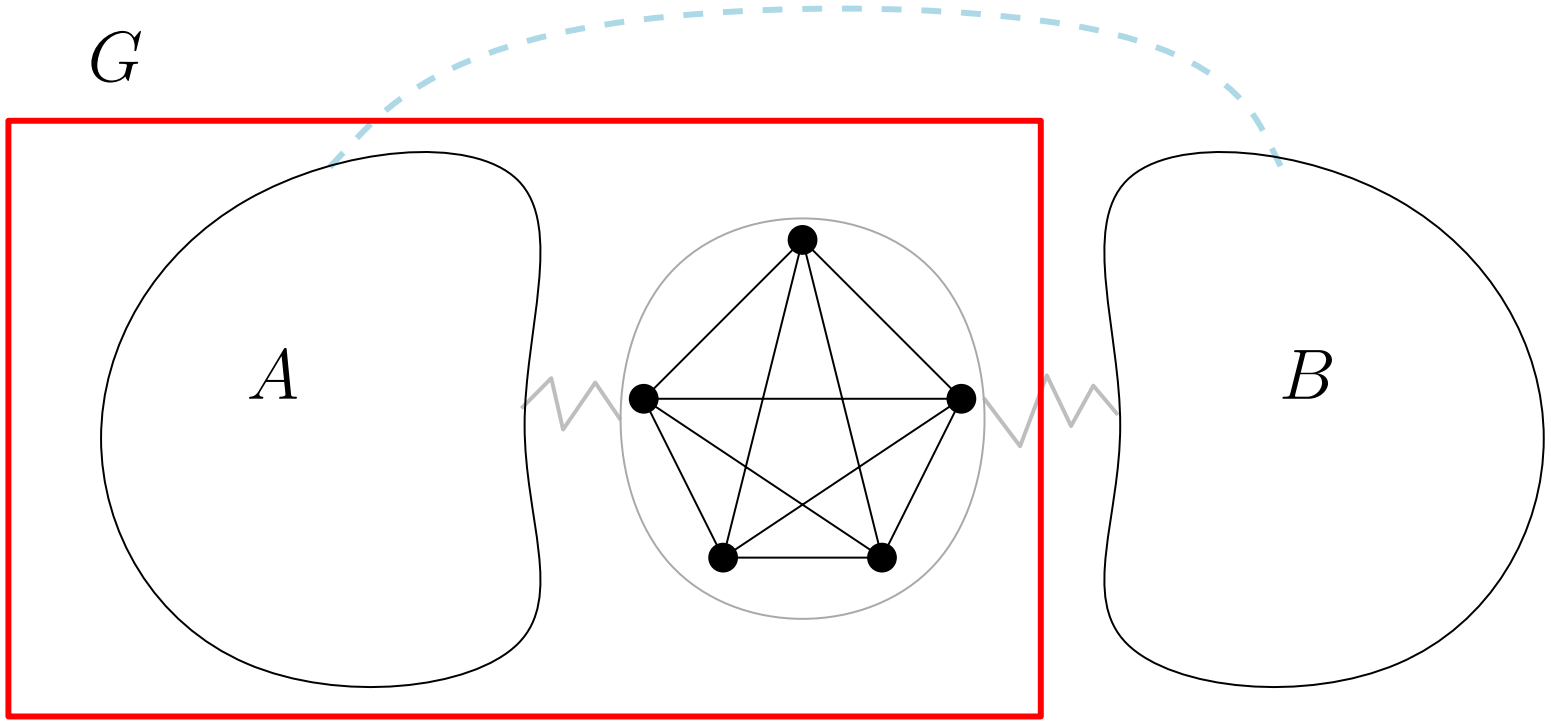
- Soit G est *facilement parfait* $\rightarrow$ Une contradiction.
- Soit G admet une clique-déconnectante. $\rightarrow$ Prouvons qu'il est parfait



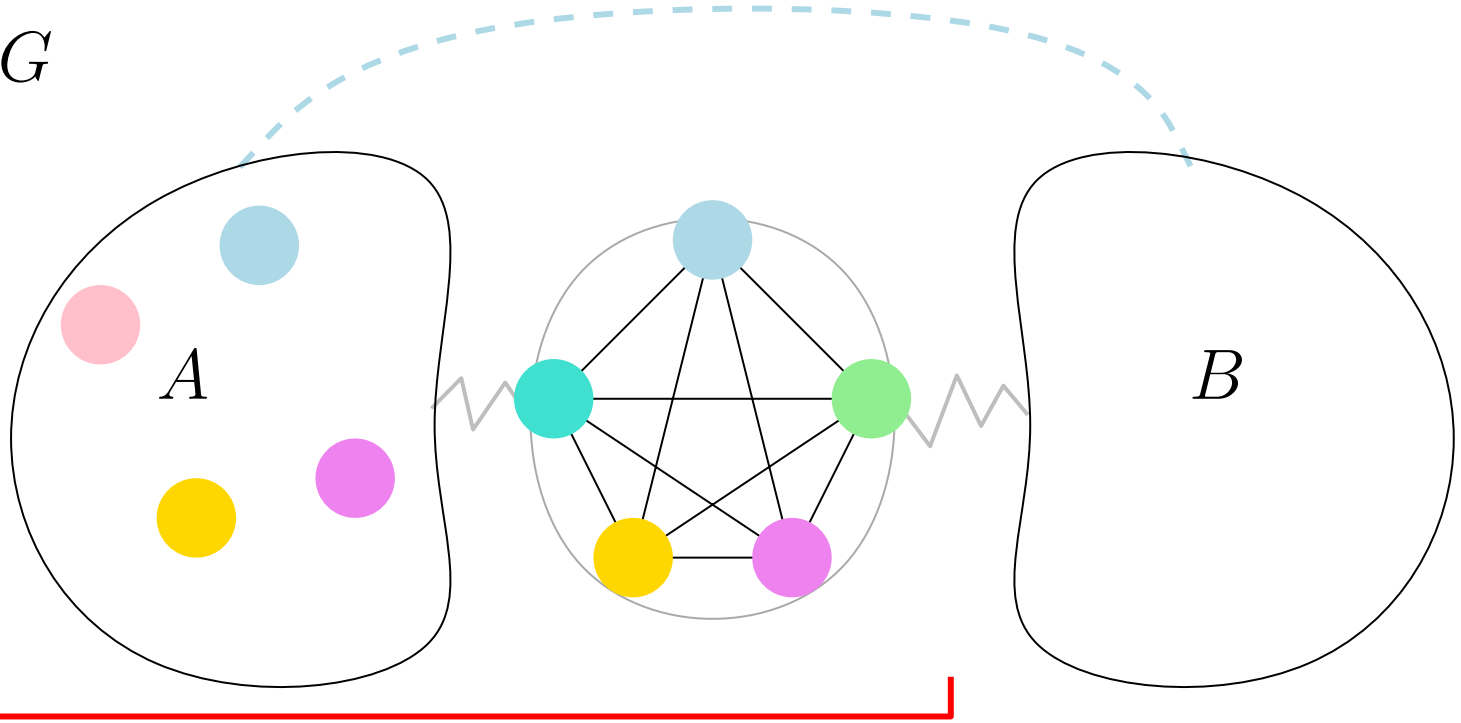
Clique K



Clique K



$G_A := G[A \cup K]$ parfait Clique K



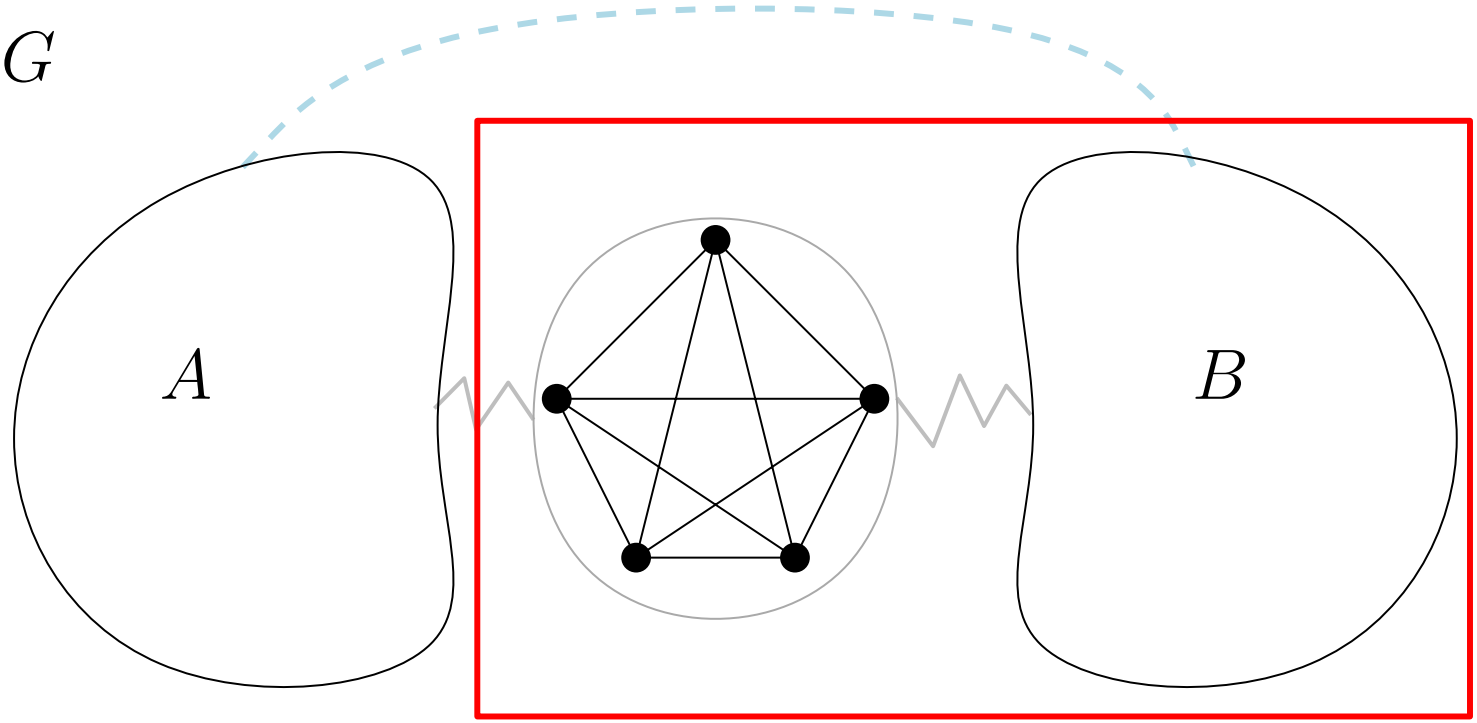
Clique K

G_A

$\chi(A \cup K) = \omega(A \cup K)$

χ : nb couleurs

ω : taille max. clique

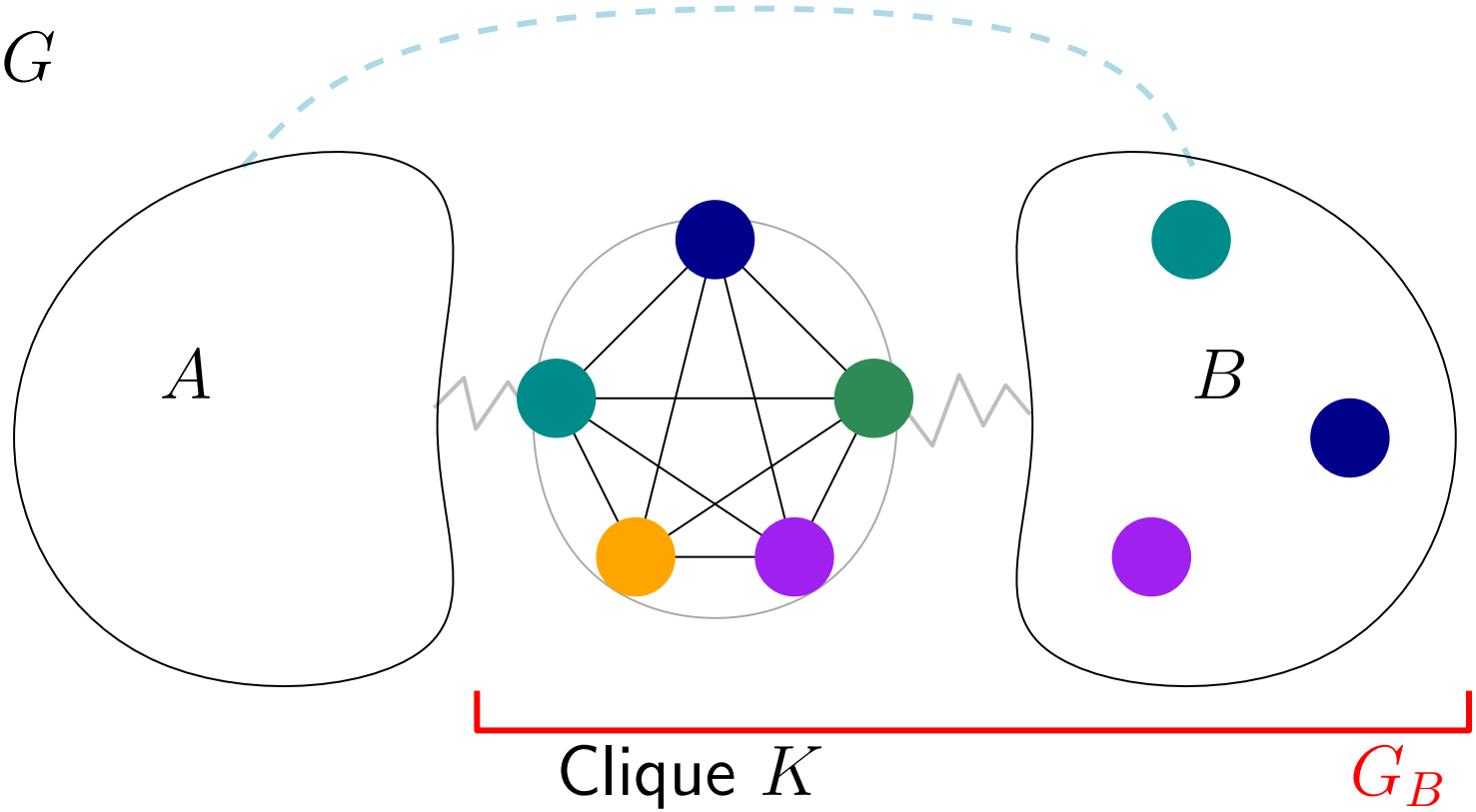


Clique K

$G_B := G[B \cup K]$ parfai

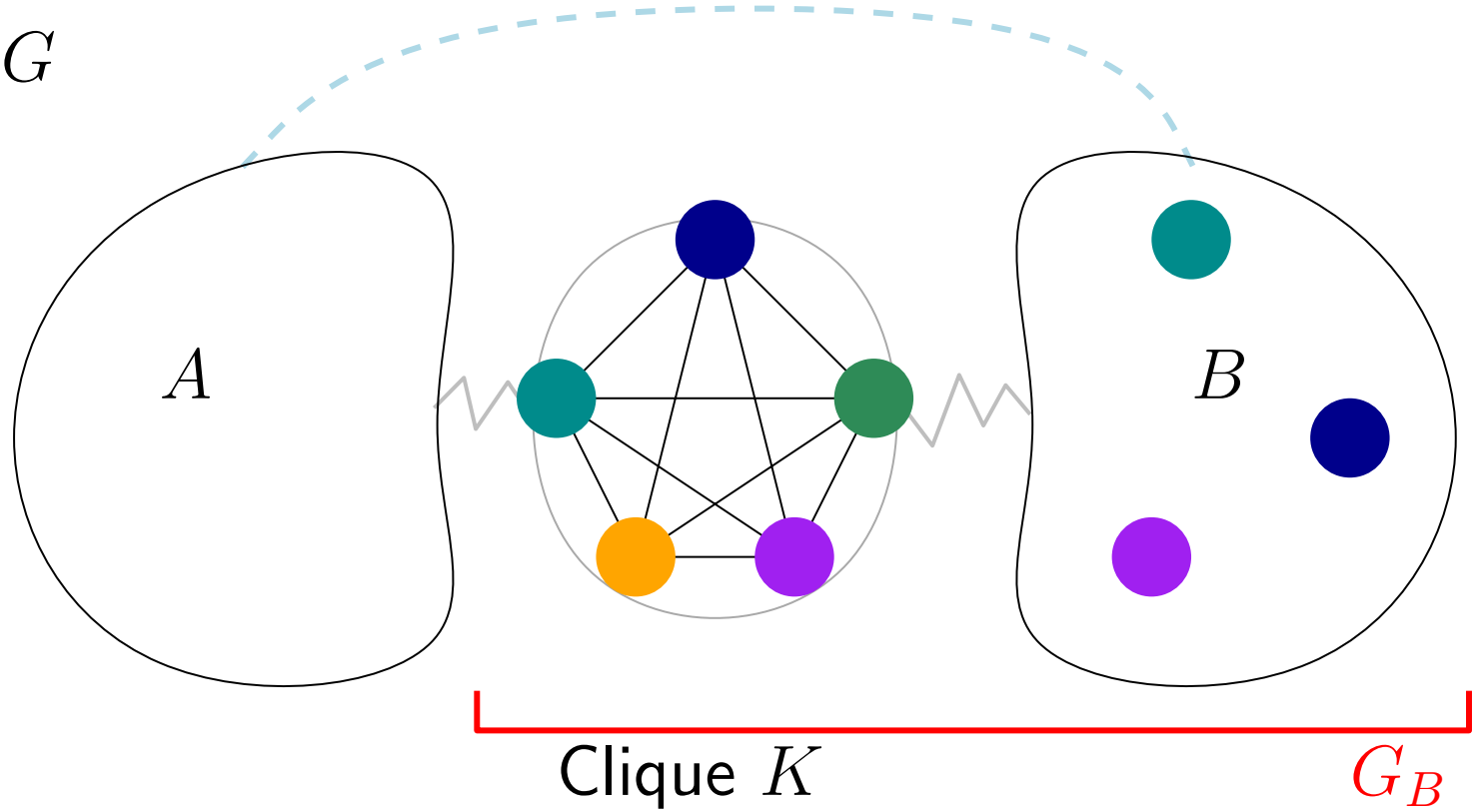
$\chi(A \cup K) = \omega(A \cup K)$

χ : nb couleurs
ω : taille max. clique



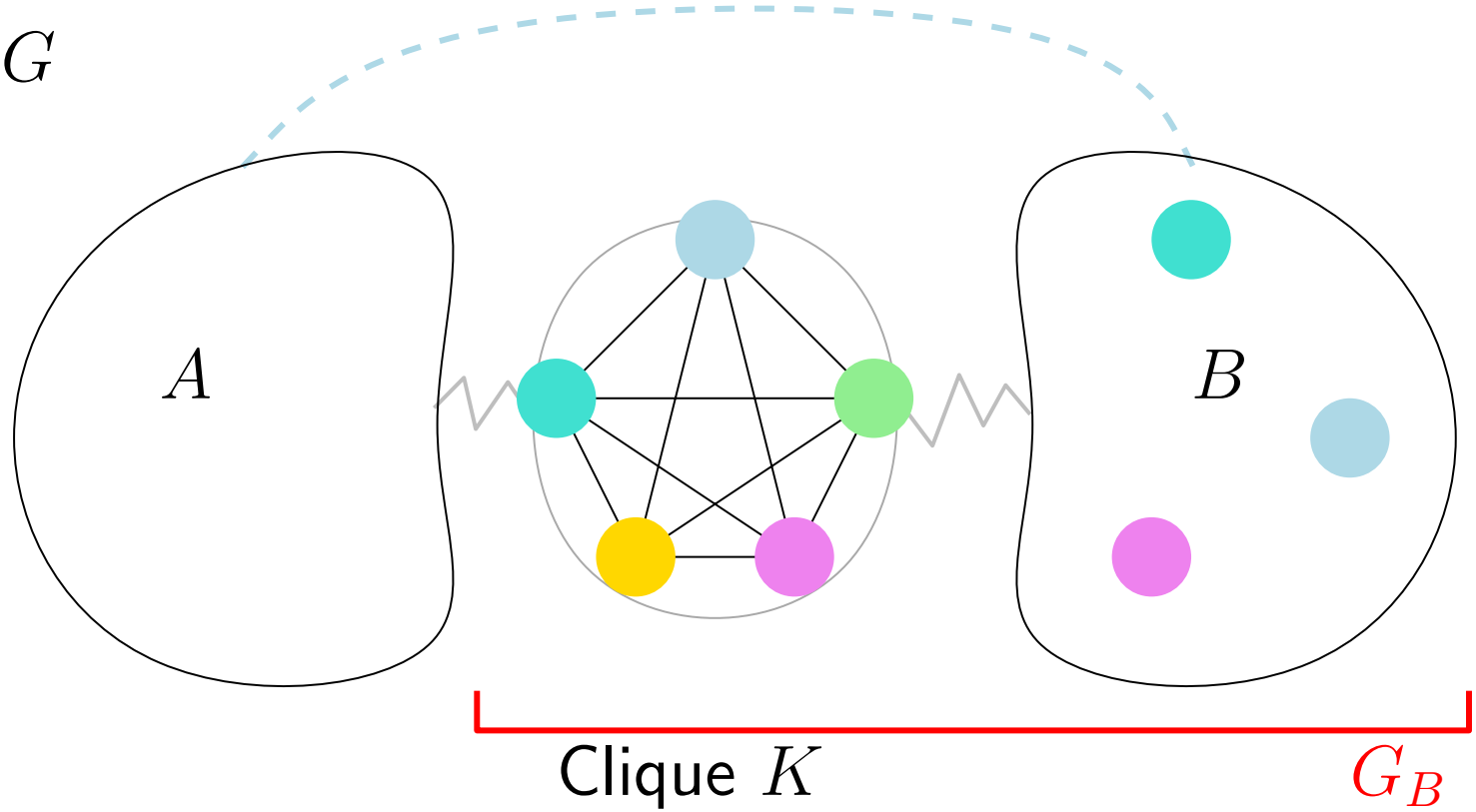
$\chi(B \cup K) = \omega(B \cup K)$

χ : nb couleurs
ω : taille max. clique



$\chi(B \cup K) = \omega(B \cup K)$

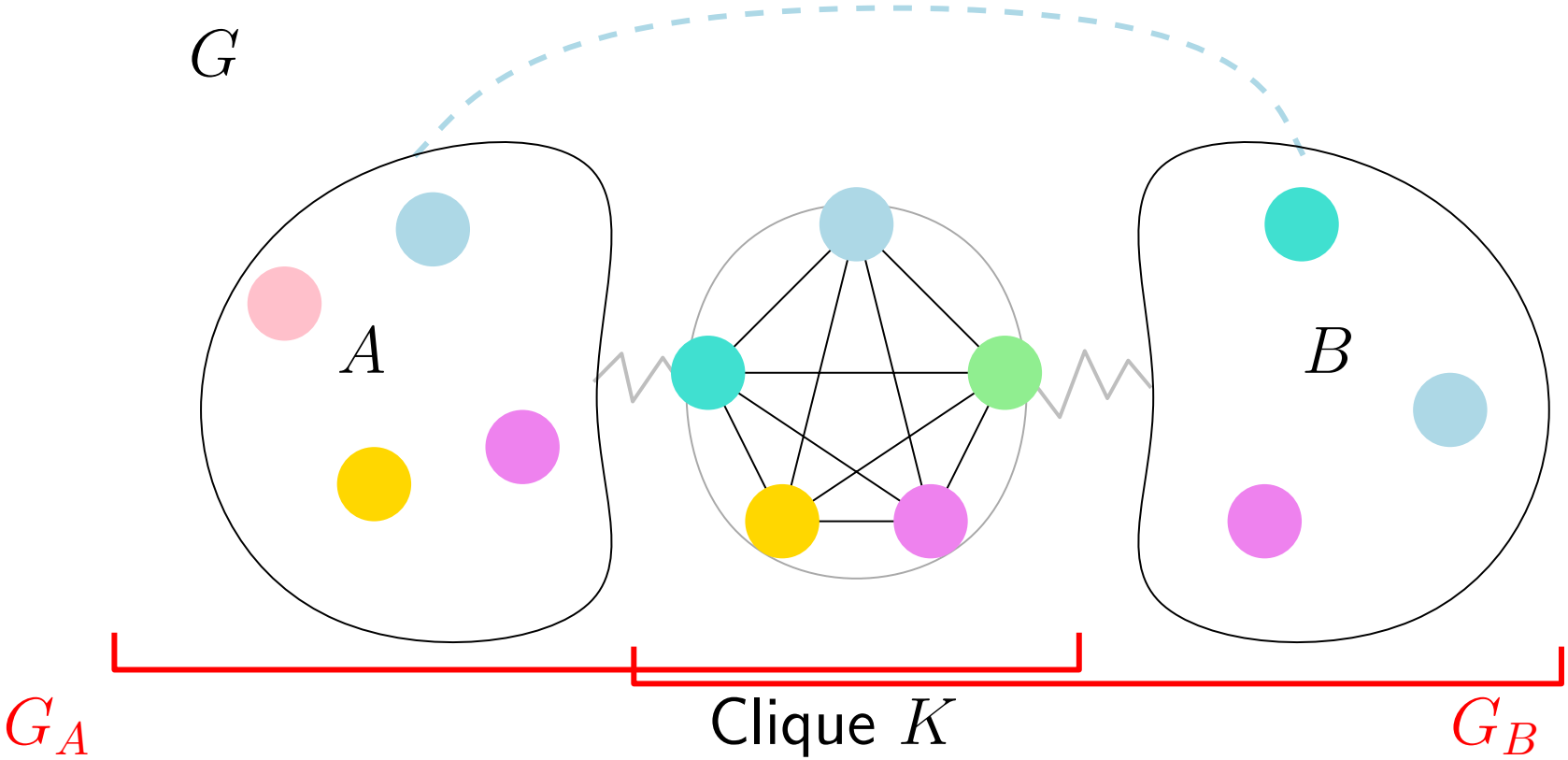
χ : nb couleurs
ω : taille max. clique



$\chi(B \cup K) = \omega(B \cup K)$

χ : nb couleurs

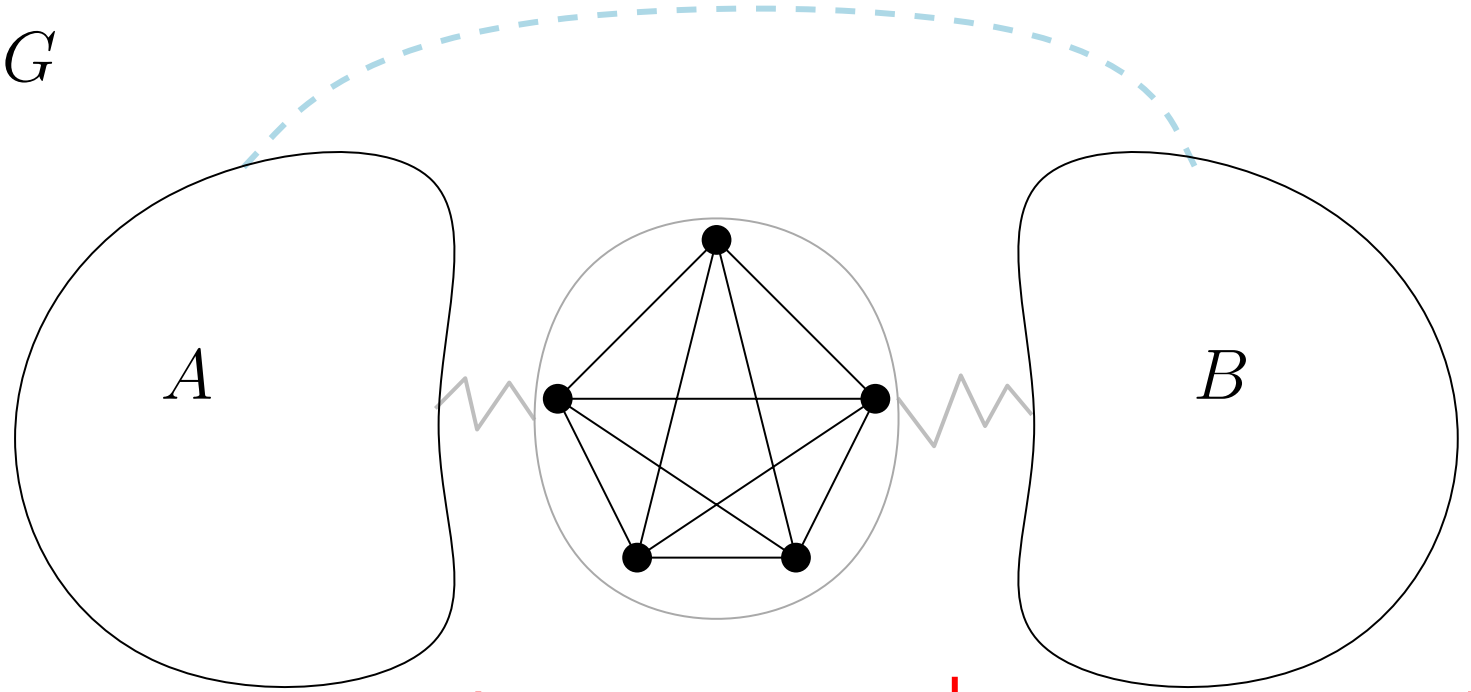
ω : taille max. clique



$\chi(B \cup K) = \omega(B \cup K)$

χ : nb couleurs

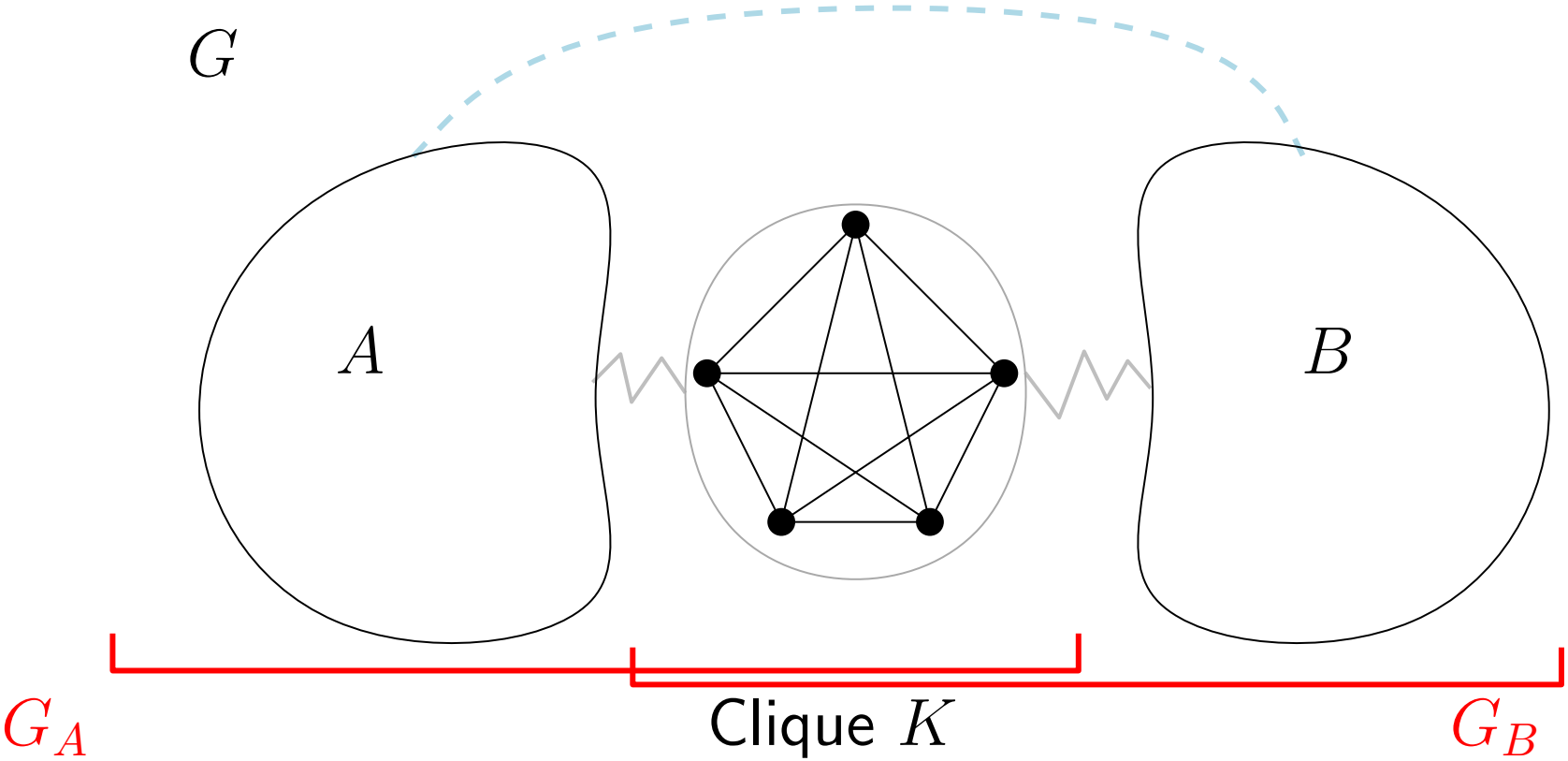
ω : taille max. clique



$\chi(G) \leq \max(\omega(A \cup K), \omega(B \cup K))$

χ : nb couleurs

ω : taille max. clique

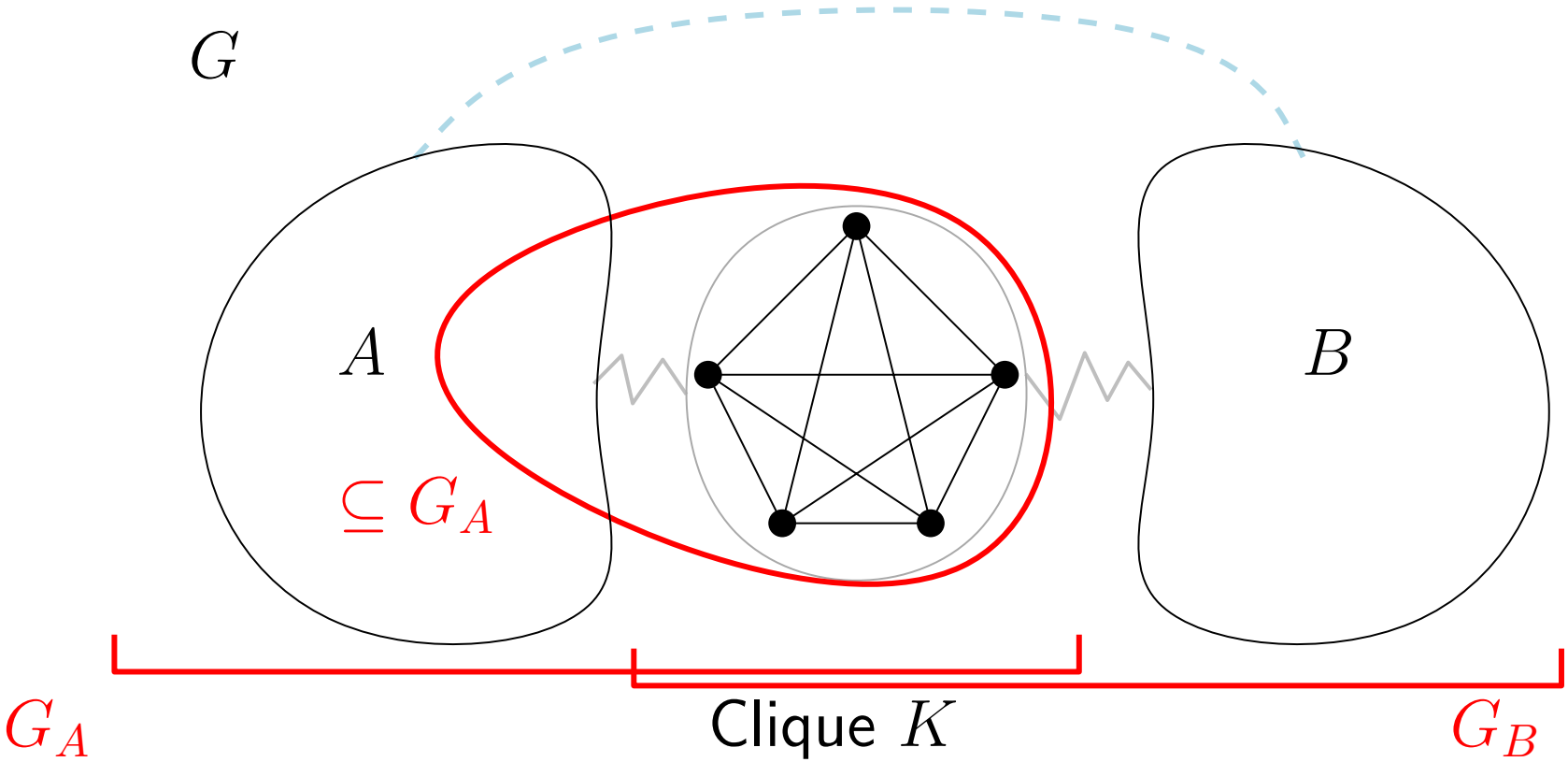


$\chi(G) \leq \max(\omega(A \cup K), \omega(B \cup K))$

$\omega(G) = \max(\omega(A \cup K), \omega(B \cup K))$?

χ : nb couleurs

ω : taille max. clique



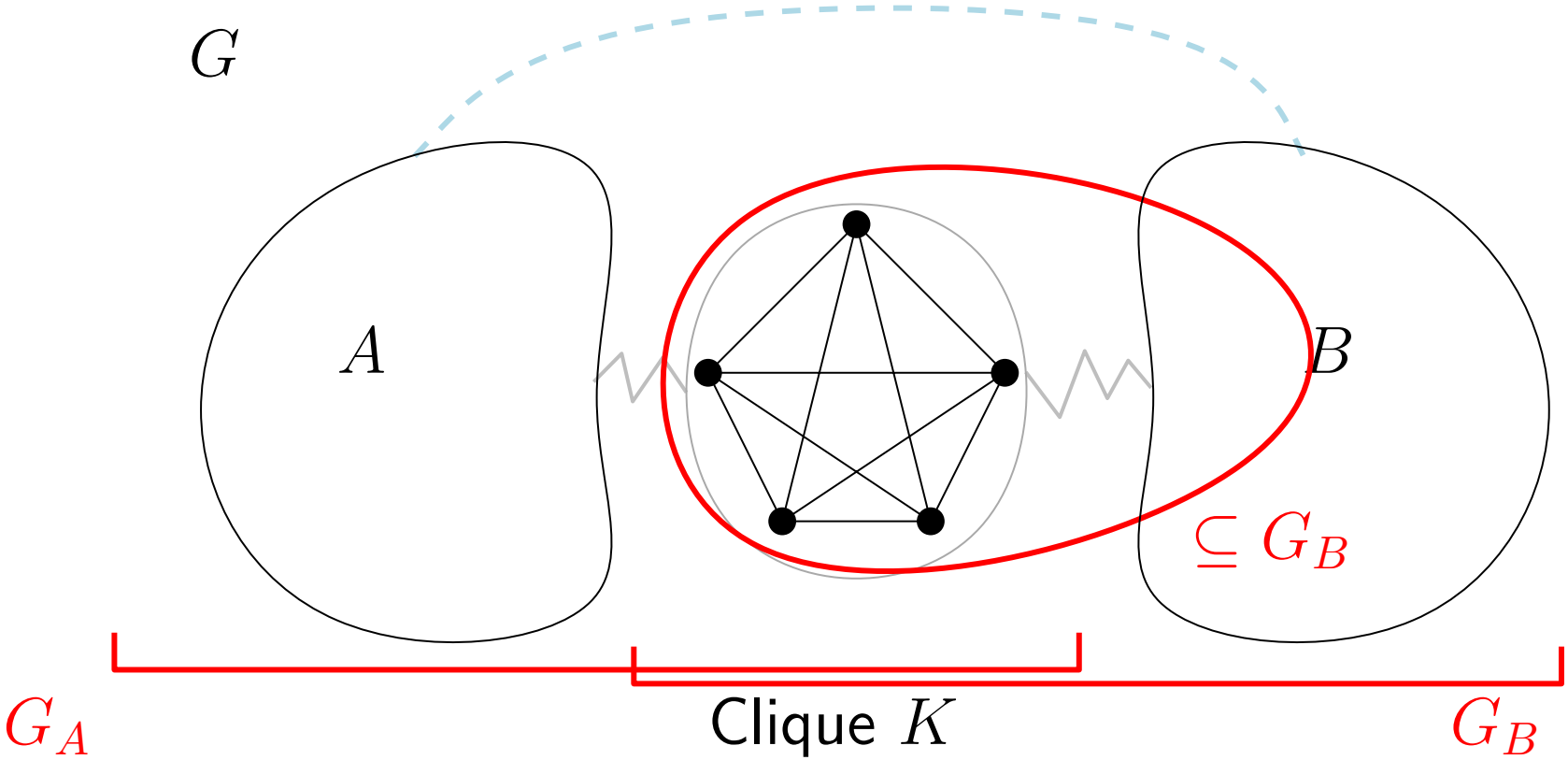
$\chi(G) \leq \max(\omega(A \cup K), \omega(B \cup K))$

$\omega(G) = \max(\omega(A \cup K), \omega(B \cup K))$?

Taille de la clique : $\leq \omega(A \cup K)$

χ : nb couleurs

ω : taille max. clique



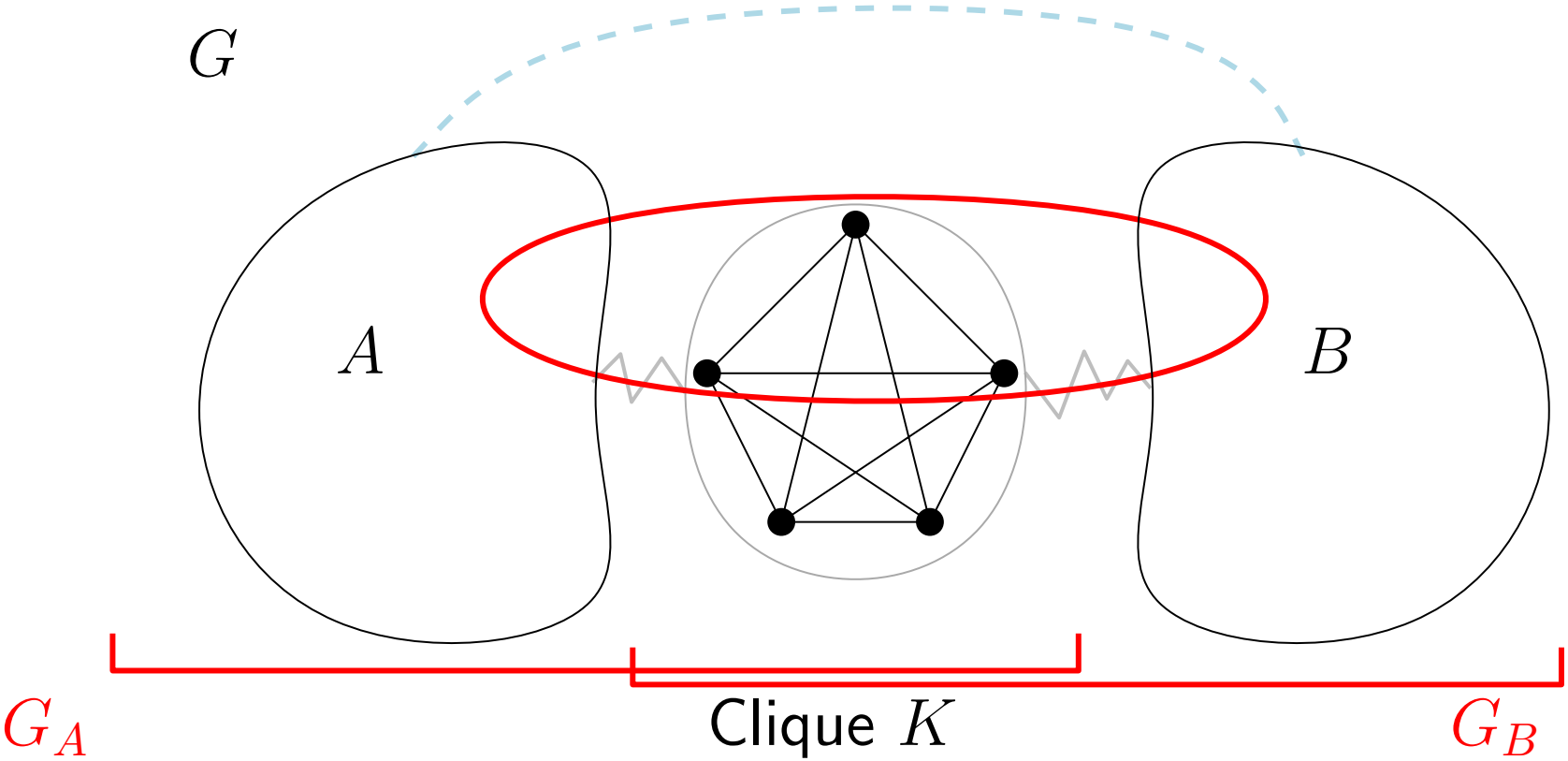
$\chi(G) \leq \max(\omega(A \cup K), \omega(B \cup K))$

$\omega(G) = \max(\omega(A \cup K), \omega(B \cup K))$?

Taille de la clique : $\leq \omega(B \cup K)$

χ : nb couleurs

ω : taille max. clique



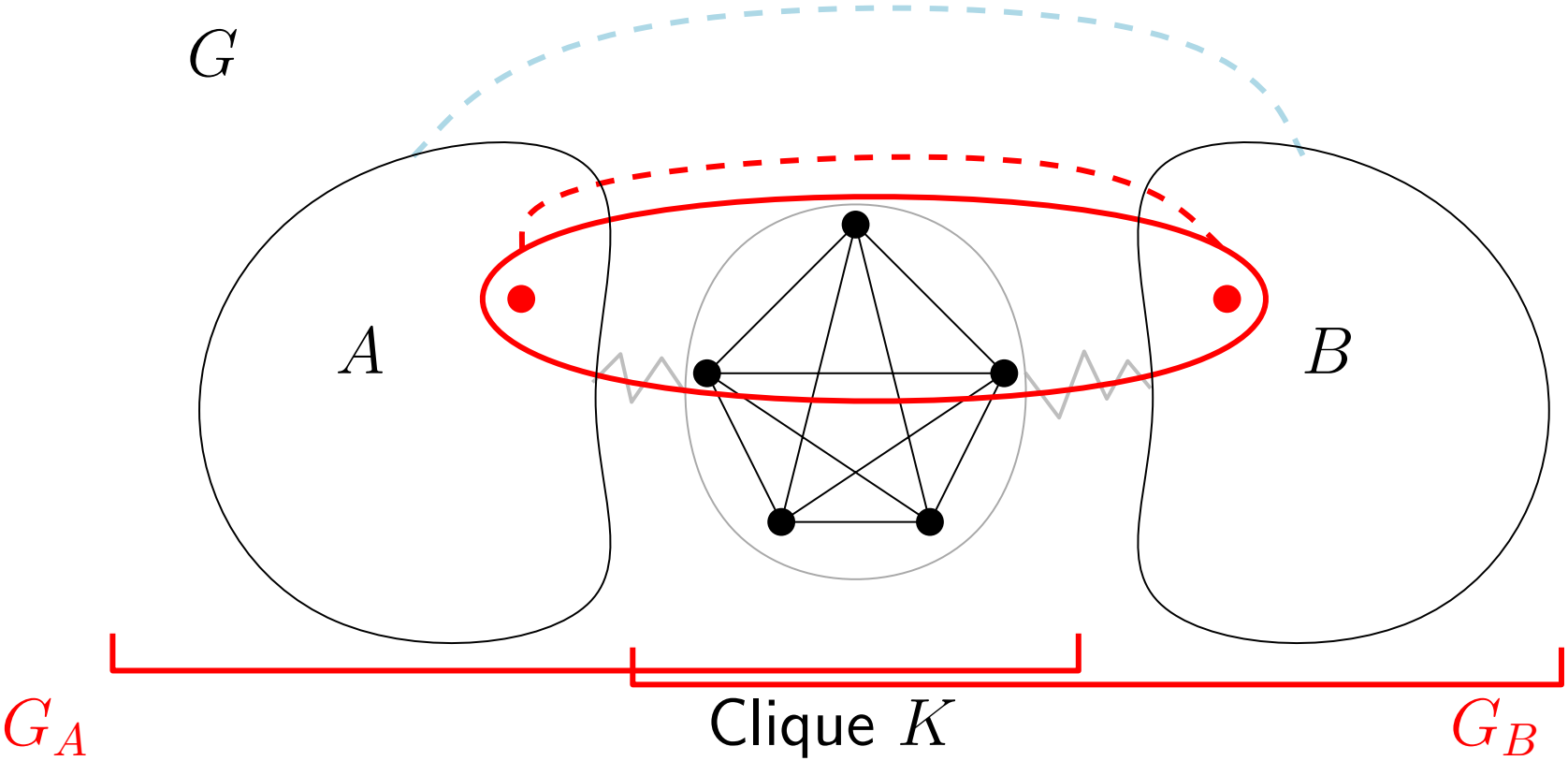
$\chi(G) \leq \max(\omega(A \cup K), \omega(B \cup K))$

$\omega(G) = \max(\omega(A \cup K), \omega(B \cup K))$?

Taille de la clique : ?

χ : nb couleurs

ω : taille max. clique



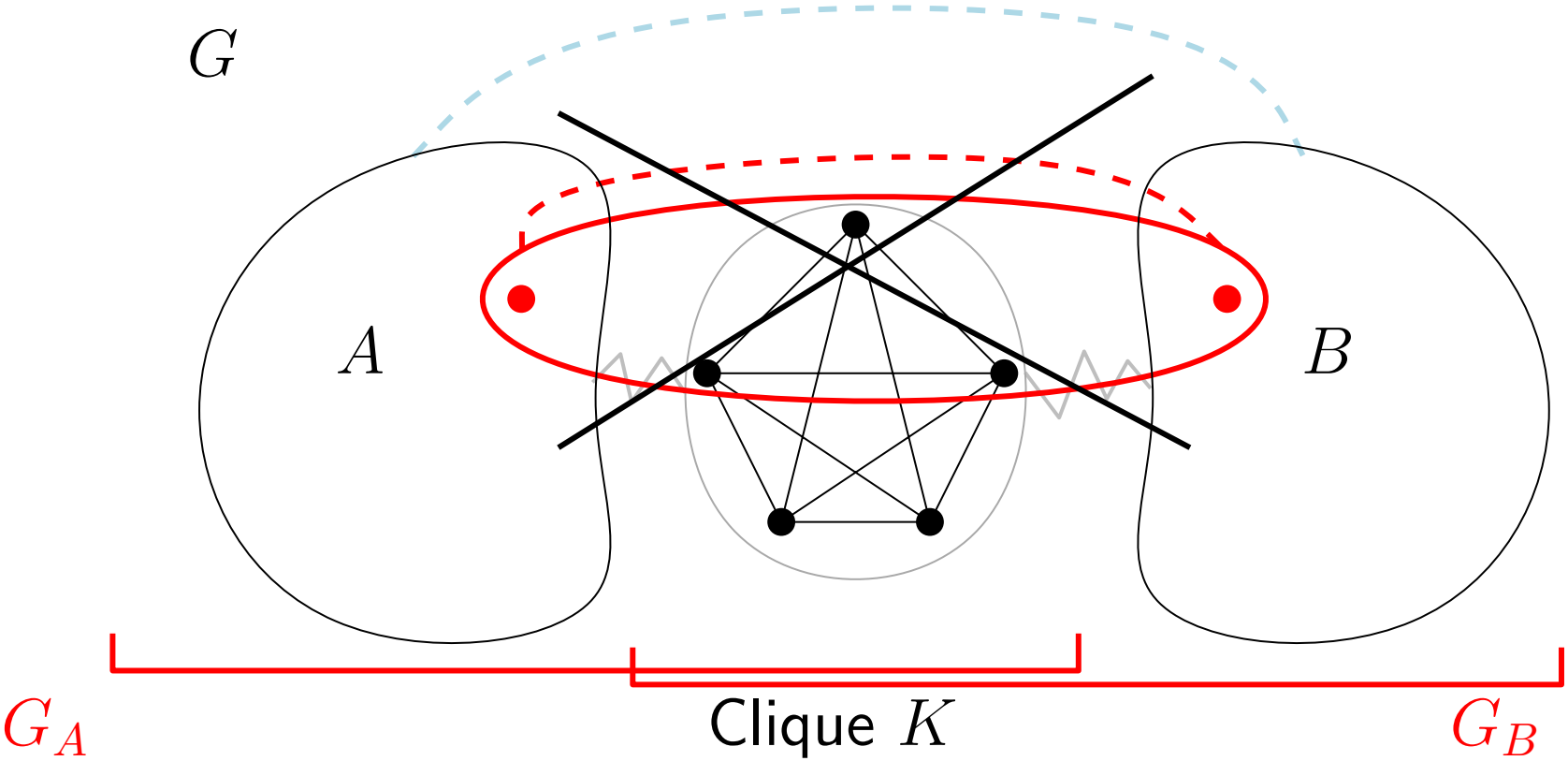
$\chi(G) \leq \max(\omega(A \cup K), \omega(B \cup K))$

$\omega(G) = \max(\omega(A \cup K), \omega(B \cup K))$?

Taille de la clique : ?

χ : nb couleurs

ω : taille max. clique



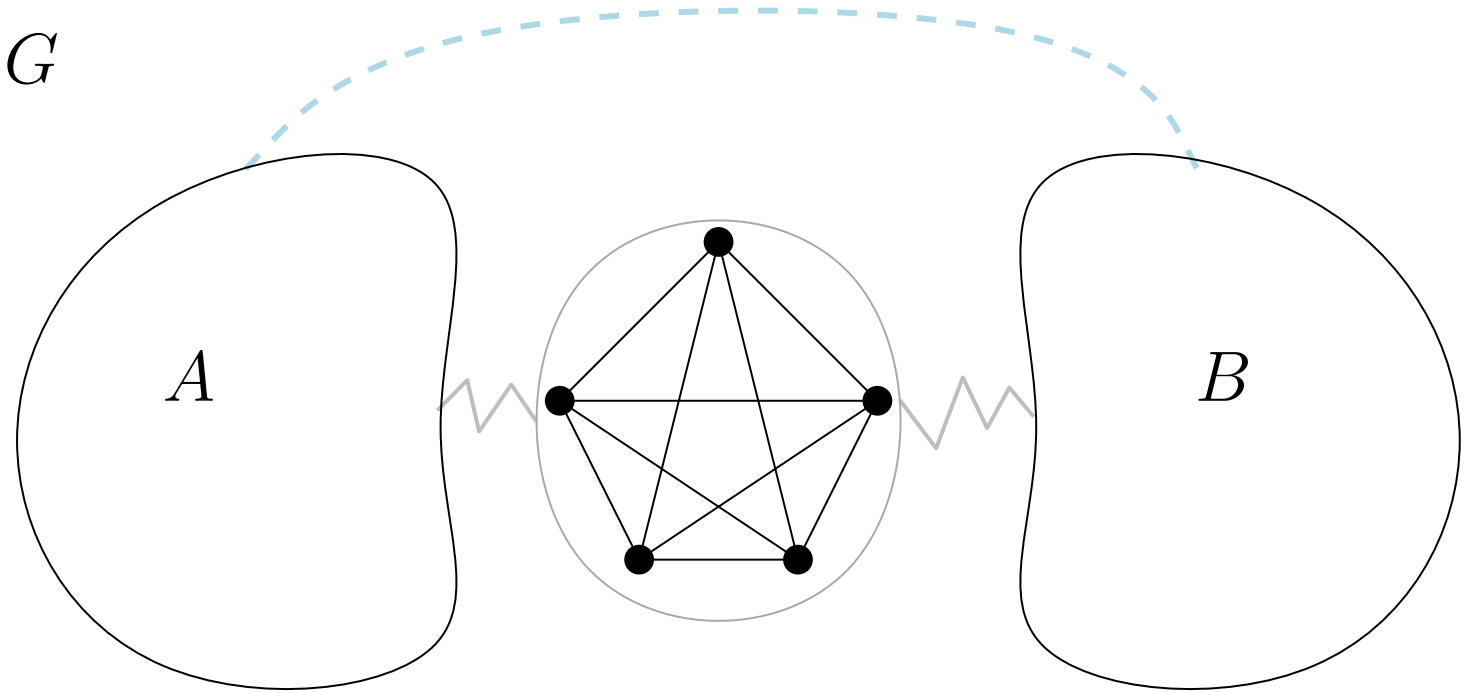
$\chi(G) \leq \max(\omega(A \cup K), \omega(B \cup K))$

$\omega(G) = \max(\omega(A \cup K), \omega(B \cup K))$?

Taille de la clique : ?

χ : nb couleurs

ω : taille max. clique

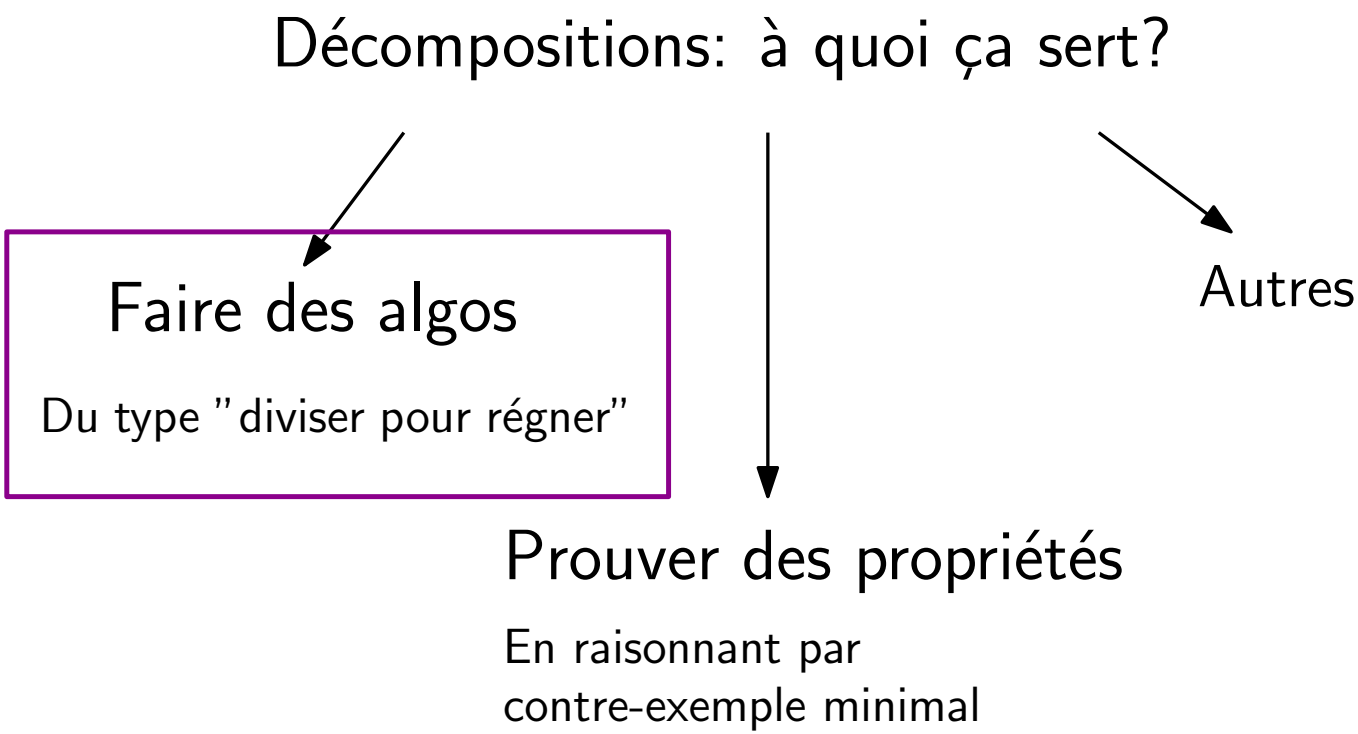


$\chi(G) \leq \max(\omega(A \cup K), \omega(B \cup K))$
$\omega(G) = \max(\omega(A \cup K), \omega(B \cup K))$?

χ : nb couleurs
ω : taille max. clique

$\chi(G) = \omega(G)$ donc G est parfait $\rightarrow$ CQFD
$\rightarrow$ Théorème fort des graphes parfaits.

Méthodes de décompositions



Méthodes de décompositions

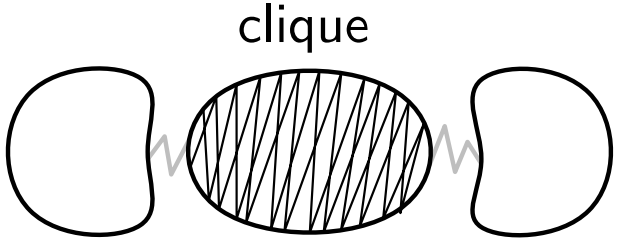
Exemple 2 - But : écrire un algorithme qui colore optimalement tout graphe parfait.

Combinaison de 3 ingrédients :

1. Choisir un problème : algo à écrire, propriété à montrer, ...
   Exemple : algorithme de coloration
2. Identifier les décompositions qui se comportent "bien" par rapport à ce problème-là
   Exemple : clique déconnectante
3. Se restreindre à une classe structurée dans laquelle on peut (presque) toujours trouver une décomposition listée à l'item 2.
   Exemple : graphe parfaits (avec le théorème ficitif de décomposition)

Décompositions

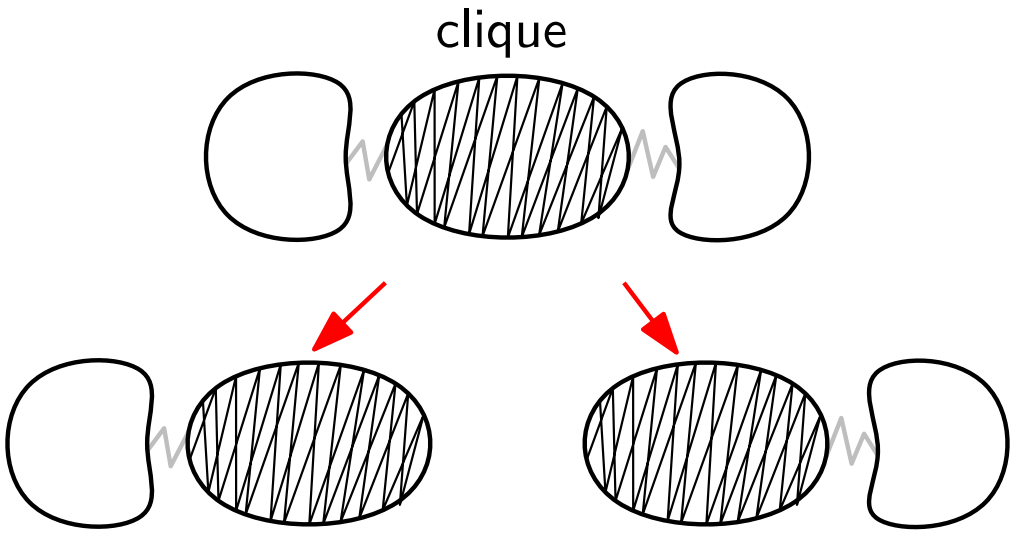
Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes
Compromis : s'autoriser à moins "diviser"



Décompositions

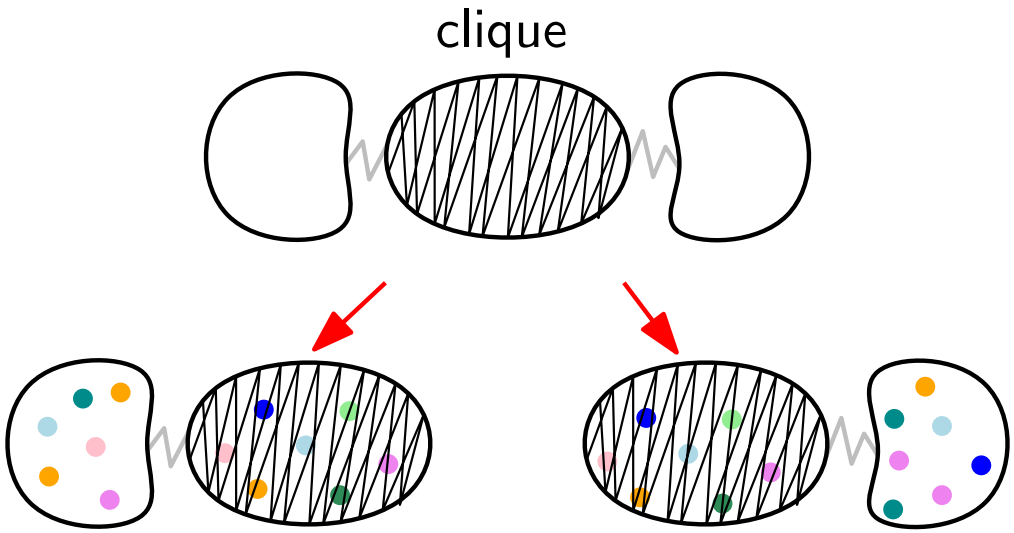
Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes

Compromis : s'autoriser à moins "diviser"



Décompositions

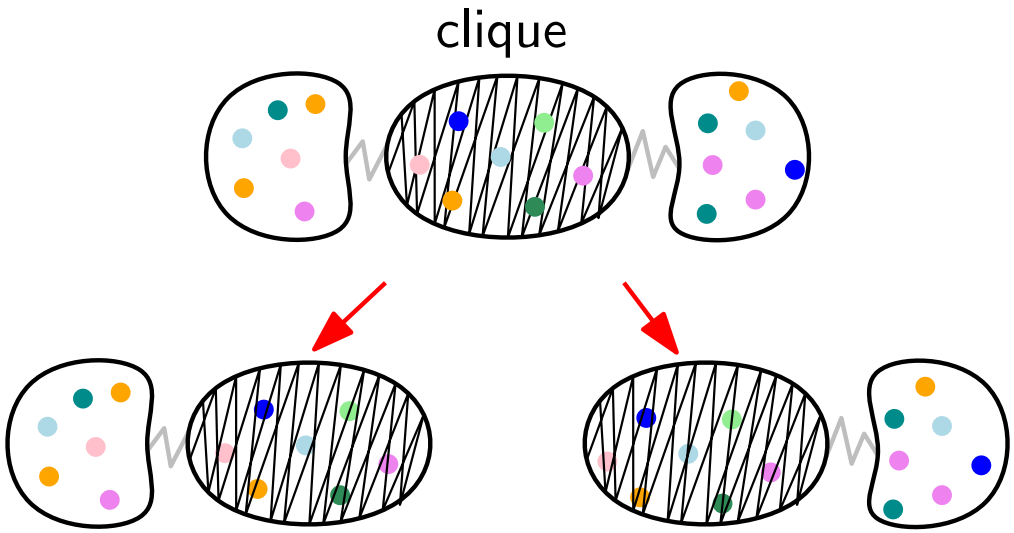
Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes
Compromis : s'autoriser à moins "diviser"



Décompositions

Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes

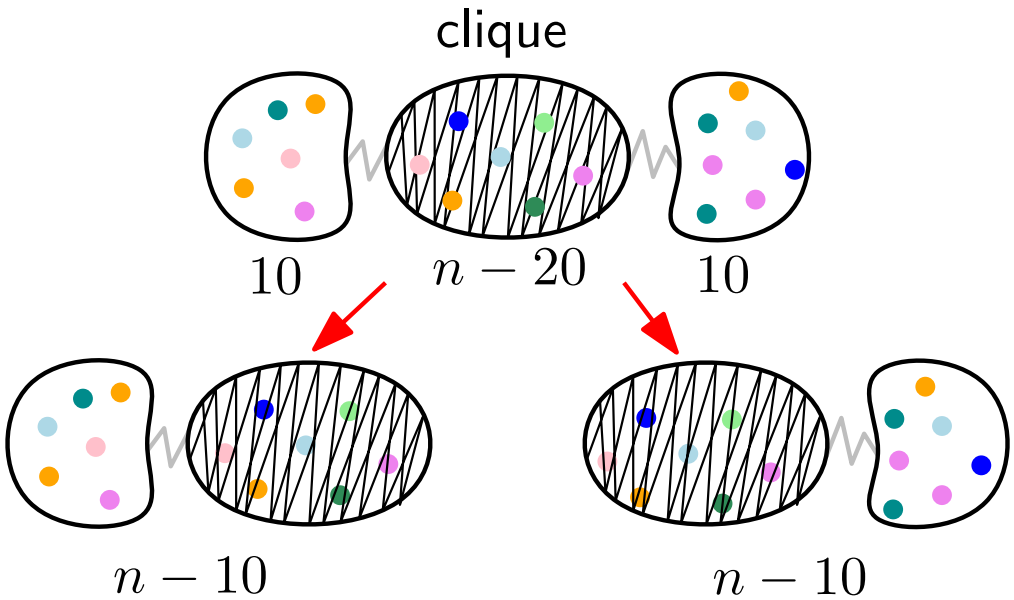
Compromis : s'autoriser à moins "diviser"



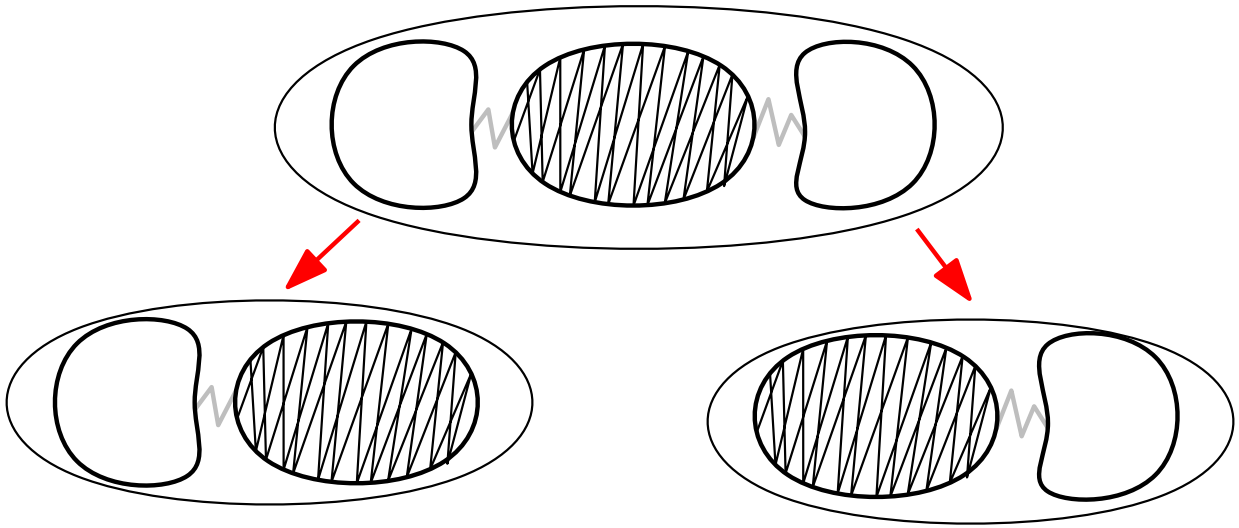
Décompositions

Approche "Diviser pour régner" $\rightarrow$ Ne capture pas assez de graphes

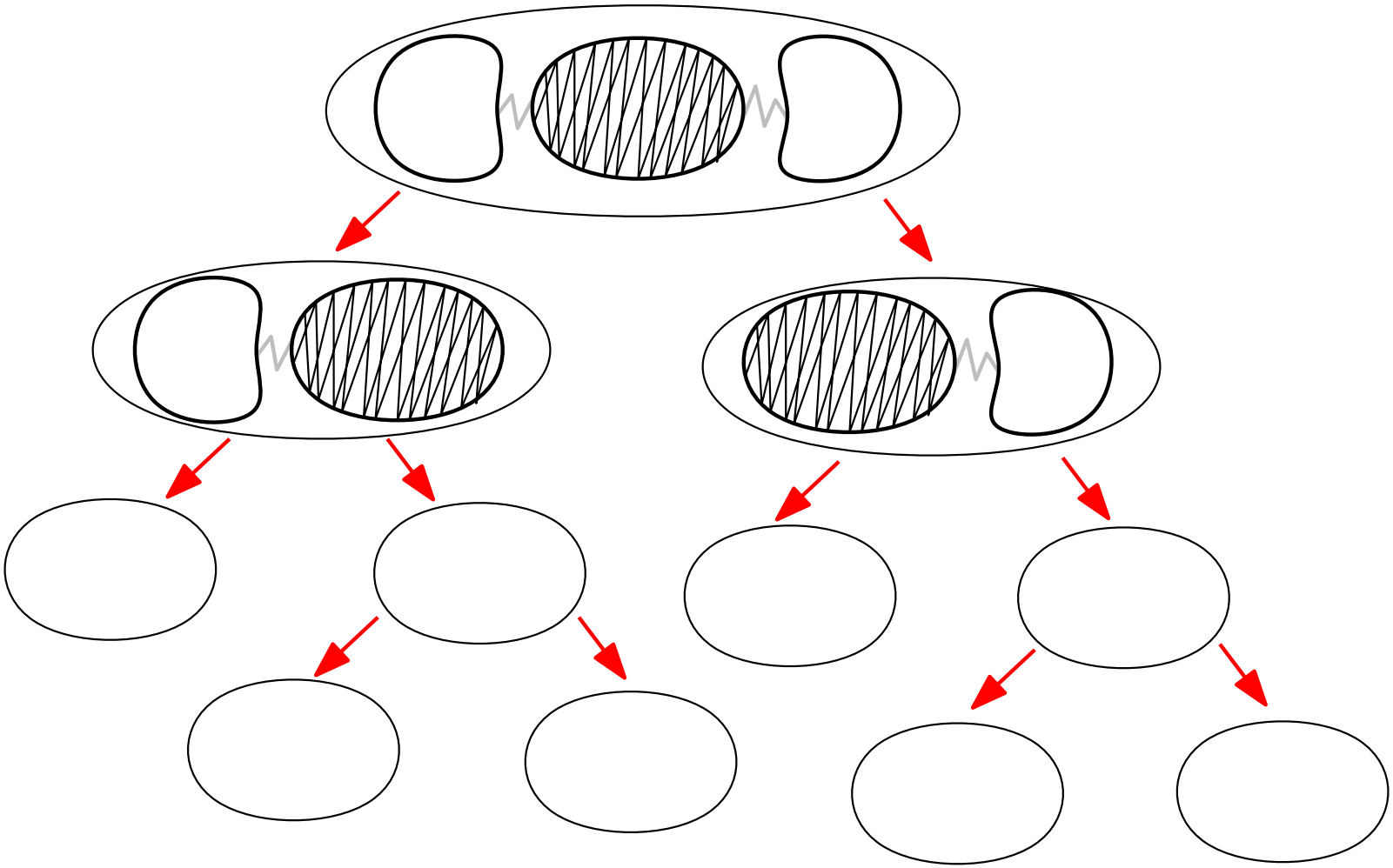
Compromis : s'autoriser à moins "diviser"



Gérer l'explosion des recouvrements



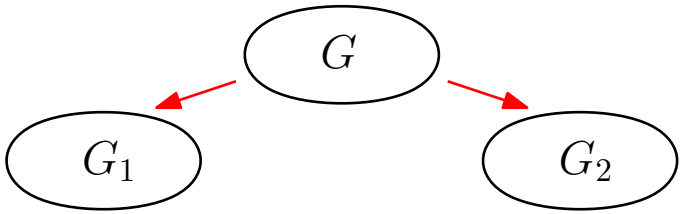
Gérer l'explosion des recouvrements



Algorithme de coloration :

Algorithme de coloration :

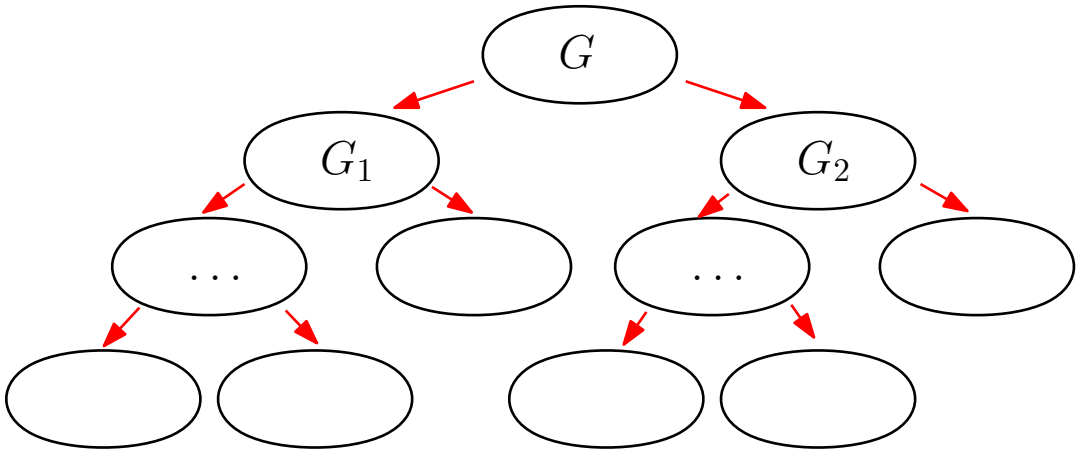
1. Décomposer le graphe récursivement en temps polynomial :



- Savoir trouver une décomposition en temps polynomial

Algorithme de coloration :

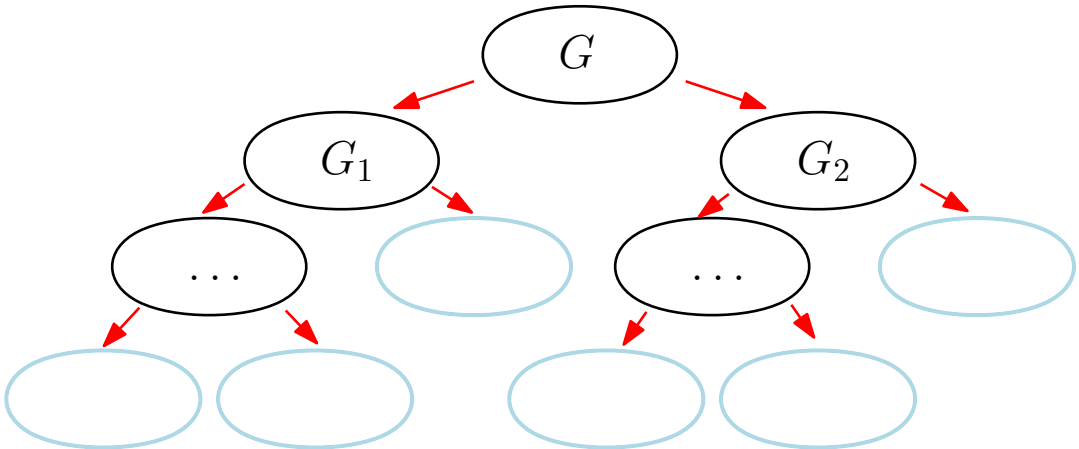
1. Décomposer le graphe récursivement en temps polynomial :



   - Savoir trouver une décomposition en temps polynomial
   - Prouver que la taille de l'arbre reste polynomiale

Algorithme de coloration :

1. Décomposer le graphe récursivement en temps polynomial :

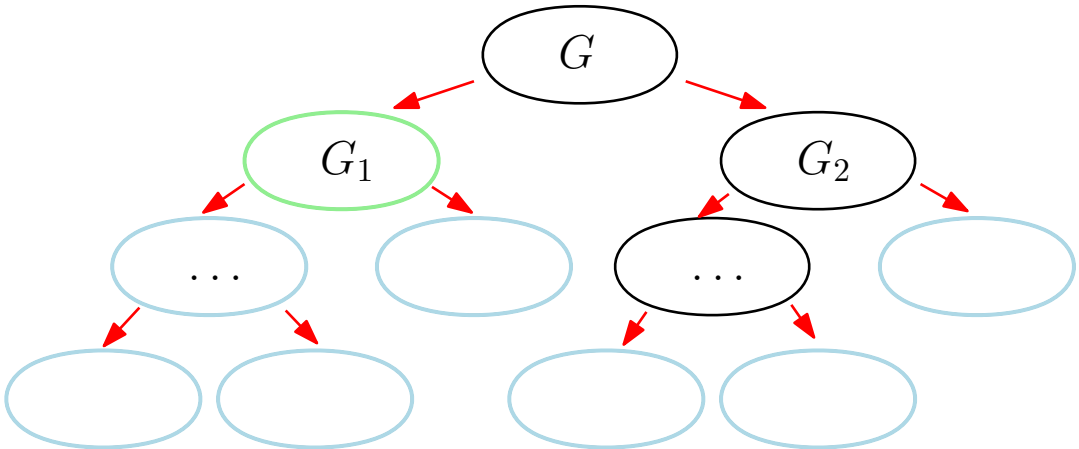


   - Savoir trouver une décomposition en temps polynomial
   - Prouver que la taille de l'arbre reste polynomiale

2. Calculer une coloration sur les **feuilles**
   ($\rightarrow$ graphes *plus simples*).

Algorithme de coloration :

1. Décomposer le graphe récursivement en temps polynomial :

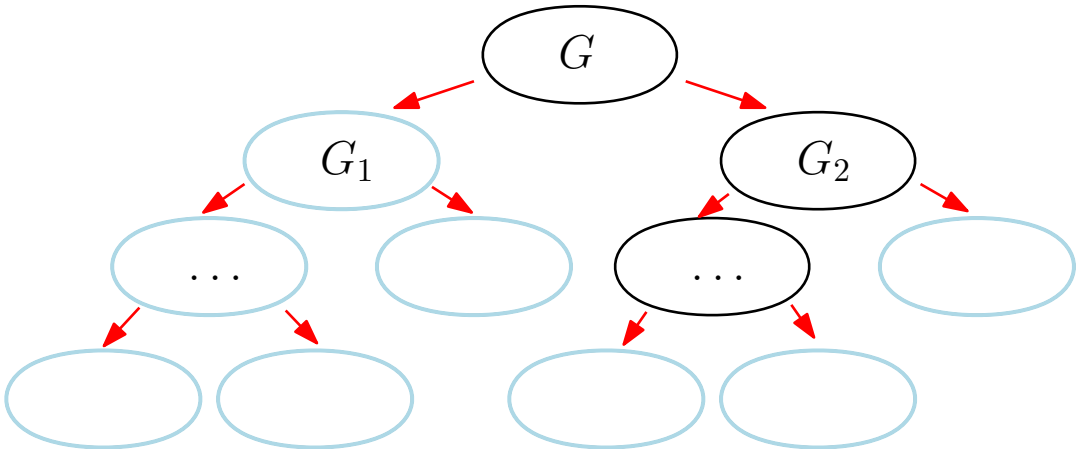


   - Savoir trouver une décomposition en temps polynomial
   - Prouver que la taille de l'arbre reste polynomiale

2. Calculer une coloration sur les **feuilles** ($\rightarrow$ graphes *plus simples*).
3. De bas en haut : **combiner les colorations des fils** pour obtenir une coloration du père.

Algorithme de coloration :

1. Décomposer le graphe récursivement en temps polynomial :



   - Savoir trouver une décomposition en temps polynomial
   - Prouver que la taille de l'arbre reste polynomiale

2. Calculer une coloration sur les **feuilles** ($\rightarrow$ graphes *plus simples*).
3. De bas en haut : **combiner les colorations des fils** pour obtenir une coloration du père.

Algorithme pour colorer les graphes parfaits

Question (2002 $\rightarrow$ ...)

Existe-t-il un algorithme de coloration purement combinatoire pour les graphes parfaits ? Peut-on utiliser le théorème de décomposition dans ce but ?

Algorithme pour colorer les graphes parfaits

Question (2002 $\rightarrow$ ...)

Existe-t-il un algorithme de coloration purement combinatoire pour les graphes parfaits ? Peut-on utiliser le théorème de décomposition dans ce but ?

Théoreme [Grötschel, Lovász, Schrijver 1981]

Grâce à la méthode de l'ellipsoïde, il existe un algorithme polynomial pour colorer optimalement tout graphe parfait.

$\rightarrow$ Algorithme non combinatoire.

Algorithme pour colorer les graphes parfaits

Question (2002 $\rightarrow$ ...)

Existe-t-il un algorithme de coloration purement combinatoire pour les graphes parfaits ? Peut-on utiliser le théorème de décomposition dans ce but ?

Résultats partiels :

Théorème [Chudnovsky, Trotignon, Trunck, Vušković 2015]

Oui si le graphe parfait en entrée n'admet pas de décomposition dite *BSP* ("mauvaise décomposition").

Théorème [Chudnovsky, Lagoutte, Seymour, Spirkl 2017]

Oui, en temps $\mathcal{O}(n^{(k+1)^2}) \rightarrow$ Polynomial si k est fixé.

Algorithme pour colorer les graphes parfaits

Question (2002 $\rightarrow$ ...)

Existe-t-il un algorithme de coloration purement combinatoire pour les graphes parfaits ? Peut-on utiliser le théorème de décomposition dans ce but ?

Résultats partiels :

Théorème [Chudnovsky, Trotignon, Trunck, Vušković 2015]

Oui si le graphe parfait en entrée n'admet pas de décomposition dite *BSP* ("mauvaise décomposition").

Théorème [Chudnovsky, Lagoutte, Seymour, Spirkl 2017]

Oui, en temps $\mathcal{O}(n^{(k+1)^2}) \rightarrow$ Polynomial si k est fixé.

Merci pour votre attention !