Complexité avancée - UMIN 345
Théorie de la NP-Complétude (1)

Christophe PAUL

September 21, 2007

Bibliographie

- *A Guide to the Theory of NP-Completenes*, M.R. Garey and D.S. Johnson
- *Computational Complexity*, C. Papadimitriou.
- *Parameterized Complexity*, R. Downey and M. Fellows.
- *Invitation to Fixed-Parameter Algorithms*, R. Niedermeier.

Algorithme [tiré de Wikipedia]

Un **algorithme** est un moyen présenter la résolution par calcul d'un problème. C'est un énoncé dans un langage bien défini d'une suite d'opérations permettant de résoudre par calcul un problème.

Le mot vient du nom du mathématicien **Al Khuwarizmi**, qui, au IXe siècle écrivit le premier ouvrage systématique sur la solution des équations linéaires et quadratiques

L'algorithme le plus célèbre est celui qui se trouve dans le livre 7 des **Eléments d'Euclide**. Il permet de trouver le **plus grand diviseur commun**, ou PGCD, de deux nombres.

Un tri bizarre. . . [Chvàtal]

Données: σ une permutation de $[1, n]$
Résultat: une permutation σ' telle que $\sigma'(1) = 1$
tant que $\sigma(1) \neq 1$ **faire**
    Renverser l'ordre des $\sigma(1)$ premiers éléments
fin

Un tri bizarre. . . [Chvàtal]

Données: σ une permutation de $[1, n]$
Résultat: une permutation σ' telle que $\sigma'(1) = 1$
tant que $\sigma(1) \neq 1$ **faire**
    Renverser l'ordre des $\sigma(1)$ premiers éléments
fin

Exemple

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

Un tri bizarre. . . [Chvàtal]

Données: σ une permutation de $[1, n]$
Résultat: une permutation σ' telle que $\sigma'(1) = 1$
tant que $\sigma(1) \neq 1$ **faire**
    Renverser l'ordre des $\sigma(1)$ premiers éléments
fin

Exemple

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

Un tri bizarre. . . [Chvàtal]

Données: σ une permutation de $[1, n]$
Résultat: une permutation σ' telle que $\sigma'(1) = 1$
tant que $\sigma(1) \neq 1$ **faire**
    Renverser l'ordre des $\sigma(1)$ premiers éléments
fin

Exemple

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

Un tri bizarre. . . [Chvàtal]

Données: σ une permutation de $[1, n]$
Résultat: une permutation σ' telle que $\sigma'(1) = 1$
tant que $\sigma(1) \neq 1$ **faire**
    Renverser l'ordre des $\sigma(1)$ premiers éléments
fin

Exemple

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

Un tri bizarre. . . [Chvàtal]

Données: σ une permutation de $[1, n]$
Résultat: une permutation σ' telle que $\sigma'(1) = 1$
tant que $\sigma(1) \neq 1$ **faire**
    Renverser l'ordre des $\sigma(1)$ premiers éléments
fin

Ce que l'on sait :

- il termine en un nombre fini d'étapes (possiblement $O(2^n)$)

Un tri bizarre. . . [Chvàtal]

Données: σ une permutation de $[1, n]$
Résultat: une permutation σ' telle que $\sigma'(1) = 1$
tant que $\sigma(1) \neq 1$ **faire**
    Renverser l'ordre des $\sigma(1)$ premiers éléments
fin

Ce que l'on sait :

- il termine en un nombre fini d'étapes (possiblement $O(2^n)$)
- en temps **polynomial**, on obtient une 2-**approximation** du nombre d'étapes.

Face à un problème, on se demande :

1. s'il peut se résoudre *efficacement* (i.e. en temps polynomial) ou non

Face à un problème, on se demande :

1. s'il peut se résoudre *efficacement* (i.e. en temps polynomial) ou non
2. si oui, on souhaite **exhiber un algorithme** et analyser sa complexité (e.g. le tri classique nécessite $O(n \log n)$ étapes)

Face à un problème, on se demande :

1. s'il peut se résoudre *efficacement* (i.e. en temps polynomial) ou non
2. si oui, on souhaite **exhiber un algorithme** et analyser sa complexité (e.g. le tri classique nécessite $O(n \log n)$ étapes)
3. si non, on peut essayer montrer qu'une **approximation** est possible

Face à un problème, on se demande :

1. s'il peut se résoudre *efficacement* (i.e. en temps polynomial) ou non
2. si oui, on souhaite **exhiber un algorithme** et analyser sa complexité (e.g. le tri classique nécessite $O(n \log n)$ étapes)
3. si non, on peut essayer montrer qu'une **approximation** est possible
4. ou on peut chercher un **algorithme paramétré**

Face à un problème, on se demande :

1. s'il peut se résoudre *efficacement* (i.e. en temps polynomial) ou non
2. si oui, on souhaite **exhiber un algorithme** et analyser sa complexité (e.g. le tri classique nécessite $O(n \log n)$ étapes)
3. si non, on peut essayer montrer qu'une **approximation** est possible
4. ou on peut chercher un **algorithme paramétré**
5. ou on peut chercher un **algorithme exponentiel** exact

Face à un problème, on se demande :

1. s'il peut se résoudre *efficacement* (i.e. en temps polynomial) ou non
2. si oui, on souhaite **exhiber un algorithme** et analyser sa complexité (e.g. le tri classique nécessite $O(n \log n)$ étapes)
3. si non, on peut essayer montrer qu'une **approximation** est possible
4. ou on peut chercher un **algorithme paramétré**
5. ou on peut chercher un **algorithme exponentiel** exact
6. ...
7. ou on propose une heuristique **seulement si** on peut offrir une validation expérimentale

Moralité :

Un algorithme doit **toujours** offrir une certaine **garantie** sur le résultat!!!

Heuristique [tiré de Wikipedia]

L'heuristique est l'utilisation de règles empiriques pratiques, simples et rapides [...]
Les heuristiques sont, à la différence des algorithmes, tirées de l'expérience ou d'analogies, plutôt que d'une analyse scientifique trop complexe car recensant le maximum d'éléments, et donc difficile, voire impossible à mener et exploiter. L'inconvénient c'est qu'une méthode trop simplifiée peut conduire à des biais cognitifs.

Théorie de la complexité

La **théorie de la complexité** s'intéresse à l'étude formelle de la difficulté des problèmes en informatique.

- **Gödel (1931) :** premier théorème d'incomplétude (toute théorie contient des énoncés non démontrables)

Théorie de la complexité

La **théorie de la complexité** s'intéresse à l'étude formelle de la difficulté des problèmes en informatique.

- **Gödel (1931) :** premier théorème d'incomplétude (toute théorie contient des énoncés non démontrables)
- **Turing, Church (1936) :** réponse négative au problème de la décision (cf. machine de Turing et le problème de l'arrêt)

Théorie de la complexité

La **théorie de la complexité** s'intéresse à l'étude formelle de la difficulté des problèmes en informatique.

- **Gödel (1931) :** premier théorème d'incomplétude (toute théorie contient des énoncés non démontrables)
- **Turing, Church (1936) :** réponse négative au problème de la décision (cf. machine de Turing et le problème de l'arrêt)
- **Edmonds (1965) :** Distinction entre les problèmes P et NP

Théorie de la complexité

La **théorie de la complexité** s'intéresse à l'étude formelle de la difficulté des problèmes en informatique.

- **Gödel (1931) :** premier théorème d'incomplétude (toute théorie contient des énoncés non démontrables)
- **Turing, Church (1936) :** réponse négative au problème de la décision (cf. machine de Turing et le problème de l'arrêt)
- **Edmonds (1965) :** Distinction entre les problèmes P et NP
- **Cook (1971) :** le problème SAT est NP-Complet.

Théorie de la complexité

La **théorie de la complexité** s'intéresse à l'étude formelle de la difficulté des problèmes en informatique.

- **Gödel (1931) :** premier théorème d'incomplétude (toute théorie contient des énoncés non démontrables)
- **Turing, Church (1936) :** réponse négative au problème de la décision (cf. machine de Turing et le problème de l'arrêt)
- **Edmonds (1965) :** Distinction entre les problèmes P et NP
- **Cook (1971) :** le problème SAT est NP-Complet.
- **Karp (1972) :** Liste de 21 nouveaux problèmes NP-Complet.

Théorie de la complexité

La **théorie de la complexité** s'intéresse à l'étude formelle de la difficulté des problèmes en informatique.

- **Gödel (1931) :** premier théorème d'incomplétude (toute théorie contient des énoncés non démontrables)
- **Turing, Church (1936) :** réponse négative au problème de la décision (cf. machine de Turing et le problème de l'arrêt)
- **Edmonds (1965) :** Distinction entre les problèmes P et NP
- **Cook (1971) :** le problème SAT est NP-Complet.
- **Karp (1972) :** Liste de 21 nouveaux problèmes NP-Complet.
- Aujourd'hui la **conjecture** $P = NP$ fait partie des 7 problèmes du millénaire. Le *Clay Mathematical Institute* remettra 1 million USD celui-celle qui le résoudra!

Mesure de la complexité d'un algorithme $\mathcal{A}$

ou mesure des ressources nécessaire à la résolution d'un problème en fonction de la taille des données.

- Temps d'exécution = nombre d'étapes, $t(\mathcal{A})$

Mais aussi

Mesure de la complexité d'un algorithme $\mathcal{A}$

ou mesure des ressources nécessaire à la résolution d'un problème en fonction de la taille des données.

- Temps d'exécution = nombre d'étapes, $t(\mathcal{A})$
- Espace mémoire : taille de l'espace mémoire nécessaire

Mais aussi

Mesure de la complexité d'un algorithme $\mathcal{A}$

ou mesure des ressources nécessaire à la résolution d'un problème en fonction de la taille des données.

- Temps d'exécution = nombre d'étapes, $t(\mathcal{A})$
- Espace mémoire : taille de l'espace mémoire nécessaire

Mais aussi

- Nombre de communications,
- Nombre de processeurs...

Types de problèmes

Peut-on trouver ce qu'on peut prouver ? A. Sebö
(`http://www.larecherche.fr/special/math346/seb.html`)

- **Trouver :** S'il existe une solution au problème, la trouver.

Types de problèmes

Peut-on trouver ce qu'on peut prouver ? A. Sebö
(`http://www.larecherche.fr/special/math346/seb.html`)

- **Trouver :** S'il existe une solution au problème, la trouver.
- **Décider :** Décider s'il existe un solution.

Types de problèmes

Peut-on trouver ce qu'on peut prouver ? A. Sebö
(`http://www.larecherche.fr/special/math346/seb.html`)

- **Trouver :** S'il existe une solution au problème, la trouver.
- **Décider :** Décider s'il existe un solution.
- **Prouver :** Prouver qu'une proposition est une solution au problème

Types de problèmes

Peut-on trouver ce qu'on peut prouver ? A. Sebö
(`http://www.larecherche.fr/special/math346/seb.html`)

- **Trouver :** S'il existe une solution au problème, la trouver.
- **Décider :** Décider s'il existe un solution.
- **Prouver :** Prouver qu'une proposition est une solution au problème

$$\text{Trouver} \Rightarrow \text{Décider} \Rightarrow \text{Prouver}$$

Trouver $\neq$ décider ?

- Soit S un ensemble de n entiers positifs tels que $\Sigma S < 2^n - 1$

Trouver $\neq$ décider ?

- Soit S un ensemble de n entiers positifs tels que $\Sigma S < 2^n - 1$
- Existe-t-il S_1 et S_2 tels que $\Sigma S_1 = \Sigma S_2$?

Trouver $\neq$ décider ?

- Soit S un ensemble de n entiers positifs tels que $\Sigma S < 2^n - 1$
- Existe-t-il S_1 et S_2 tels que $\Sigma S_1 = \Sigma S_2$?

$\longrightarrow$ **Oui ! Toujours**

Trouver $\neq$ décider ?

- Soit S un ensemble de n entiers positifs tels que $\Sigma S < 2^n - 1$
- Existe-t-il S_1 et S_2 tels que $\Sigma S_1 = \Sigma S_2$?

$\rightarrow$ **Oui ! Toujours**

Trouver $\neq$ décider ?

- Soit S un ensemble de n entiers positifs tels que $\Sigma S < 2^n - 1$

Trouver $\neq$ décider ?

- Soit S un ensemble de n entiers positifs tels que $\Sigma S < 2^n - 1$
- Existe-t-il S_1 et S_2 tels que $\Sigma S_1 = \Sigma S_2$?

$\rightarrow$ **Oui ! Toujours**

Trouver $\neq$ décider ?

- Soit S un ensemble de n entiers positifs tels que $\Sigma S < 2^n - 1$
- Trouver S_1 et S_2 tels que $\Sigma S_1 = \Sigma S_2$?

Trouver $\neq$ décider ?

- Soit S un ensemble de n entiers positifs tels que $\Sigma S < 2^n - 1$
- Existe-t-il S_1 et S_2 tels que $\Sigma S_1 = \Sigma S_2$?

$\rightarrow$ **Oui ! Toujours**

Trouver $\neq$ décider ?

- Soit S un ensemble de n entiers positifs tels que $\Sigma S < 2^n - 1$
- Trouver S_1 et S_2 tels que $\Sigma S_1 = \Sigma S_2$?

On ne sait pas faire...

Enoncé des problèmes

- Problème d'optimisation:
  - Donnée : Un graphe $G = (V, E)$
  - Question : Quelle est la taille de la plus grande clique ?

Enoncé des problèmes

- Problème d'optimisation:
  - Donnée : Un graphe $G = (V, E)$
  - Question : Quelle est la taille de la plus grande clique ?
- Problème de décision:
  - Donnée : Un graphe $G = (V, E)$, un entier k
  - Question : G admet-il une clique de taille k ?

Enoncé des problèmes

- Problème d'optimisation:
  - Donnée : Un graphe $G = (V, E)$
  - Question : Quelle est la taille de la plus grande clique ?
- Problème de décision:
  - Donnée : Un graphe $G = (V, E)$, un entier k
  - Question : G admet-il une clique de taille k ?
- Certification
  - Donnée : Un graphe $G = (V, E)$, un ensemble C de sommets
  - Question : C est-il une clique ?

Enoncé des problèmes

- Problème d'optimisation:
  - Donnée : Un graphe $G = (V, E)$
  - Question : Quelle est la taille de la plus grande clique ?
- Problème de décision:
  - Donnée : Un graphe $G = (V, E)$, un entier k
  - Question : G admet-il une clique de taille k ?
- Certification
  - Donnée : Un graphe $G = (V, E)$, un ensemble C de sommets
  - Question : C est-il une clique ?

La théorie de la complexité se concentre sur les problèmes de décision!

La classe **P**

La classe **P** contient les problèmes de décision pour lesquels il existe un algorithme déterministe $\mathcal{A}$ tel que $t(\mathcal{A}) \in O(n^c)$ avec n la taille de la donnée et c une constante.

La classe **P**

La classe **P** contient les problèmes de décision pour lesquels il existe un algorithme déterministe $\mathcal{A}$ tel que $t(\mathcal{A}) \in O(n^c)$ avec n la taille de la donnée et c une constante.

Remarque : il faut être capable de prouver la réponse oui ou non (i.e. fournir un **certificat**)

Exemple : 2-couleurs

- **Donnée :** Un graphe $G = (V, E)$
- **Question :** Les sommets de V peuvent-ils être coloriés avec deux couleurs de sorte qu'aucune arête ne soit incidente à deux sommets de même couleur.

Exemple : 2-couleurs

- **Donnée :** Un graphe $G = (V, E)$
- **Question :** Les sommets de V peuvent-ils être coloriés avec deux couleurs de sorte qu'aucune arête ne soit incidente à deux sommets de même couleur.

Certificats

- **positif :** une coloration propre des sommets en deux couleurs

Exemple : 2-couleurs

- **Donnée :** Un graphe $G = (V, E)$
- **Question :** Les sommets de V peuvent-ils être coloriés avec deux couleurs de sorte qu'aucune arête ne soit incidente à deux sommets de même couleur.

Certificats

- **positif :** une coloration propre des sommets en deux couleurs
- **négatif :** un cycle de longueur impair

Exercice

1. Montrer qu'un algorithme faisant appel à un nombre constant d'algorithmes polynomiaux est lui même polynomial.

Exercice

1. Montrer qu'un algorithme faisant appel à un nombre constant d'algorithmes polynomiaux est lui même polynomial.
2. Que dire d'un algorithme faisant appel à un nombre polynomial d'algorithmes polynomiaux.

Exemple : 2-SAT

- **Donnée :** Une conjonction de clauses à 2 littéraux sur n variables

  $$(x_1 \vee \overline{x_2}) \wedge (x_3 \vee \overline{x_1}) \wedge (x_2 \vee \overline{x_4}) \wedge (x_3 \vee x_4) \wedge (x_1 \vee \overline{x_5}) \wedge (x_2 \vee x_5)$$

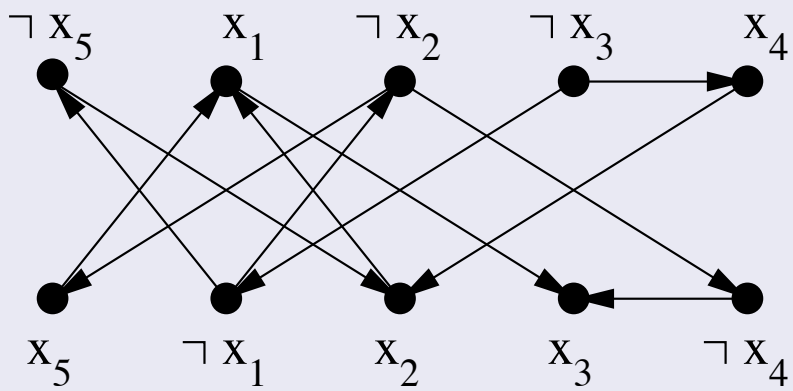
- **Question :** Existe-t-il une affectation des variables satisfaisant la formule ?

Exemple : 2-SAT

- **Donnée :** Une conjonction de clauses à 2 littéraux sur n variables

$$(x_1 \vee \overline{x_2}) \wedge (x_3 \vee \overline{x_1}) \wedge (x_2 \vee \overline{x_4}) \wedge (x_3 \vee x_4) \wedge (x_1 \vee \overline{x_5}) \wedge (x_2 \vee x_5)$$

- **Question :** Existe-t-il une affectation des variables satisfaisant la formule ?

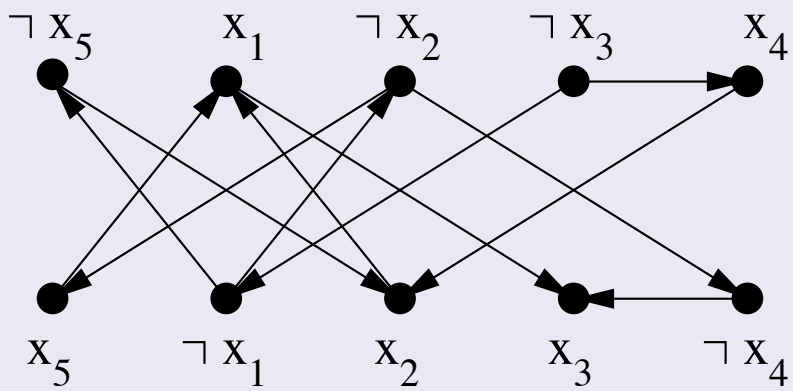


Exemple : 2-SAT

- **Donnée :** Une conjonction de clauses à 2 littéraux sur n variables

  $(x_1 \vee \overline{x_2}) \wedge (x_3 \vee \overline{x_1}) \wedge (x_2 \vee \overline{x_4}) \wedge (x_3 \vee x_4) \wedge (x_1 \vee \overline{x_5}) \wedge (x_2 \vee x_5)$

- **Question :** Existe-t-il une affectation des variables satisfaisant la formule ?



La formule est satisfiable ssi pour tout i, x_i et $\overline{x_i}$ ne sont pas dans la même composante fortement connexe.

Réduction polynomiale

Comment peut-on dire qu'un problème Π_1 est plus facile qu'un problème Π_2 ?

1. **Idée :** Π_1 est plus facile que Π_2 si Π_1 peut être résolu grâce à un algorithme polynomial $\mathcal{A}_2$ pour Π_2.

Réduction polynomiale

Comment peut-on dire qu'un problème Π_1 est plus facile qu'un problème Π_2 ?

1. **Idée :** Π_1 est plus facile que Π_2 si Π_1 peut être résolu grâce à un algorithme polynomial $\mathcal{A}_2$ pour Π_2.
2. **Mais** il faut
   - **transformer** une instance $\mathcal{I}_1$ de Π_1 en une instance $\mathcal{I}_2$ de Π_2

     ...

Réduction polynomiale

Comment peut-on dire qu'un problème Π_1 est plus facile qu'un problème Π_2 ?

1. **Idée :** Π_1 est plus facile que Π_2 si Π_1 peut être résolu grâce à un algorithme polynomial $\mathcal{A}_2$ pour Π_2.
2. **Mais** il faut
   - **transformer** une instance $\mathcal{I}_1$ de Π_1 en une instance $\mathcal{I}_2$ de Π_2 ...en temps polynomial

Réduction polynomiale

Comment peut-on dire qu'un problème Π_1 est plus facile qu'un problème Π_2 ?

1. **Idée :** Π_1 est plus facile que Π_2 si Π_1 peut être résolu grâce à un algorithme polynomial $\mathcal{A}_2$ pour Π_2.
2. **Mais** il faut
   - **transformer** une instance $\mathcal{I}_1$ de Π_1 en une instance $\mathcal{I}_2$ de Π_2 ...en temps polynomial
   - avoir certaine garantie sur la **taille** de $\mathcal{I}_2$ en fonction de celle de $\mathcal{I}_1$ (cf exercice précédent)

Réduction polynomiale

Comment peut-on dire qu'un problème Π_1 est plus facile qu'un problème Π_2 ?

1. **Idée :** Π_1 est plus facile que Π_2 si Π_1 peut être résolu grâce à un algorithme polynomial $\mathcal{A}_2$ pour Π_2.
2. **Mais** il faut
   - **transformer** une instance $\mathcal{I}_1$ de Π_1 en une instance $\mathcal{I}_2$ de Π_2 ...en temps polynomial
   - avoir certaine garantie sur la **taille** de $\mathcal{I}_2$ en fonction de celle de $\mathcal{I}_1$ (cf exercice précédent)
   - assurer une **correspondance** entre les instances positives de Π_1 et celles obtenues par la transformation.

Plus formellement, la **réduction de Karp** est:

$\Pi_1 \leqslant_K \Pi_2$ ssi

- il existe un algorithme polynomial $\mathcal{A}$ transformant $\mathcal{I}_1$ en $\mathcal{I}_2 = \mathcal{A}(\mathcal{I}_2)$;

Plus formellement, la **réduction de Karp** est:

$\Pi_1 \leqslant_K \Pi_2$ ssi

- il existe un algorithme polynomial $\mathcal{A}$ transformant $\mathcal{I}_1$ en $\mathcal{I}_2 = \mathcal{A}(\mathcal{I}_2)$;
- il existe une constance $c > 0$ telle que $|\mathcal{I}_2| \in O(|\mathcal{I}_1|^c)$;

Plus formellement, la **réduction de Karp** est:

$\Pi_1 \leqslant_K \Pi_2$ ssi

- il existe un algorithme polynomial $\mathcal{A}$ transformant $\mathcal{I}_1$ en $\mathcal{I}_2 = \mathcal{A}(\mathcal{I}_2)$;
- il existe une constance $c > 0$ telle que $|\mathcal{I}_2| \in O(|\mathcal{I}_1|^c)$;
- $\Pi_2(\mathcal{I}_2)$ est positive ssi $\Pi_1(\mathcal{I}_1)$ est positive.

Plus formellement, la **réduction de Karp** est:

$\Pi_1 \leqslant_K \Pi_2$ ssi

- il existe un algorithme polynomial $\mathcal{A}$ transformant $\mathcal{I}_1$ en $\mathcal{I}_2 = \mathcal{A}(\mathcal{I}_2)$;
- il existe une constance $c > 0$ telle que $|\mathcal{I}_2| \in O(|\mathcal{I}_1|^c)$;
- $\Pi_2(\mathcal{I}_2)$ est positive ssi $\Pi_1(\mathcal{I}_1)$ est positive.

Lemme

Supposons que $\Pi_1 \leqslant_K \Pi_2$

1. si Π_2 appartient à **P** , alors Π_1 appartient à **P**
2. si Π_1 n'appartient pas à **P** , alors Π_2 n'appartient pas à **P** ,

Exercice

1. Montrer que la relation $\leqslant_K$ est un pré-ordre (reflexive et transitive)
2. Montrer que COUPLAGE BIPARTI $\leqslant_K$ FLOT MAXIMUM.

Exercice

1. Montrer que la relation $\leqslant_K$ est un pré-ordre (reflexive et transitive)
2. Montrer que COUPLAGE BIPARTI $\leqslant_K$ FLOT MAXIMUM.

Exercice

1. INDEPENDENT SET : $\exists$? k sommets deux à deux non-adjacents
2. VERTEX COVER : $\exists$? k sommets tels que toute arête est incidente à au moins un de ces k sommets
3. CLIQUE : $\exists$? k sommets deux à deux adjacents

Montrer que INDEPENDENT SET $\leqslant_K$ VERTEX COVER

Certificat polynomial

Soit une instance $\mathcal{I}$ d'un problème Π, un **certificat** $\mathcal{C}$ est un objet qui prouve l'existance d'une solution.

Le certificat est polynomial s'il existe un algorithme $\mathcal{B}$ polynomial qui étant donée $\mathcal{I}$ et $\mathcal{C}$ retourne vrai ssi $\mathcal{I}$ est une instance positive.

Certificat polynomial

Soit une instance $\mathcal{I}$ d'un problème Π, un **certificat** $\mathcal{C}$ est un objet qui prouve l'existance d'une solution.

Le certificat est polynomial s'il existe un algorithme $\mathcal{B}$ polynomial qui étant donée $\mathcal{I}$ et $\mathcal{C}$ retourne vrai ssi $\mathcal{I}$ est une instance positive.

Remarque

L'algorithme $\mathcal{B}$ pourrait être utilisé pour un algorithme "brut-force" qui essairait tous les objets $\mathcal{C}$ possibles.

Certificat polynomial

Soit une instance $\mathcal{I}$ d'un problème Π, un **certificat** $\mathcal{C}$ est un objet qui prouve l'existance d'une solution.

Le certificat est polynomial s'il existe un algorithme $\mathcal{B}$ polynomial qui étant donée $\mathcal{I}$ et $\mathcal{C}$ retourne vrai ssi $\mathcal{I}$ est une instance positive.

Remarque

L'algorithme $\mathcal{B}$ pourrait être utilisé pour un algorithme "brut-force" qui essairait tous les objets $\mathcal{C}$ possibles.

Exemples

- Un ensemble de sommets pour les problèmes CLIQUE, STABLE ...
- Une affectation des variables pour le problème SAT

Un exemple moins trivial

Problème de planarité : étant donné un graphe $G = (V, E)$, est-il possible de dessiner G dans le plan sans croisement d'arêtes ?

Un exemple moins trivial

Problème de planarité : étant donné un graphe $G = (V, E)$, est-il possible de dessiner G dans le plan sans croisement d'arêtes ?

Théorème de Kuratowski (1930)
Un graphe G est planaire ssi il ne contient pas de K_5 ni de $K_{3,3}$ comme mineur exclu.

La classe **NP**

La classe **NP** contient les problèmes pour lesquels il existe un certificat polynomial
Un problème de décision Π appartient à la classe NP s'il existe un algorithme non-déterministe $\mathcal{A}$ tel que $t(\mathcal{A}) \in O(n^c)$ avec n la taille de la donnée et c une constante.

Observation

La classe **P** est incluse dans la classe **NP**.

VISIBILITÉ

- **Donnée :** Un graphe $G = (V, E)$
- **Question :** G est-il le graphe de visibilité d'un polygone simple ?

VISIBILITÉ

- **Donnée :** Un graphe $G = (V, E)$
- **Question :** G est-il le graphe de visibilité d'un polygone simple ?

Est-ce que VISIBILITÉ appartient à **NP**?

VISIBILITÉ

- **Donnée :** Un graphe $G = (V, E)$
- **Question :** G est-il le graphe de visibilité d'un polygone simple ?

Est-ce que VISIBILITÉ appartient à **NP**?

Problème: la taille du certificat n'est pas polynomial (coordonnées, angles...)

SAT

- **Donnée :** Une conjonction de clauses sur n variables

$$(x_1 \vee \overline{x_2} \vee \overline{x_4} \vee x_5) \wedge (x_3 \vee \overline{x_2} \vee \overline{x_1}) \wedge (x_2 \vee \overline{x_4}) \wedge (x_3) \wedge (x_1 \vee \overline{x_5})$$

- **Question :** Existe-t-il une affectation des variables satisfaisant la formule ?

SAT

- **Donnée :** Une conjonction de clauses sur n variables

$$(x_1 \vee \overline{x_2} \vee \overline{x_4} \vee x_5) \wedge (x_3 \vee \overline{x_2} \vee \overline{x_1}) \wedge (x_2 \vee \overline{x_4}) \wedge (x_3) \wedge (x_1 \vee \overline{x_5})$$

- **Question :** Existe-t-il une affectation des variables satisfaisant la formule ?

Théorème [Karp, 1971]

Le problème SAT est **NP**-Complet

SAT

- **Donnée :** Une conjonction de clauses sur n variables

$$(x_1 \vee \overline{x_2} \vee \overline{x_4} \vee x_5) \wedge (x_3 \vee \overline{x_2} \vee \overline{x_1}) \wedge (x_2 \vee \overline{x_4}) \wedge (x_3) \wedge (x_1 \vee \overline{x_5})$$

- **Question :** Existe-t-il une affectation des variables satisfaisant la formule ?

Théorème [Karp, 1971]

Le problème SAT est **NP**-Complet

La classe des problèmes **NP**-Complets

Un problème Π est NP-Complet ssi $\mathrm{SAT} \leqslant_K \Pi$.

Retour sur le problème de l'arrêt

1. **Pour chaque** variable x d'une instance de SAT, choisir une valeur de manière indéterministe
2. **Si** les n affectations de variables forment une affectation positive **alors** stop
3. **Sinon** faire une boucle infinie

$$\Rightarrow \mathrm{SAT} \leqslant_K \text{problème de l'arrêt}$$

3-SAT

- **Donnée :** Une conjonction de clauses à 3 littéraux sur n variables, un entier k
- **Question :** Existe-t-il une affectation des variables satisfaisant la formule ?

Exercice: Montrer que SAT $\leqslant_K$ 3-SAT

Pour chaque clause $(a_1 \vee a_2 \vee \cdots \vee a_k)$ ayant $k > 3$:

3-SAT

- **Donnée :** Une conjonction de clauses à 3 littéraux sur n variables, un entier k
- **Question :** Existe-t-il une affectation des variables satisfaisant la formule ?

Exercice: Montrer que SAT $\leqslant_K$ 3-SAT

Pour chaque clause $(a_1 \vee a_2 \vee \cdots \vee a_k)$ ayant $k > 3$:

1. on introduit $k - 3$ nouvelles variables $z_2, \ldots z_{k-2}$

3-SAT

- **Donnée :** Une conjonction de clauses à 3 littéraux sur n variables, un entier k
- **Question :** Existe-t-il une affectation des variables satisfaisant la formule ?

Exercice: Montrer que SAT $\leqslant_K$ 3-SAT

Pour chaque clause $(a_1 \vee a_2 \vee \cdots \vee a_k)$ ayant $k > 3$:

1. on introduit $k-3$ nouvelles variables $z_2, \ldots z_{k-2}$
2. on remplace la clause par:

$$(a_1 \vee a_2 \vee \cdots \vee a_k) \quad \rightarrow \quad \begin{cases} (a_1 \vee a_2 \vee z_2) \\ (\overline{z_2} \vee a_3 \vee z_3) \\ \ldots \\ (\overline{z_{k-2}} \vee a_{k-1} \vee a_k) \end{cases}$$

MAX-2-SAT

- **Donnée :** Une conjonction de clauses à 2 littéraux sur n variables, un entier k
- **Question :** Existe-t-il une affectation des variables satisfaisant au moins k clauses ?

Exercice: Montrer que 3-SAT $\leqslant_K$ MAX-2-SAT

MAX-2-SAT

- **Donnée :** Une conjonction de clauses à 2 littéraux sur n variables, un entier k
- **Question :** Existe-t-il une affectation des variables satisfaisant au moins k clauses ?

Exercice: Montrer que 3-SAT $\leqslant_K$ MAX-2-SAT

1. On ajoute une nouvelle variable d et chaque clause est transformée comme suit:

$$(a \vee b \vee c) \quad \rightarrow \quad \left\{ \begin{array}{llll} (a) & (b) & (c) & (d) \\ (\overline{a} \vee \overline{b}) & (\overline{b} \vee \overline{c}) & (\overline{a} \vee \overline{c}) & \\ (a \vee \overline{d}) & (b \vee \overline{d}) & (c \vee \overline{d}) & \end{array} \right.$$

MAX-2-SAT

- **Donnée :** Une conjonction de clauses à 2 littéraux sur n variables, un entier k
- **Question :** Existe-t-il une affectation des variables satisfaisant au moins k clauses ?

Exercice: Montrer que 3-SAT $\leqslant_K$ MAX-2-SAT

1. On ajoute une nouvelle variable d et chaque clause est transformée comme suit:

$$(a \vee b \vee c) \quad \rightarrow \quad \left\{ \begin{array}{llll} (a) & (b) & (c) & (d) \\ (\overline{a} \vee \overline{b}) & (\overline{b} \vee \overline{c}) & (\overline{a} \vee \overline{c}) & \\ (a \vee \overline{d}) & (b \vee \overline{d}) & (c \vee \overline{d}) & \end{array} \right.$$

2. Si l'instance de SAT possède m clauses, on pause $k = 7m$

Exercice Montrer que 3-SAT $\leqslant_K$ CLIQUE

Exercice Montrer que 3-SAT $\leqslant_K$ CLIQUE

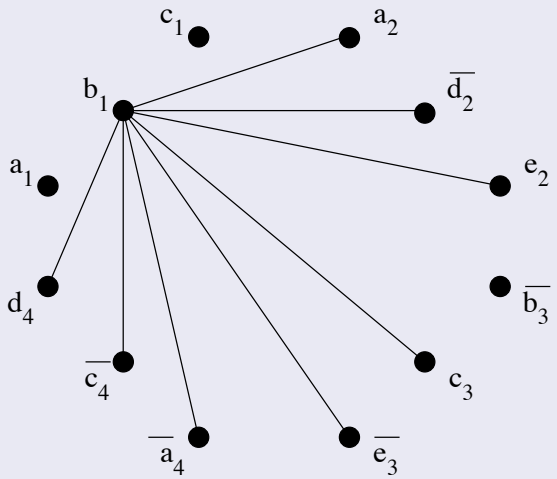
1. à chaque clause $c_i = (x_i^1 \vee x_i^2 \vee x_i^3)$ on associe un stable à 3 sommets x_i^1, x_i^2, x_i^3

Exercice Montrer que 3-SAT $\leqslant_K$ CLIQUE

1. à chaque clause $c_i = (x_i^1 \vee x_i^2 \vee x_i^3)$ on associe un stable à 3 sommets x_i^1, x_i^2, x_i^3
2. on ajoute une arête entre x_i^l et x_j^k ssi $x_i^l \neq \overline{x_j^k}$.

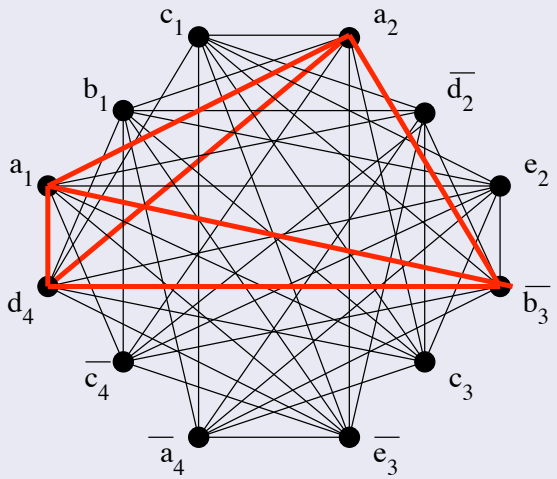
Exercice Montrer que 3-SAT $\leqslant_K$ CLIQUE

1. à chaque clause $c_i = (x_i^1 \vee x_i^2 \vee x_i^3)$ on associe un stable à 3 sommets x_i^1, x_i^2, x_i^3
2. on ajoute une arête entre x_i^l et x_j^k ssi $x_i^l \neq \overline{x_j^k}$.
3. $(a \vee b \vee c) \wedge (a \vee \overline{d} \vee e) \wedge (\overline{b} \vee c \vee \overline{e}) \wedge (\overline{a} \vee \overline{c} \vee d)$



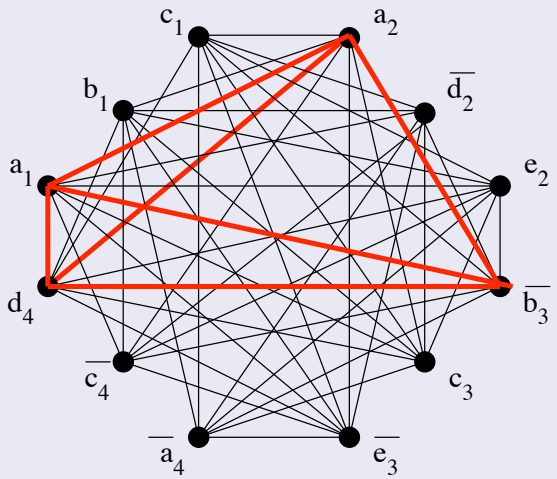
Exercice Montrer que 3-SAT $\leqslant_K$ CLIQUE

1. à chaque clause $c_i = (x_i^1 \vee x_i^2 \vee x_i^3)$ on associe un stable à 3 sommets x_i^1, x_i^2, x_i^3
2. on ajoute une arête entre x_i^l et x_j^k ssi $x_i^l \neq \overline{x_j^k}$.
3. $(a \vee b \vee c) \wedge (a \vee \overline{d} \vee e) \wedge (\overline{b} \vee c \vee \overline{e}) \wedge (\overline{a} \vee \overline{c} \vee d)$



Exercice Montrer que 3-SAT $\leqslant_K$ CLIQUE

1. à chaque clause $c_i = (x_i^1 \vee x_i^2 \vee x_i^3)$ on associe un stable à 3 sommets x_i^1, x_i^2, x_i^3
2. on ajoute une arête entre x_i^l et x_j^k ssi $x_i^l \neq \overline{x_j^k}$.
3. $(a \vee b \vee c) \wedge (a \vee \overline{d} \vee e) \wedge (\overline{b} \vee c \vee \overline{e}) \wedge (\overline{a} \vee \overline{c} \vee d)$



4. La formule m clauses est satisfiable ssi le graphe contient une clique de taille m.

EXERCICE : Montrer que toute instance de K-SAT à m clauses, $m \leqslant 2^k$ est satisfiable

EXERCICE : Montrer que toute instance de K-SAT à m clauses, $m \leqslant 2^k$ est satisfiable

1. La valeur de chaque variable est tirée de manière équiprobable VRAI, FAUX

EXERCICE : Montrer que toute instance de K-SAT à m clauses, $m \leqslant 2^k$ est satisfiable

1. La valeur de chaque variable est tirée de manière équiprobable VRAI, FAUX
2. Soit X la v.a. indiquant le nombre de clauses satisfaites.

EXERCICE : Montrer que toute instance de K-SAT à m clauses, $m \leqslant 2^k$ est satisfiable

1. La valeur de chaque variable est tirée de manière équiprobable VRAI, FAUX
2. Soit X la v.a. indiquant le nombre de clauses satisfaites.
3. $X = \Sigma_{i=1}^{m} X_i$ et $E[X] = \Sigma_{i=1}^{m} E[X_i] = \Sigma_{i=1}^{m} \frac{1}{2^k} = \frac{m}{2^k} < 1$

EXERCICE : Montrer que toute instance de K-SAT à m clauses, $m \leqslant 2^k$ est satisfiable

1. La valeur de chaque variable est tirée de manière équiprobable VRAI, FAUX
2. Soit X la v.a. indiquant le nombre de clauses satisfaites.
3. $X = \Sigma_{i=1}^{m} X_i$ et $E[X] = \Sigma_{i=1}^{m} E[X_i] = \Sigma_{i=1}^{m} \frac{1}{2^k} = \frac{m}{2^k} < 1$
4. Donc $Prob(X < 1) > 0 \Rightarrow$ il existe une affectation des variables qui satisfait la formule.