Introduction à la théorie de la complexité

Informatique Génomique - Master 1

Stéphane Vialette

LIGM UMR 8049,
Bureau 4B066
Université Paris-Est Marne La Vallée
vialette@univ-mlv.fr
http://igm.univ-mlv.fr/∼vialette

13 février 2012

Plan

Plan

La complexité algorithmique

Idées intuitives

- Le temps d'exécution d'un algorithme dépend de la taille de l'entrée.
- Les constantes sont ignorées (considérées comme négligeables devant la taille de l'entrée lorsque celle-ci est grande).
- Quelle complexités
  - Complexité en pire cas.
  - Complexité en moyenne.
  - Complexité en espace.
  - . . .

La complexité algorithmique

Mesurer la complexité

- Abstraction par rapport à la machine utilisée.
- Abstraction par rapport aux données : taille uniquement
- Notation $\mathcal{O}$.
- Critère d'efficacité : temps d'exécution polynomial.

Plan

La notation $\mathcal{O}$

Definition

Soient f et g deux fonctions croissantes. Alors $f(n) = \mathcal{O}(g(n))$ s'il existe un entier positif N et une constante c tels que

$$\forall\, n \geq N, \quad f(n) \leq c\, g(n)$$

- n désigne en général la *taille du problème*.
- "f *est majorée par* g" .

Quelques propriétés de la notation $\mathcal{O}$

Soient f, g, r et s quatre fonctions et c une constante.

Si $f(n) = \mathcal{O}(s(n))$ et $g(n) = \mathcal{O}(r(n))$ alors

$$f(n) + g(n) = \mathcal{O}(r(n) + s(n)) = \mathcal{O}(max\{r(n), s(n)\})$$

Si $f(n) = \mathcal{O}(s(n))$ et $g(n) = \mathcal{O}(r(n))$ alors

$$f(n)\ g(n) = \mathcal{O}(r(n)\ s(n))$$

$$\mathcal{O}(c\ f(n)) = c\ \mathcal{O}(f(n)) = \mathcal{O}(f(n))$$

Ω et θ

Definition

Soient f et g deux fonctions croissantes. Alors $f(n) = \Omega(g(n))$ s'il existe un entier positif N et une constante c tels que

$$\forall\, n \geq N, \quad f(n) \geq c\, g(n)$$

$f(n)$ domine toujours $g(n)$

Definition

Soient f et g deux fonctions croissantes. Si $f(n) = \mathcal{O}(g(n))$ et $f(n) = \Omega(g(n))$, alors

$$f(n) = \theta(g(n))$$

Complexité algorithmique : une approche concrête

$\log n$	$\sqrt{n}$	n	$n \log n$	n^2	2^n
4	4	10	34	100	1024
7	10	100	665	10 000	$\sim 10^{30}$
10	32	1 000	9966	1 000 000	$\sim 10^{300}$
14	100	10 000	132 878	100 000 000	overflow
17	317	100 000	1 660 965	10^{10}	overflow
20	1000	1 000 000	19 931 569	10^{12}	overflow

Plan

Problèmes . . .

Un problème est composé :

- d'un ensemble d'instances,
- d'une "*question*" sur ces instances,
- d'un ensemble de solutions pour chaque instance.

Problèmes . . .

Un problème est composé :

- d'un ensemble d'instances,
- d'une "*question*" sur ces instances,
- d'un ensemble de solutions pour chaque instance.

Exemple

Arbre couvrant de poids minimum $\equiv$ *un sous-ensemble qui est un arbre et qui connecte tous les sommets ensemble et dont la somme des poids des arcs est mini.*

Problèmes . . .

Un problème est composé :

- d'un ensemble d'instances,
- d'une "*question*" sur ces instances,
- d'un ensemble de solutions pour chaque instance.

Exemple

Arbre couvrant de poids minimum

- *Instances : tous les graphes possibles.*

Problèmes . . .

Un problème est composé :

- d'un ensemble d'instances,
- d'une "*question*" sur ces instances,
- d'un ensemble de solutions pour chaque instance.

Exemple

Arbre couvrant de poids minimum

- *Instances : tous les graphes possibles.*
- *Question : un arbre couvrant de poid minimum . . .*

Problèmes . . .

Un problème est composé :

- d'un ensemble d'instances,
- d'une "*question*" sur ces instances,
- d'un ensemble de solutions pour chaque instance.

Exemple

Arbre couvrant de poids minimum

- *Instances : tous les graphes possibles.*
- *Question : un arbre couvrant de poid minimum . . .*
- *Solutions : tous les arbres couvrants.*

Problèmes . . . notations

Definition

Un problème Π est composé :

- d'un ensemble d'instances noté $\text{inst}(\Pi)$,
- d'une "*question*" sur ces instances,
- d'un ensemble de solutions, noté $\text{sol}_\Pi(x)$, pour chaque instance $x \in \text{inst}(\Pi)$.

Question . . .

- Détermine la nature du problème.

Problème de décision vs. d'optimisation

Problème de décision

Problème dont la réponse attendue est *oui* ou *non*.

- L'objet X vérifie-t-il la propriété P ?
- Notion centrale dans la théorie de la NP-complétude.

Problème de décision vs. d'optimisation

Problème de décision

Problème dont la réponse attendue est *oui* ou *non*.

- L'objet X vérifie-t-il la propriété P ?
- Notion centrale dans la théorie de la NP-complétude.

Problème d'optimisation

Problème dont la réponse attendue est une valeur calculée.

- Maximisation ou minimization d'une certaine fonction.
- La valeur attendue n'est pas nécessairement numérique, *e.g.*, une plus grande sous-séquence commune.

Problème de comptage vs. d'énumération

Problème de comptage

Problème dont la réponse attendue est le nombre de solutions.

- Calculer le nombre de solutions est plus difficile que décider s'il existe au moins une solution.
- Il n'est pas demandé de les énumérer.

Problème de comptage vs. d'énumération

Problème de comptage

Problème dont la réponse attendue est le nombre de solutions.

- Calculer le nombre de solutions est plus difficile que décider s'il existe au moins une solution.
- Il n'est pas demandé de les énumérer.

Problème d'énumération

Problème dont la réponse attendue est l'ensemble des solutions.

- La complexité dépend de toute façon du nombre de solutions.
- Si nous savons énumérer les solutions, alors nous pouvons en même temps les compter.

Machine déterministe vs. non-déterministe

Machine déterministe

- Machine exécutant une séquence d'instructions bien déterminée.
- Notion classique en algorithmique.

Machine non-déterministe

- Machine ayant la capacité de toujours choisir la meilleur séquences d'instructions qui mène à la bonne solution (lorsque celle-ci existe).
- N'existe pas ailleurs que dans l'esprit d'un théoricien !
- Cette machine abstraite est centrale dans la théorie de la NP-complétude.

La classe P

Le classe P contient tous les problèmes relativements faciles, c'est-à-dire ceux pour lesquels nous connaissons des algorithmes efficaces.

Definition (La classe P)

La classe P est constituée de tous les problèmes pour lesquels nous pouvons construire une machine déterministe en temps polynomial.

- La lettre P signifie Polynomial time.

Vérificateur en temps polynomial

Definition

Un vérificateur en temps polynomial pour un problème Π est une machine *déterministe* M qui, pour toute paire (x, y) où $x \in \text{inst}(\Pi)$, retourne *oui* en temps polynomial en la taille de x si et seulement si $y \in \text{sol}_\Pi(x)$.

- Pouvons-nous vérifier en temps polynomial une solution ?
- Notion centrale dans le théorie de la NP-complétude.

La classe NP

Le classe NP contient tous les problèmes dont nous pouvons vérifier une solution en temps polynomial.

Definition (La classe NP)

La classe NP est constituée de tous les problèmes pour lesquels nous pouvons construire une machine non-déterministe en temps polynomial.

- Les lettres NP signifient Non-Deterministic Polynomial time (et non Non-Polynomial Time).

NP et vérificateur en temps polynomial

Théorème

Un problème Π *appartient à la classe* NP *si et seulement si il existe un vérificateur en temps polynomial pour* Π.

Transformation polynomiale

Passer d'un problème à un autre . . .

- Dans le cadre des problèmes décidables, une transformation permet de ramener la décidabilité d'un problème à celle d'un autre problème.
- On introduit des transformations qui prennent en compte le temps de calcul de la fonction qui fait passer d'un problème Π à un autre problème Π'.
- Une transformation polynomiale fait correspondre à une instance du problème Π une instance du problème Π' qui a la même réponse et qui est calculable en temps polynomial.

Transformation polynomiale

Definition (Transformation polynomiale)

Soient Π et Π' deux problèmes de décision. Une transformation polynomiale de Π' vers Π, notée $\Pi' \leq_{\mathbf{P}} \Pi$, est une fonction $f : \text{inst}(\Pi') \to \text{inst}(\Pi)$ qui satisfait les conditions suivantes :

1. f est calculable en temps polynomial ;
2. $\forall\, x, \quad x \in \text{inst}(\Pi')$ ssi $f(x) \in \text{inst}(\Pi)$.

NP-difficile et NP-complet

Definition (NP-difficile)

Un problème de décision Π est NP-difficile si tout problème de décision $\Pi' \in \mathbf{NP}$ se réduit polynomialement à Π, *i.e.*,

$$\forall\, \Pi' \in \mathbf{NP}, \qquad \Pi' \leq_{\mathbf{P}} \Pi$$

Definition (NP-complet)

Un problème de décision Π est NP-complet si les deux conditions suivantes sont vérifiées :

1. Π est NP-difficile $\quad (\forall\, \Pi' \in \mathbf{NP}, \qquad \Pi' \leq_{\mathbf{P}} \Pi)$,
2. $\Pi \in \mathbf{NP}$.

Propriétés de $\leq_{\mathbf{P}}$

P et NP

Si $\Pi' \leq_{\mathbf{P}} \Pi$, alors

- Si $\Pi \in \mathbf{P}$ alors $\Pi' \in \mathbf{P}$,
- Si $\Pi' \notin \mathbf{P}$ alors $\Pi \notin \mathbf{P}$,

Transitivité

Si $\Pi' \leq_{\mathbf{P}} \Pi$ et $\Pi \leq_{\mathbf{P}} \Pi''$, alors

$$\Pi' \leq_{\mathbf{P}} \Pi''$$

Deux propriétés immédiates et une remarque

1. Si Π est NP-complet et $\Pi \in \mathbf{P}$, alors $\mathbf{P} = \mathbf{NP}$.

2 Si Π est NP-complet et $\mathbf{P} \neq \mathbf{NP}$, alors il n'existe aucun algorithme en temps polynomial pour Π.

- $\mathbf{P} = \mathbf{NP}$ (?)
- aucune preuve à ce jour . . .

Remarque

Il n'est *à priori* pas évident qu'il existe au moins un problème NP-complet.

Satisfaisabilité des formules booléennes

Satisfaisabilité des formules booléennes

- Variables boolénnes $X = \{x_1, x_2, \ldots, x_n\}$
- Clause : disjonction de litéraux, *e.g.*,

$$(x_4 \vee \overline{x_{15}} \vee x_{23} \vee \overline{x_{24}})$$

- Forme normale conjonctive : conjonction de clauses, *e.g.*,

$$\ldots \wedge (x_2 \vee x_3) \wedge (x_4 \vee \overline{x_{15}} \vee x_{23} \vee \overline{x_{24}}) \wedge (x_2 \vee \overline{x_{15}} \vee \overline{x_{23}}) \wedge \ldots$$

- Une affectation est une fonction $\nu : X \to \{\texttt{vrai}, \texttt{faux}\}$.
- Une formule est satisfaisable s'il existe une affection qui satisfait la formule.

Satisfaisabilité des formules booléennes

Satisfaisabilité des formules booléennes

- Toutes les formules booléennes ne sont pas satisfaisables, *e.g.*,

$$\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$$

- Soit ϕ une formule boolénne sous *forme normale conjonctive*. Alors ϕ est satisfaisable si et seulement si chaque clause est satisfaite.
- Toute formule booléenne peut être mise sous forme normale conjonctive.
- Il existe 2^n affectations différentes.

Le problème SAT est NP-complet

SAT

Instance : Une formule booléenne ϕ sous forme normale conjonctive (CNF).

Question : La formule ϕ est-elle satisfaisable ?

Le problème SAT est NP-complet

SAT

Instance : Une formule booléenne ϕ sous forme normale conjonctive (CNF).

Question : La formule ϕ est-elle satisfaisable ?

Théorème (Cook 1971)

Le problème SAT *est* NP*-complet.*

Le problème k-SAT est NP-complet

k-SAT

Instance : Une formule booléenne ϕ sous forme normale conjonctive (CNF) où chaque clause contient au plus k littéraux.

Question : La formule ϕ est-elle satisfaisable ?

Le problème k-SAT est NP-complet

k-SAT

Instance : Une formule booléenne ϕ sous forme normale conjonctive (CNF) où chaque clause contient au plus k littéraux.

Question : La formule ϕ est-elle satisfaisable ?

Théorème

Le problème k-SAT, $k \geq 3$, est NP*-complet.*

Quelques remarques sur le problème SAT

- Il existe un algorithme en temps polynomial (en fait linéaire) pour le problème 2-SAT.
- Une formule booléenne de Horn est une conjonction de clauses telle que chaque clause contient au plus une variable positive, *e.g.*,

$$(\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4}) \wedge (x_2)$$

  Il existe un algorithme en temps polynomial pour le problème SAT restreint aux formules booléennes de Horn.
- Il existe des algorithmes "*rapides*" pour résoudre des instances du problème SAT (`Sat Contest`).

Quelques variantes difficiles du problème SAT

1/3-SAT

Instance : Une formule booléenne ϕ sous forme normale conjonctive (CNF) où chaque clause contient au plus 3 littéraux.

Question : Existe-t-il une affectation qui satisfait exactement un littéral par clause ?

Théorème

Le probème 1/3-SAT *est* NP*-complet.*

Quelques variantes difficiles du problème SAT

NEQ-3-SAT

Instance : Une formule booléenne ϕ sous forme normale conjonctive (CNF) où chaque clause contient au plus 3 littéraux.

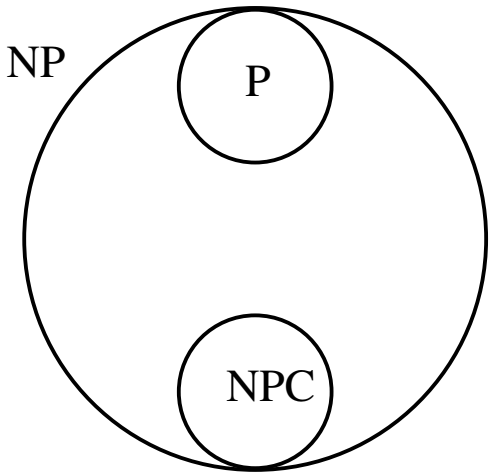
Question : Existe-t-il une affectation qui satisfait au moins 1 littéral par clause et au plus 2 ?

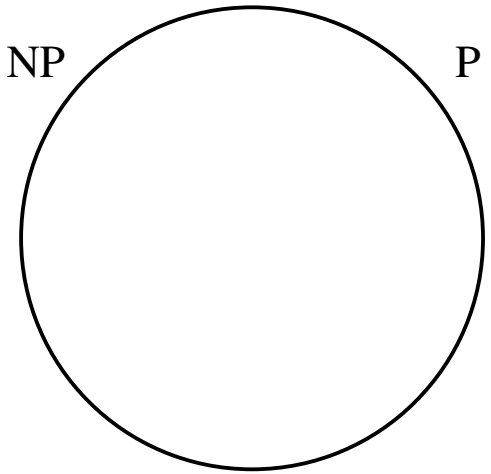
Théorème

Le probème NEQ-3-SAT *est* NP*-complet.*

Relations entre P et NP . . .

- Les problèmes NP-complets sont les problèmes les plus difficiles de la classe NP.
- **P** $\subseteq$ **NP**
- mais





Ensemble Stable

Ensemble Stable

Instance : Un graphe G d'ordre n et un entier positif k.

Question : Existe-t-il un ensemble stable de taille k dans G, *i.e.*, un sous-ensemble de k sommets non connectés deux à deux ?

- Problème de maximisation.
- Problème très étudié.
- Incompatibilités, . . .

Notre première preuve de NP-complétude

Théorème

Le problème Ensemble Stable *est* NP*-complet.*

Notre première preuve de NP-complétude

Théorème

Le problème Ensemble Stable *est* NP*-complet.*

Démonstration.

- Ensemble Stable $\in$ **NP**
- 3-Sat $\leq_{\mathbf{P}}$ Ensemble Stable

Notre première preuve de NP-complétude

Théorème

Le problème Ensemble Stable *est* NP*-complet.*

Démonstration.

- Il existe un vérificateur en temps polynomial pour le problème Ensemble Stable, et par conséquent

$$\text{Ensemble Stable} \in \mathbf{NP}$$

□

Notre première preuve de NP-complétude

Théorème

Le problème Ensemble Stable *est* NP*-complet.*

Démonstration.

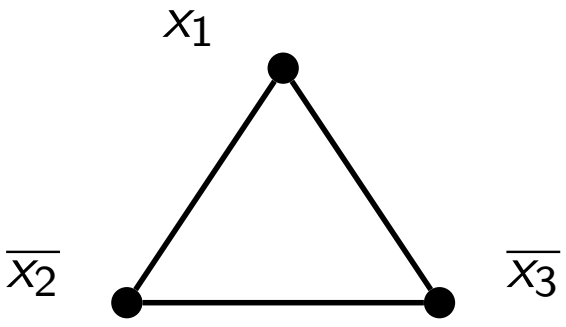
- Réduction polynomiale depuis le problème 3-Sat.
- $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ un formule booléenne sous forme normale conjonctive.
- Il faut construire un graphe G et un entier positif k tels que ϕ est satisfaisable si et seulement si il existe un ensemble stable de taille k dans G.

Le problème Ensemble Stable est NP-complet

$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

Le problème Ensemble Stable est NP-complet

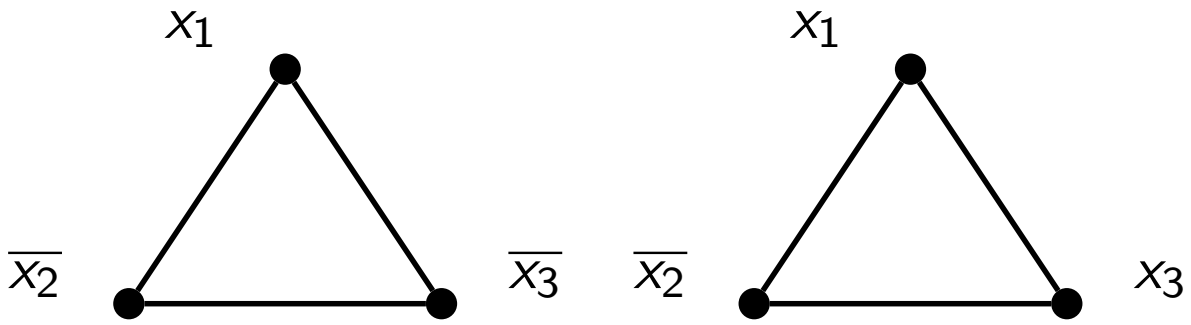
$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



$k = 3$ (k = nombre de clauses)

Le problème Ensemble Stable est NP-complet

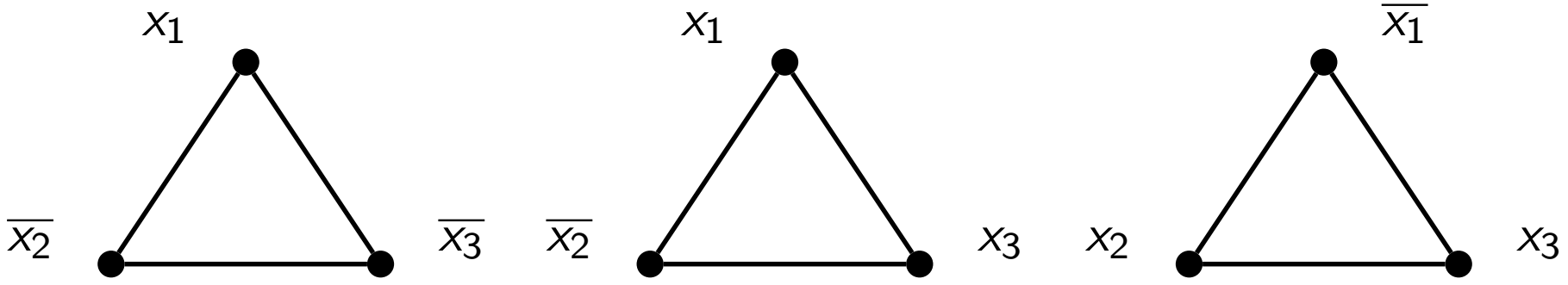
$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



$k = 3$ ($k =$ nombre de clauses)

Le problème ENSEMBLE STABLE est NP-complet

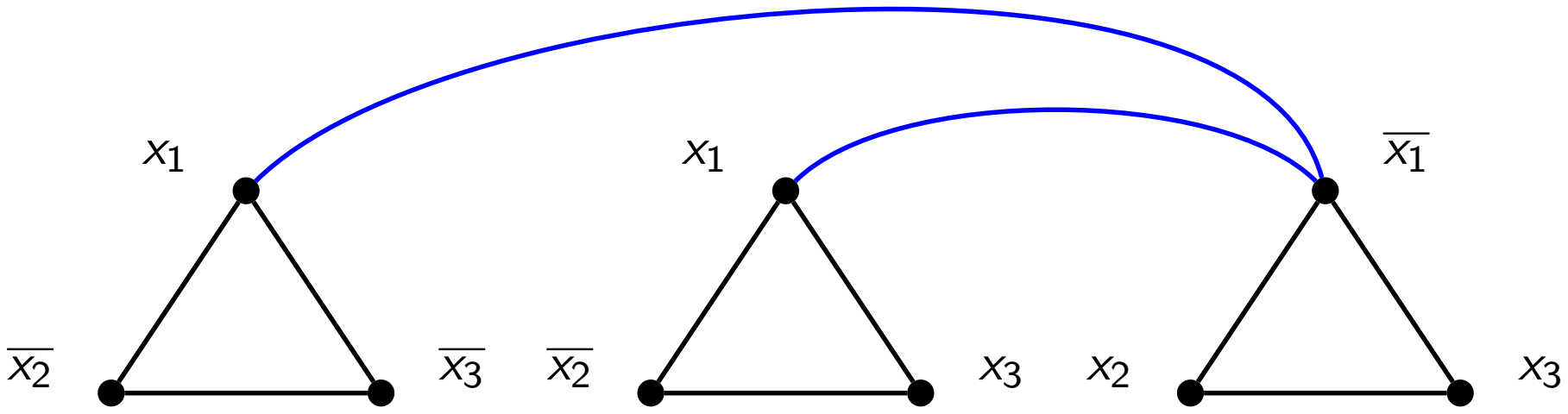
$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



$k = 3$ (k = nombre de clauses)

Le problème Ensemble Stable est NP-complet

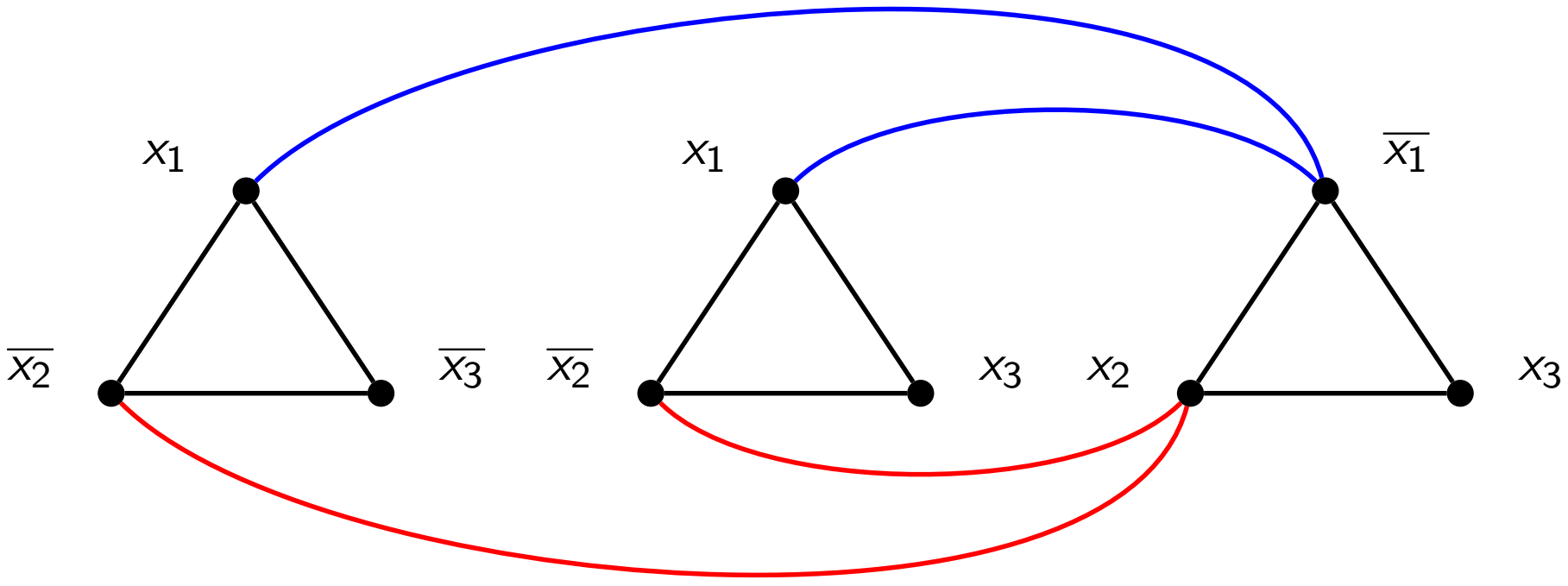
$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



$k = 3$ (k = nombre de clauses)

Le problème ENSEMBLE STABLE est NP-complet

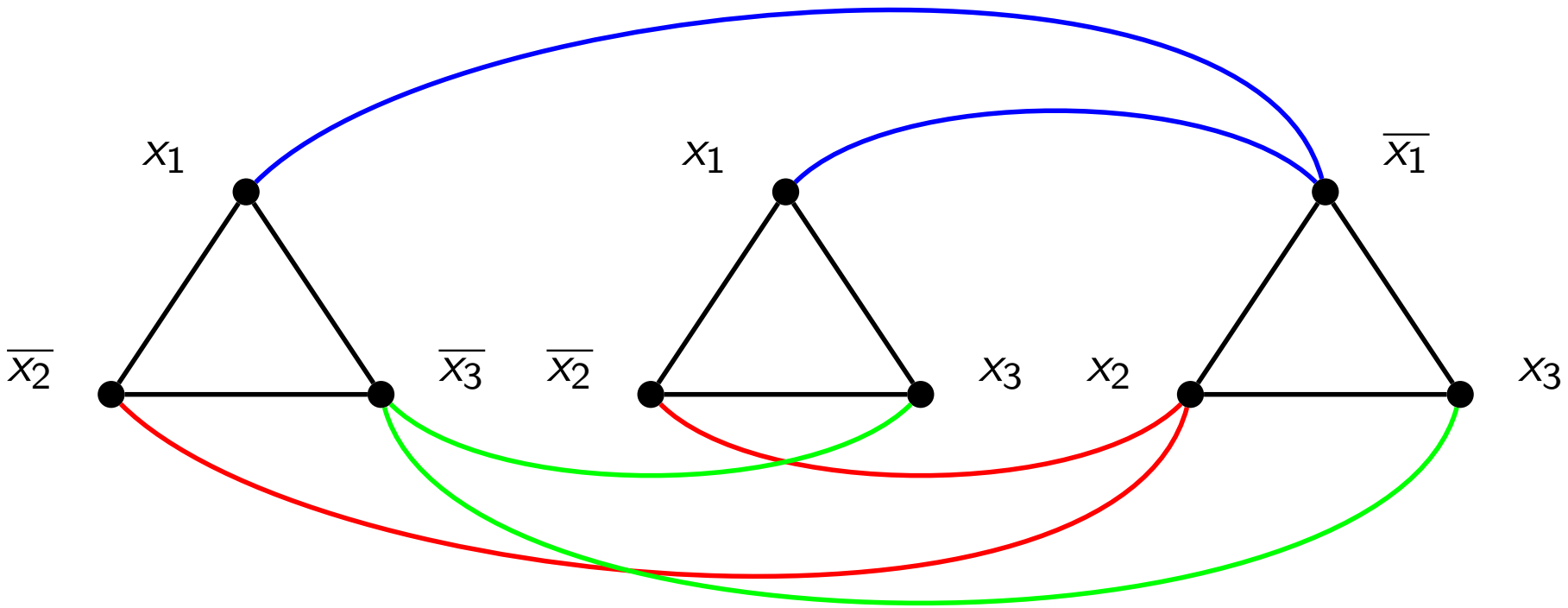
$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



$k = 3$ (k = nombre de clauses)

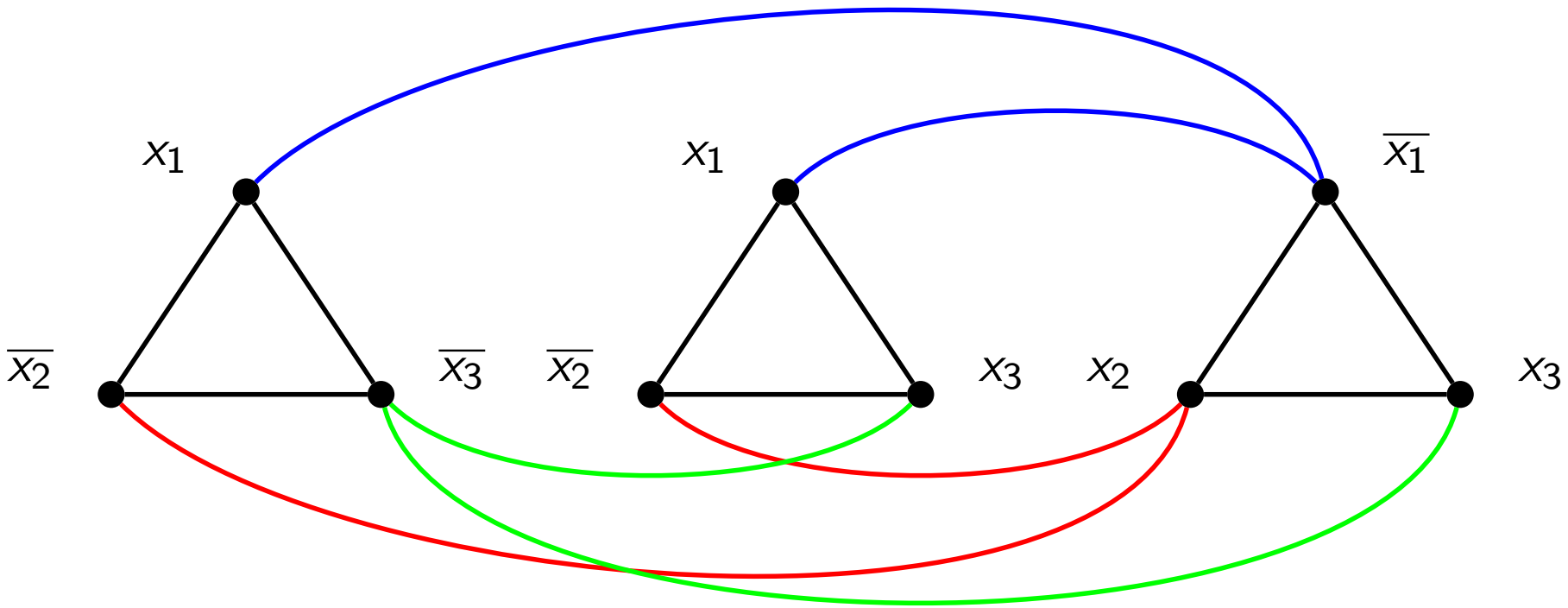
Le problème Ensemble Stable est NP-complet

$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



Le problème ENSEMBLE STABLE est NP-complet

$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



$k = 3$ (k = nombre de clauses)

Le problème Ensemble Stable est NP-complet

($\Rightarrow$) Si ϕ est satisfaisable, alors il existe un ensemble stable de taille $k = 3$ dans G.

Le problème Ensemble Stable est NP-complet

($\Rightarrow$) Si ϕ est satisfaisable, alors il existe un ensemble stable de taille $k = 3$ dans G.

$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

Le problème Ensemble Stable est NP-complet

($\Rightarrow$) Si ϕ est satisfaisable, alors il existe un ensemble stable de taille $k = 3$ dans G.

$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

Le problème Ensemble Stable est NP-complet

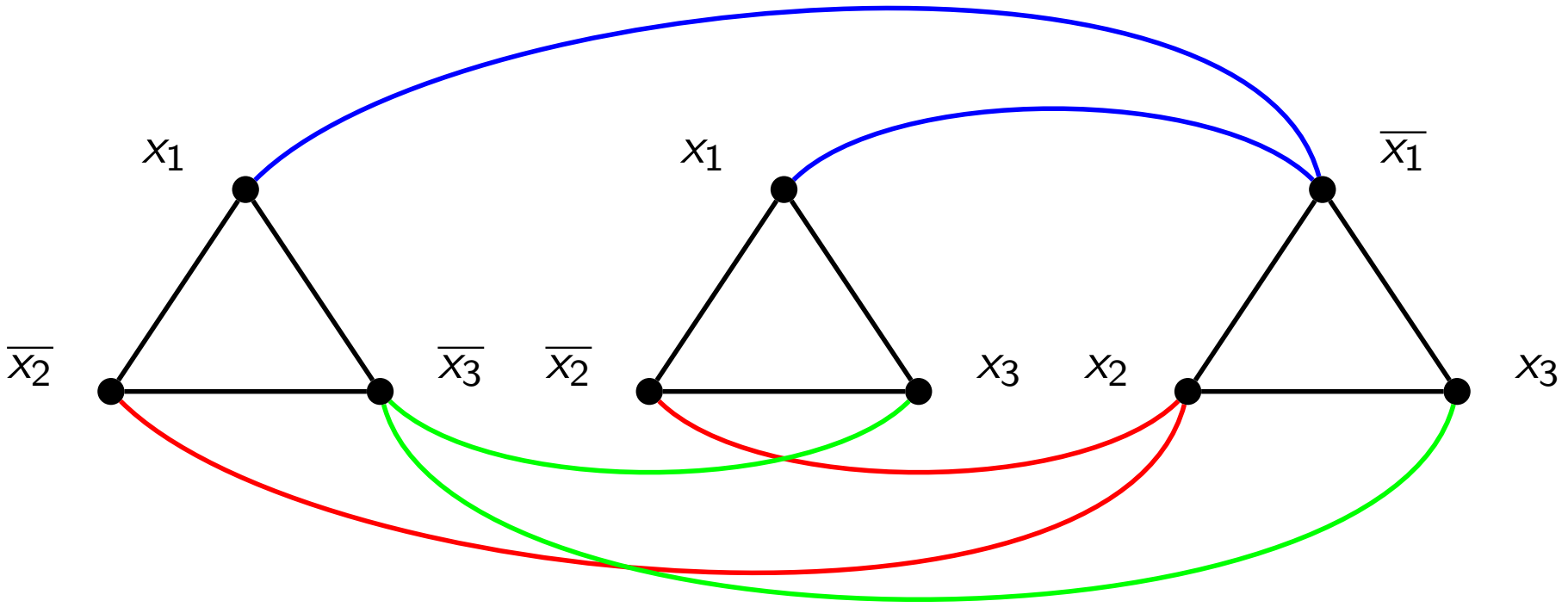
($\Rightarrow$) Si ϕ est satisfaisable, alors il existe un ensemble stable de taille $k = 3$ dans G.

$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

Le problème Ensemble Stable est NP-complet

($\Rightarrow$) Si ϕ est satisfaisable, alors il existe un ensemble stable de taille $k = 3$ dans G.

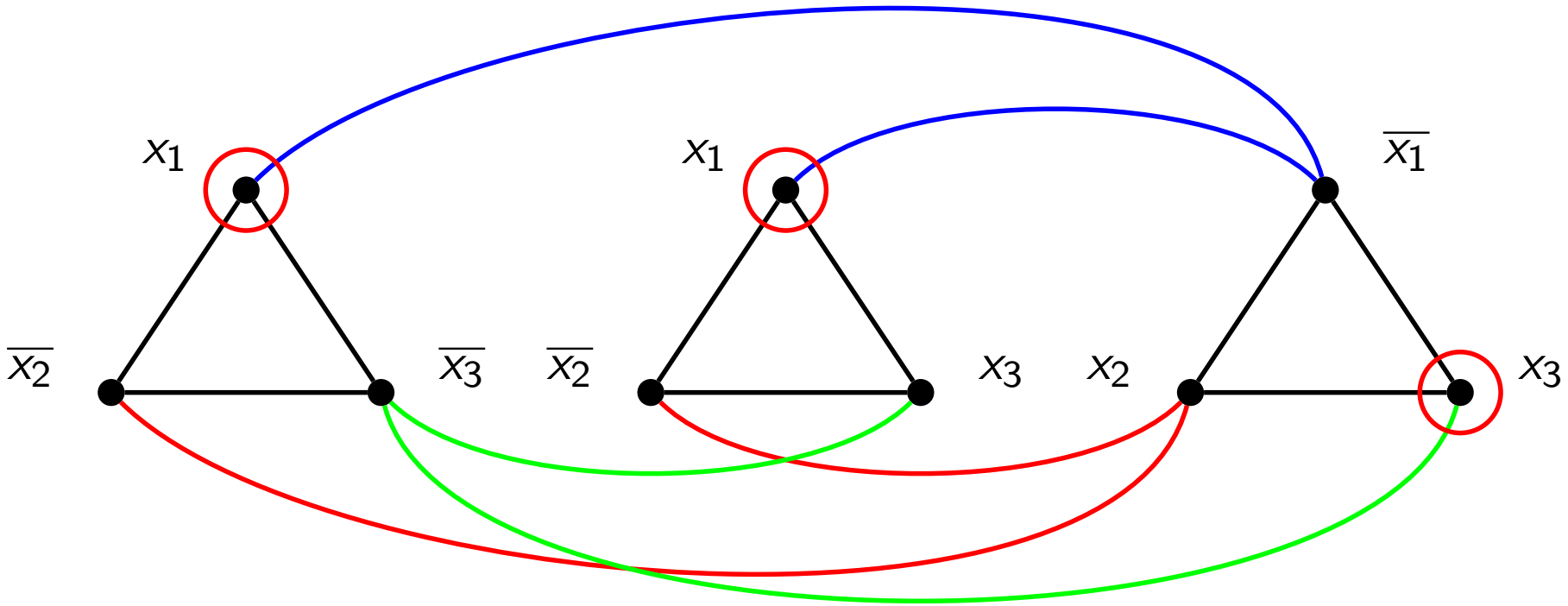
$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



Le problème Ensemble Stable est NP-complet

($\Rightarrow$) Si ϕ est satisfaisable, alors il existe un ensemble stable de taille $k = 3$ dans G.

$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



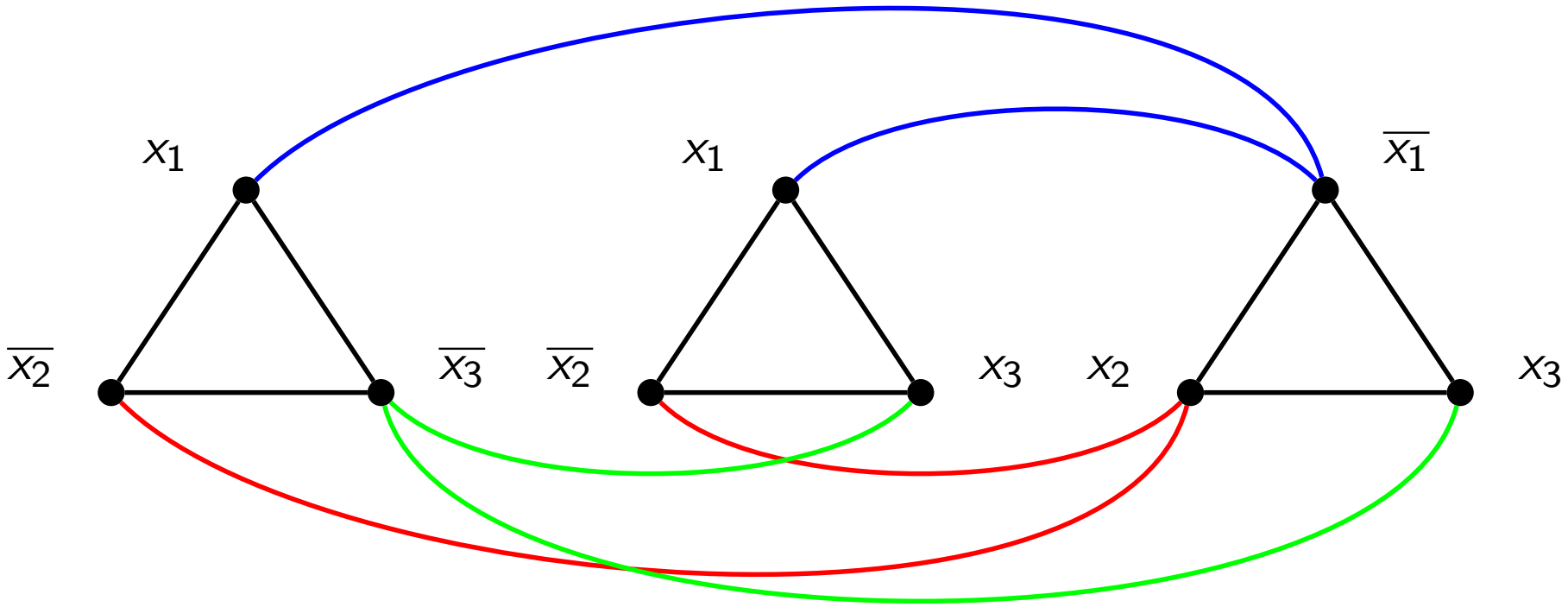
Le problème Ensemble Stable est NP-complet

($\Leftarrow$) S'il existe un ensemble stable de taille $k = 3$ ($k =$ le nombre de clauses de ϕ), alors ϕ est satisfaisable.

Le problème Ensemble Stable est NP-complet

($\Leftarrow$) S'il existe un ensemble stable de taille $k = 3$ ($k =$ le nombre de clauses de ϕ), alors ϕ est satisfaisable.

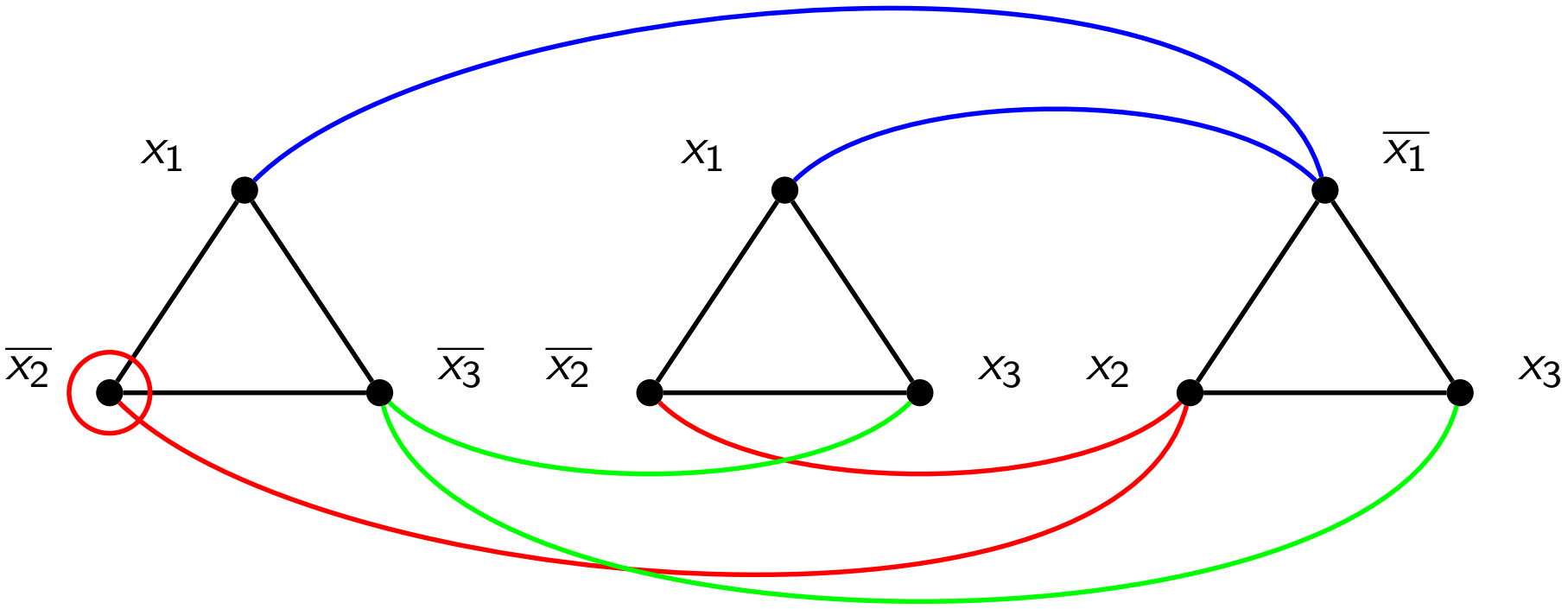
$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



Le problème ENSEMBLE STABLE est NP-complet

($\Leftarrow$) S'il existe un ensemble stable de taille $k = 3$ ($k =$ le nombre de clauses de ϕ), alors ϕ est satisfaisable.

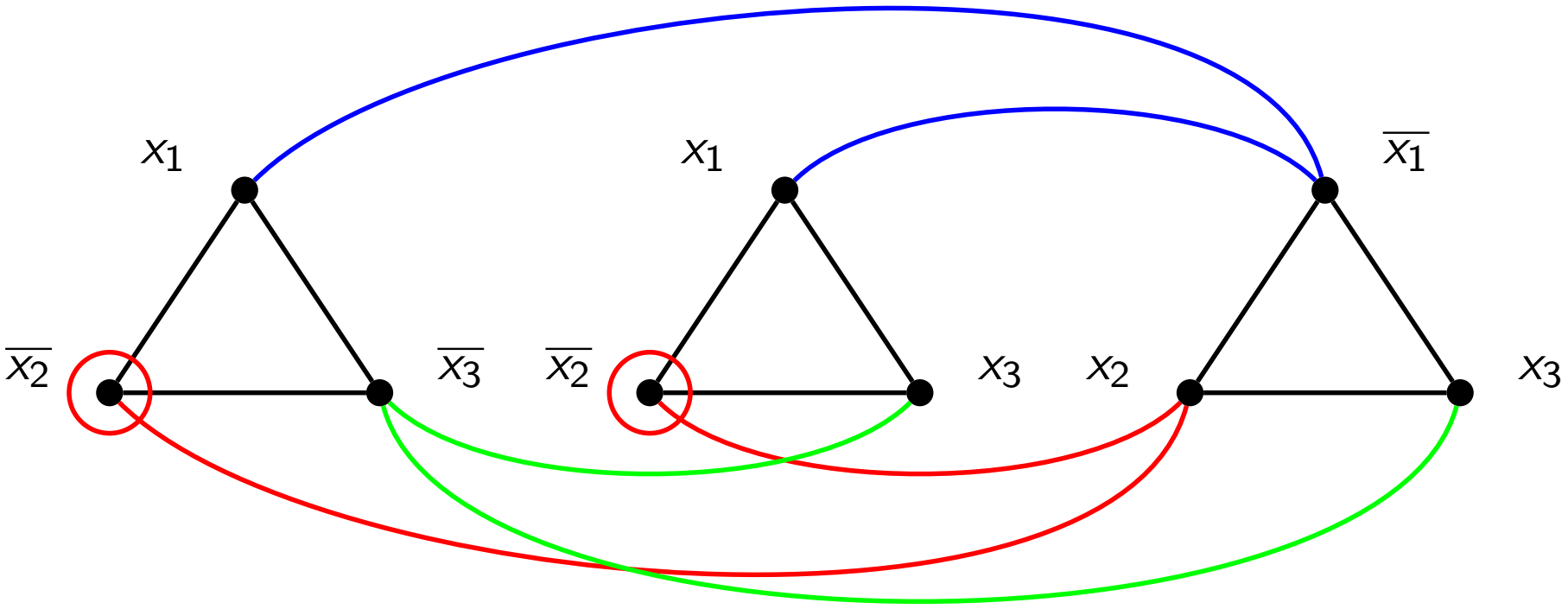
$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



Le problème ENSEMBLE STABLE est NP-complet

($\Leftarrow$) S'il existe un ensemble stable de taille $k = 3$ ($k =$ le nombre de clauses de ϕ), alors ϕ est satisfaisable.

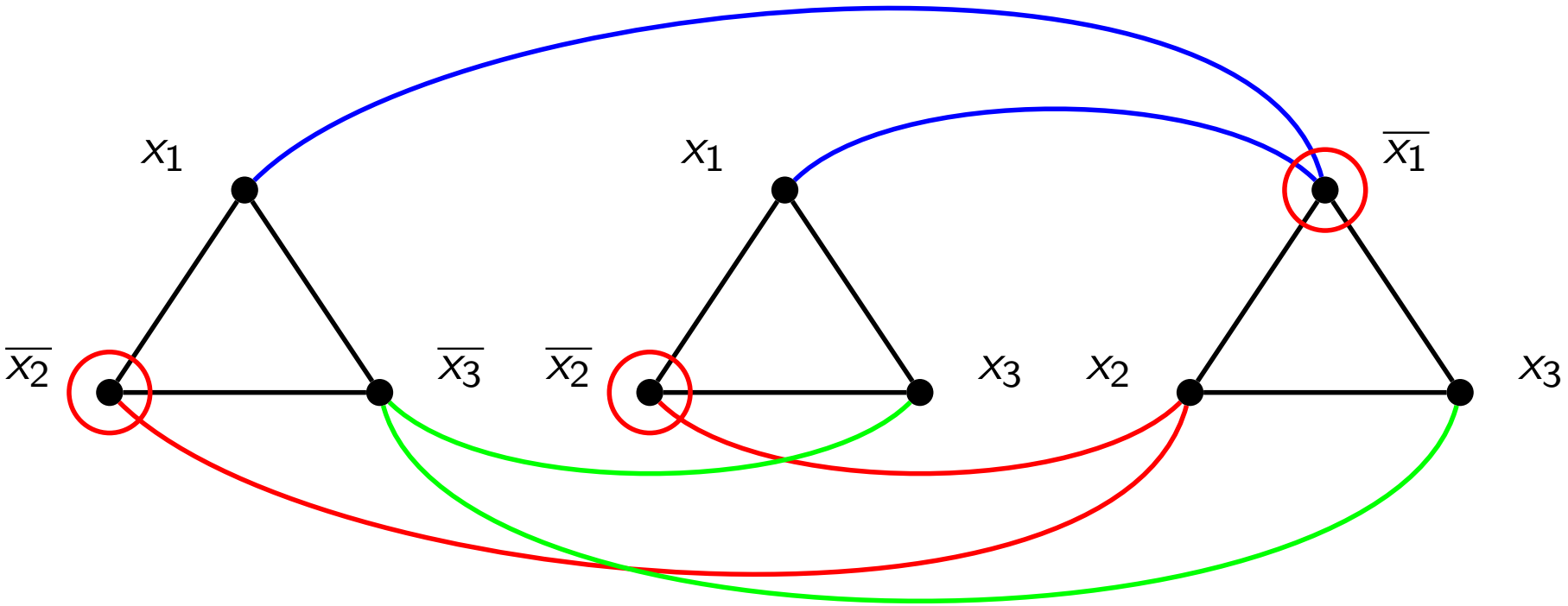
$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



Le problème ENSEMBLE STABLE est NP-complet

($\Leftarrow$) S'il existe un ensemble stable de taille $k = 3$ ($k =$ le nombre de clauses de ϕ), alors ϕ est satisfaisable.

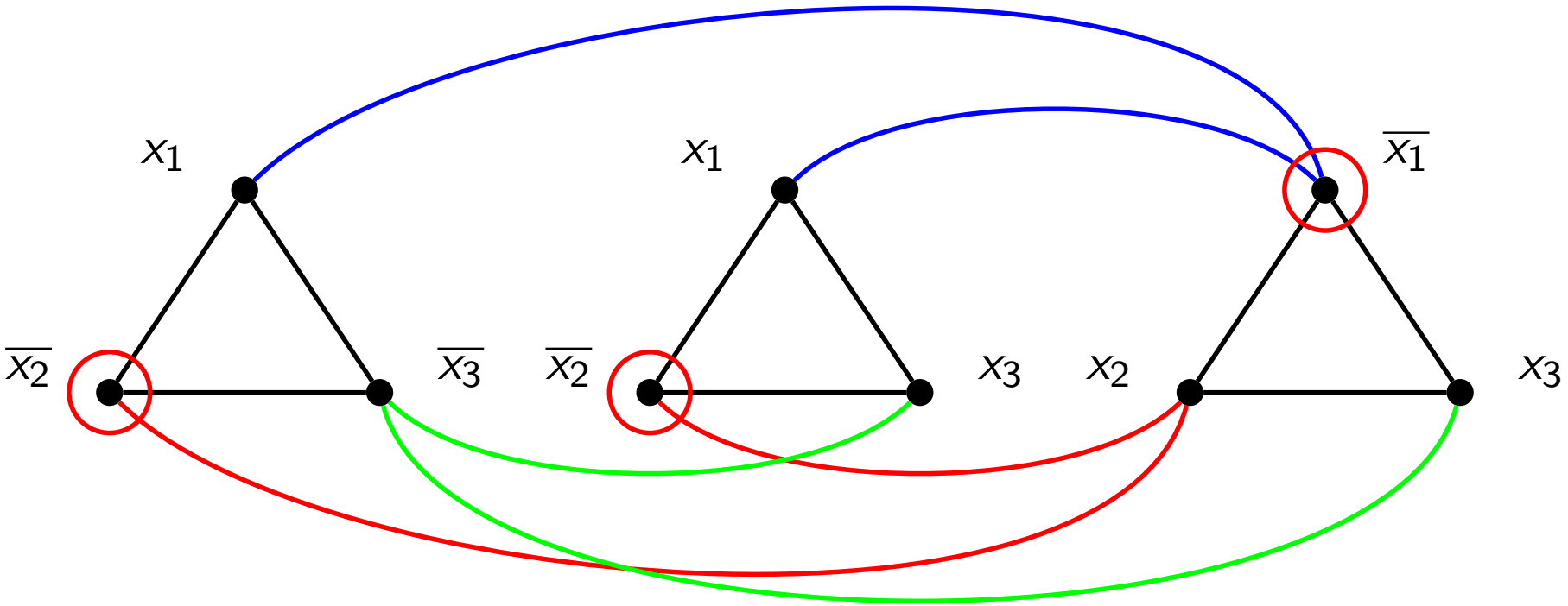
$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



Le problème Ensemble Stable est NP-complet

($\Leftarrow$) S'il existe un ensemble stable de taille $k = 3$ ($k =$ le nombre de clauses de ϕ), alors ϕ est satisfaisable.

$$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$



Attention aux valeurs de k

Théorème

Le problème Ensemble Stable *peut être résolu en temps polynomial si* $k = \mathcal{O}(1)$.

Attention aux valeurs de k

Théorème

Le problème Ensemble Stable *peut être résolu en temps polynomial si* $k = \mathcal{O}(1)$.

Démonstration.

On considère tous les sous-ensembles de k sommets dans G :

$$\mathcal{O}(\binom{n}{k} k^2) = \mathcal{O}(n^k k^2) \qquad \text{Polynomial !}$$

Clique

Clique

Instance : Un graphe G d'ordre n et un entier positif k.

Question : Existe-t-il une clique de taille k dans G, *i.e.*, un sous-ensemble de k sommets connectés deux à deux ?

- Problème de maximisation.
- Problème très étudié.
- Compatibilités, . . .

Clique

Théorème

Le problème CLIQUE *est* NP*-complet.*

Clique

Théorème

Le problème Clique *est* NP*-complet.*

Démonstration.

- Ensemble Stable $\in$ **NP**
- Ensemble Stable $\leq_{\mathbf{P}}$ Clique

Clique

Théorème

Le problème Clique *est* NP*-complet.*

Démonstration.

- Il existe un vérificateur en temps polynomial pour le problème Clique, et par conséquent

$$\textsc{Clique} \in \mathbf{NP}$$

□

Clique

Théorème

Le problème Clique *est* NP*-complet.*

Démonstration.

- Réduction polynomiale depuis le problème Ensemble Stable.
- Il existe un ensemble stable de taille k dans G si et seulement si il existe une clique de taille k dans $\overline{G}$, *i.e.*, le complémentaire de G.
- et donc Ensemble Stable $\leq_{\mathbf{P}}$ Clique.

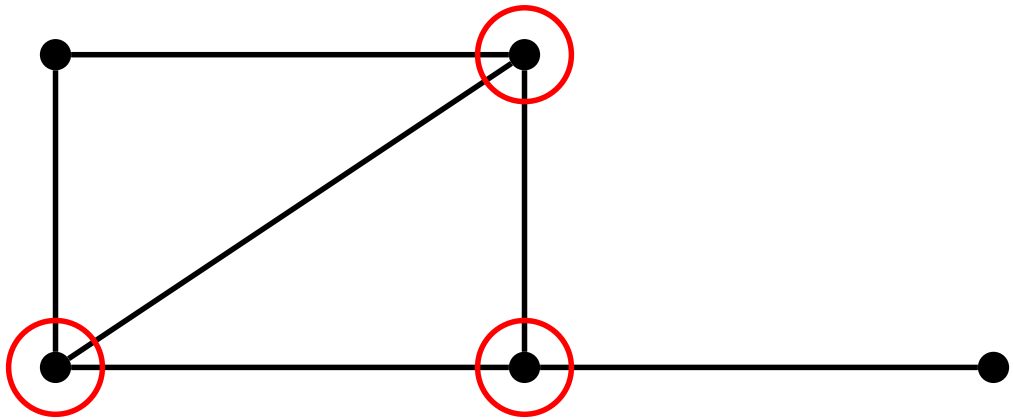
□

Quelques problèmes classiques sur les graphes

Couverture par les Sommets

Instance : Un graphe G d'ordre n et un entier positif k.

Question : Existe-t-il une couverture par les sommets de taille k dans G, *i.e.*, un sous-ensemble de k sommets tel que chaque arête de G est incidente à l'un d'entre eux ?

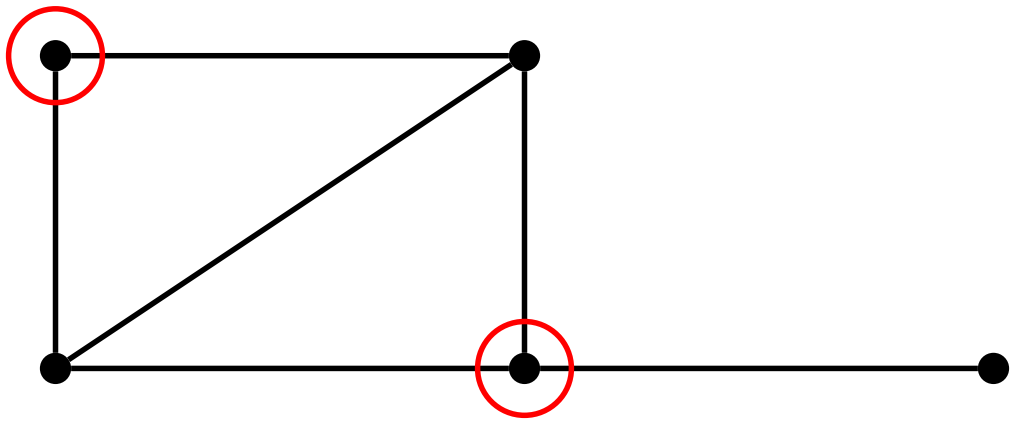


Quelques problèmes classiques sur les graphes

Ensemble Dominant

Instance : Un graphe G d'ordre n et un entier positif k.

Question : Existe-t-il un ensemble dominant de taille k dans G, *i.e.*, un sous-ensemble de k sommets tel que chaque sommet de G appartient ou est adjacent à l'un d'entre eux ?

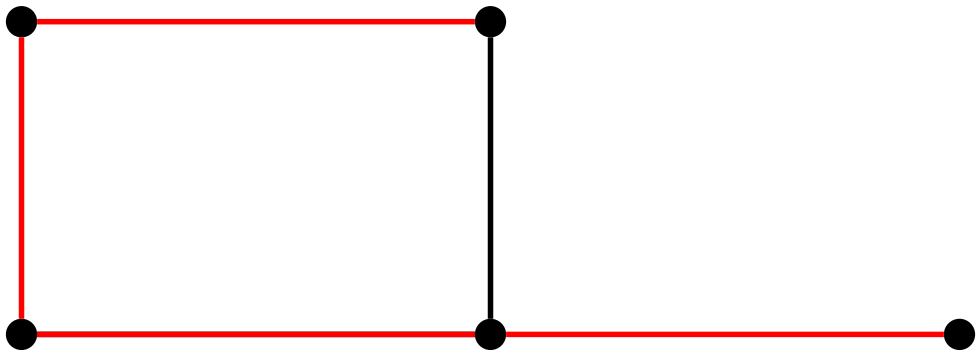


Quelques problèmes classiques sur les graphes

Chemin Hamiltonien

Instance : Un graphe G.

Question : Existe-t-il un chemin Hamiltonien dans G, *i.e.*, un chemin qui passe une et une seule fois par tous les sommets de G ?



Un problème de mots

Exact-Occurrence

Instance : Un ensemble S de N mots, chacun de longueur n, sur un alphabet Σ, et deux entiers positifs k et K.

Question : Existe-t-il un ensemble C contenant au plus K mots, chacun de longueur k, tel que tout mot $s \in S$ contienne une occurrence d'un mot $c \in C$?

- Deux paramètres.
- Couverture d'un ensemble de mots par des mots.

Exact-Occurrence

Exemple ($\Sigma = \{a, c, g, t\}$, $N = 6$,$K = 3$, $n = 8$ et $k = 3$)

$$s_1 = agttgccgt$$
$$s_2 = gttactagc$$
$$s_3 = aagttaacc$$
$$s_4 = cgtgctacg$$
$$s_5 = gcatttaga$$
$$s_6 = gctgctagc$$

Exact-Occurrence

Exemple ($\Sigma = \{a, c, g, t\}$, $N = 6$,$K = 3$, $n = 8$ et $k = 3$)

$$s_1 = agttgccgt$$
$$s_2 = gttactagc$$
$$s_3 = aagttaacc$$
$$s_4 = cgtgctacg$$
$$s_5 = gcatttaga$$
$$s_6 = gctgctagc$$

$$C = \{tgc, tta, aac\}$$

Exact-Occurrence

Exemple ($\Sigma = \{a, c, g, t\}$, $N = 6$,$K = 3$, $n = 8$ et $k = 3$)

$$s_1 = agttgccgt$$
$$s_2 = gttactagc$$
$$s_3 = aagttaacc$$
$$s_4 = cgtgctacg$$
$$s_5 = gcatttaga$$
$$s_6 = gctgctagc$$

$$C = \{tgc, tta\}$$

Exact-Occurrence

Théorème

Le problème Exact-Occurrence *est* NP*-complet.*

Exact-Occurrence

Théorème

Le problème Exact-Occurrence *est* NP*-complet.*

Démonstration.

- Exact-Occurrence $\in$ **NP**
- Couverture par les Sommets $\leq_{\mathbf{P}}$ Exact-Occurrence

Exact-Occurrence

Théorème

Le problème Exact-Occurrence *est* NP*-complet.*

Démonstration.

Soit $(G = (V, E), k')$ une instance arbitraire du problème Couverture par les Sommets.

$$\Sigma = V$$
$$S = \{uv : \{u, v\} \in E \wedge u < v\}$$
$$K = k'$$
$$k = 1$$

Ce que la NP-complétude implique ...

- Il existe au moins une instance du problème pour laquelle il est difficile.
- Sauf si **P** = **NP**, il n'existe pas un algorithme en temps polynomial pour ce problème.
- Ce problème est aussi difficile que le problème SAT, ENSEMBLE STABLE, COUVERTURE PAR LES SOMMETS, CHEMIN HAMILTONIEN, ...
- Si nous proposons un algorithme en temps polynomial pour ce problème, alors **P** = **NP** ... très peu probable !
- Ce n'est pas la fin du monde car ...

Mais la NP-complétude n'implique pas . . .

- que le problème soit difficile pour toutes les instances,
- que le problème soit difficile pour les instances intéressantes,
- qu'un problème de taille raisonnable ne puisse pas être traité,
- qu'il soit difficile de trouver une solution approchée,
- que notre problème soit bien posé.

Plan

Stratégies

Le cas d'école

1. S'il n'est pas possible de trouver un algorithme en temps polynomial pour le problème considéré.
2. Formaliser le problème sous forme d'un problème de décision.
3. Prouver que le problème est NP-complet.
   - Choisir un problème NP-complet.
   - Construire une réduction polynomiale.
   - $(\Rightarrow)$, $(\Leftarrow)$ et conclure.

Choisir un problème NP-complet . . .

- Bien souvent le point difficile.
- Il n'existe pas de règles, mais
  - Préférer un problème simple . . . même si celui-ci semble très éloigné de notre problème.
  - Ne pas toujours se fier à la règle précédente !
  - Considérer un cas particulier de notre problème . . . pour prouver qu'un problème est diffile, il peut être suffisant de prouver cela sur un cas particulier.
- articles . . .
- entrainement . . .

Attention . . .

- Ne pas tout réinventer.
  - Consulter le compendium.
  - Problèmes de graphes.
  - Problèmes de sous-ensembles.
- Cas pathologiques.
  - Éloignement par rapport au problème initial.
  - Affiner la formalisation du problème.
  - Pouvoir d'expression de la formalisation.

Une bonne formalisation du problème est la clef . . . ne pas hésiter à revenir dessus.

Plan

NP-complet . . . et après ?

Que faire si notre problème préféré est NP-complet ?

NP-complet . . . et après ?

Que faire si notre problème préféré est NP-complet ?

- Cas particulier : Quel est notre problème ?
- Approximation : une solution approchée est parfois suffisante.
- Complexité paramétrée : le paramètre considéré est petit par rapport à la taille de l'instance.
- Heuristique : En pratique . . .
- Algorithme randomisé : Trop souvent négligé.
- Algorithme exponentiel : Si rien d'autre n'a fonctionné !
- . . .

Cas particuliers

C'est quoi notre problème déjà ?

Formaliser un problème est une histoire de compromis :

- La formalisation doit être proche de notre problème . . .
- mais pas trop proche pour que nous puissions travailler dessus !

Cas particuliers

C'est quoi notre problème déjà ?

Formaliser un problème est une histoire de compromis :

- La formalisation doit être proche de notre problème . . .
- mais pas trop proche pour que nous puissions travailler dessus !

Remarques

- Si un cas particulier d'un problème est NP-complet, alors le problème *général* est NP-complet.
- Si un problème *général* est NP-complet, alors nous ne disons rien sur la complexité des cas particuliers de ce problème.

Cas particuliers

Problèmes de graphes

- Quelles classes de graphes : graphes bipartis, graphes d'intervalles, graphes triangulés, graphes parfaits, . . .
- Degré maximum d'un sommet.
- Diamètre.
- Densité.
- Largeur d'arborescence.
- . . .

En règle général, quelles sont les caractéristiques des graphes intéressants ?

Cas particuliers

Problèmes de mots

- Taille de l'alphabet.
- Répétitions et autres régularités.
- Taille des mots.
- Nombre de mots.
- Grammaire.
- . . .

En règle général, quelles sont les caractéristiques des mots considérés ?

Approximation

Idée

Un algorithme en temps polynomial qui retourne une solution approchée garantie.

- Problèmes d'optimisation.
- Maximisation *vs.* minimisation.
- Il faut pouvoir garantir la qualité de l'approximation.
- Approximation et non-approximation (en général sous des hypothèses très raisonnables de complexité, *e.g.*, $P \neq NP$).
- Branche très active de la théorie de la complexité.

Ratio d'approximation

- Problème d'optimisation Π.
- Algorithme en temps polynomial A qui retourne pour toute instance $x \in \text{inst}(\Pi)$ une solution $m(x)$.
- Solution optimale $\text{opt}(x)$.

Ratio d'approximation

Le *ratio d'approximation* de l'algorithme A est défini par :

$$r_\Pi(A) = \max_{x \in \text{inst}(\Pi)} \left\{ \frac{\text{opt}(x)}{m(x)}, \frac{m(x)}{\text{opt}(x)} \right\}$$

Ratio d'approximation

Ratio d'approximation

Le *ratio d'approximation* de l'algorithme A est défini par :

$$r_\Pi(A) = \max_{x \in \text{inst}(\Pi)} \left\{ \frac{\text{opt}(x)}{m(x)}, \frac{m(x)}{\text{opt}(x)} \right\}$$

- Définition unifiée pour les problèmes de maximisation et de minimisation :

$$\forall\, x \in \text{inst}(\Pi), \quad r_\Pi(A) \geq 1$$

- $r_\Pi(A) = 1$ si A est un algorithme exact.
- $r_\Pi(A)$ peut être arbitrairement proche de 1, c'est-à-dire $r_\Pi(A) = 1 + \varepsilon$ pour un ε arbitraire.

Approximation

Tous les problèmes difficiles ne sont pas égaux devant l'approximation.

- Schéma d'approximation en temps polynomial.
- Ratio d'approximation constant.
- Ratio d'approximation logarithmique.
- Ratio d'approximation poly-logarithmique.
- Ratio d'approximation polynomial.
- . . .

Approximation

Couverture par les Sommets

Instance : Un graphe G d'ordre n et un entier positif k.

Solution : Une couverture par les sommets de G, *i.e.*, un sous-ensemble de sommets de G tel que chaque arête de G est incidente à l'un d'entre eux ?

Théorème

Il existe un algorithme en temps polynomial pour le problème Couverture par les Sommets *dont le ratio d'approximation est* 2.

Non approximation

Idée

Attaquer un problème par les deux bouts . . .

Non approximation

Un résultat de *non approximation* pour un problème d'optimisation Π est une borne inf. pour tout ratio d'approximation.

- . . . sous des hypothèses de complexité.
- La borne inf. peut utiliser la notation $\mathcal{O}$.

Non approximation

Deux exemples . . . bien différents ! ! !

Théorème

Sauf si $\mathbf{P} = \mathbf{NP}$, *le problème* Couverture par les Sommets *n'est pas approximable par un ratio* 1.1666.

Théorème

Sauf si $\mathbf{NP} = \mathbf{ZPP}$, *le problème* Clique *n'est pas approximable par un ratio* $\mathcal{O}(n^{1-\varepsilon})$ *pour tout* $\varepsilon > 0$.

- ZPP est la classe des problèms pour lesquels il existe un algorithme randomisé qui retourne toujours une solution correcte et dont le temps moyen d'éxecution est polynomial.

La classe FPTAS

Definition

La classe FPTAS comprend tous les problèmes d'optimisation qui sont approximables en un ratio de la forme $1 + \varepsilon$ pour tout $\epsilon > 0$ en temps polynomial en la taille de l'instance et ε^{-1}.

Remarques

- FPTAS : Fully Polynomial-Time Approximation Scheme
- C'est le mieux que l'on puisse espérer.
- Complexité polynomiale pour toute valeur de ε.
- Peu de problèmes sont dans cette classe . . . malheureusement !

La classe PTAS

Definition

La classe PTAS comprend tous les problèmes d'optimisation qui sont approximables en un ratio de la forme $1 + \varepsilon$ pour tout $\epsilon > 0$ en temps polynomial en la taille de l'instance.

Remarques

- PTAS : Polynomial-Time Approximation Scheme
- **FPTAS** $\subseteq$ **PTAS**.
- Intérêt théorique la plupart du temps (complexité en temps des algorithmes).

La classe APX

Definition

La classe APX comprend tous les problèmes d'optimisation qui sont approximables en un ratio constant.

Remarques

- APX : Approximable
- **FPTAS** $\subseteq$ **PTAS** $\subseteq$ **APX**.
- Le problème Couverture par les Sommets est approximable en un ratio 2, et donc il appartient à la classe APX.

Réduction polynomiale et approximation

Utiliser une réduction polynomiale qui *préserve* l'approximation :

- *L*-réduction,
- *PTAS*-réduction,
- *A*-réduction,
- *S*-réduction,
- *AP*-réduction,
- . . .

L-réduction

Soient Π et Π' deux problèmes d'optimisation. Une L-réduction de Π' vers Π est une paire de fonctions (f, g) calculables en temps polynomial telle que

- Pour toute instance $x \in \text{inst}(\Pi')$ de solution optimale $\text{opt}_{\Pi'}(x)$, $f(x)$ est une instance de Π, *i.e.*, $f(x) \in \text{inst}(\Pi)$, de solution optimale $\text{opt}_{\Pi}(f(x))$ telle que

$$\text{opt}_{\Pi}(f(x)) \leq \alpha \, \text{opt}_{\Pi'}(x)$$

pour une constante positive α, et

- Pour toute solution $s \in \text{sol}_{\Pi}(f(x))$, $g(s)$ est une solution de $\text{sol}_{\Pi'}(x)$ telle que

$$|\text{opt}_{\Pi'}(x) - m_{\Pi'}(g(s))| \leq \beta \, |\text{opt}(_{\Pi}(f(x)) - m_{\Pi}(s)|$$

pour une constante positive β.

L-réduction

La *L*-réduction préserve l'approximation.

L-réduction

La L-réduction préserve l'approximation.

Théorème

Soient Π et Π' deux problèmes d'optimisation. S'il existe une L-réduction de Π' vers Π pour deux constantes α et β, et s'il existe un algorithme d'approximation pour Π de ratio $1+\varepsilon$, alors il existe un algorithme d'approximation pour Π' de ratio $1+\alpha\,\beta\,\varepsilon$.

L-réduction

La L-réduction préserve l'approximation.

Théorème

Soient Π et Π' deux problèmes d'optimisation. S'il existe une L-réduction de Π' vers Π pour deux constantes α et β, et s'il existe un algorithme d'approximation pour Π de ratio $1+\varepsilon$, alors il existe un algorithme d'approximation pour Π' de ratio $1+\alpha\,\beta\,\varepsilon$.

Théorème

S'il existe un schéma d'approximation en temps polynomial pour Π, alors il existe un schéma d'approximation en temps polynomial pour Π'.

Complexité paramétrée

Idée

- Isoler un paramètre k dans le problème.
- Remarquer que k est en général "*petit*" comparé à la taille n de l'instance.
- Existe-t-il un algorithme dont la complexité en temps est

$$\mathcal{O}(f(k)\, n^c)$$

où f est est une fonction arbitraire (exponentielle en k si le problème est NP-complet) et c est une constante.

- *fixed-parameter tractability* et *fixed-parameter intractability*.

Complexité paramétrée

PAR-Couverture par les Sommets

Instance : Un graphe G d'ordre n et un entier positif k.

Question : Existe-t-il une couverture par les sommets de taille k dans G, *i.e.*, un sous-ensemble de k sommets tel que chaque arête de G est incidente à l'un d'entre eux ?

Paramètre : k.

Théorème

Il existe un algorithme dont la complexité en temps est $\mathcal{O}(n + k^2\, 2^k)$ *pour le problème* PAR-Couverture par les Sommets.

Complexité paramétrée

PAR-Clique

Instance : Un graphe G d'ordre n et un entier positif k.

Question : Existe-t-il une clique de taille k dans G, *i.e.*, un sous-ensemble de k sommets connectés deux à deux ?

Paramètre : k.

Théorème

Le problème PAR-Clique *est* W[1]*-difficile,* i.e., *sous des hypothèse de complexité très raisonnables il n'existe aucun algorithme dont la complexité en temps est de la forme* $\mathcal{O}(f(k)\, n^c)$.