Cours 6

- Classement des problèmes (2)

Plan

1. Réduction polynomiale
2. Problème $\mathcal{NP}$-difficile, $\mathcal{NP}$-complet
3. Un exemple de réduction pour montrer la $\mathcal{NP}$-complétude
4. Que faire en face d'un problème $\mathcal{NP}$-complet ?

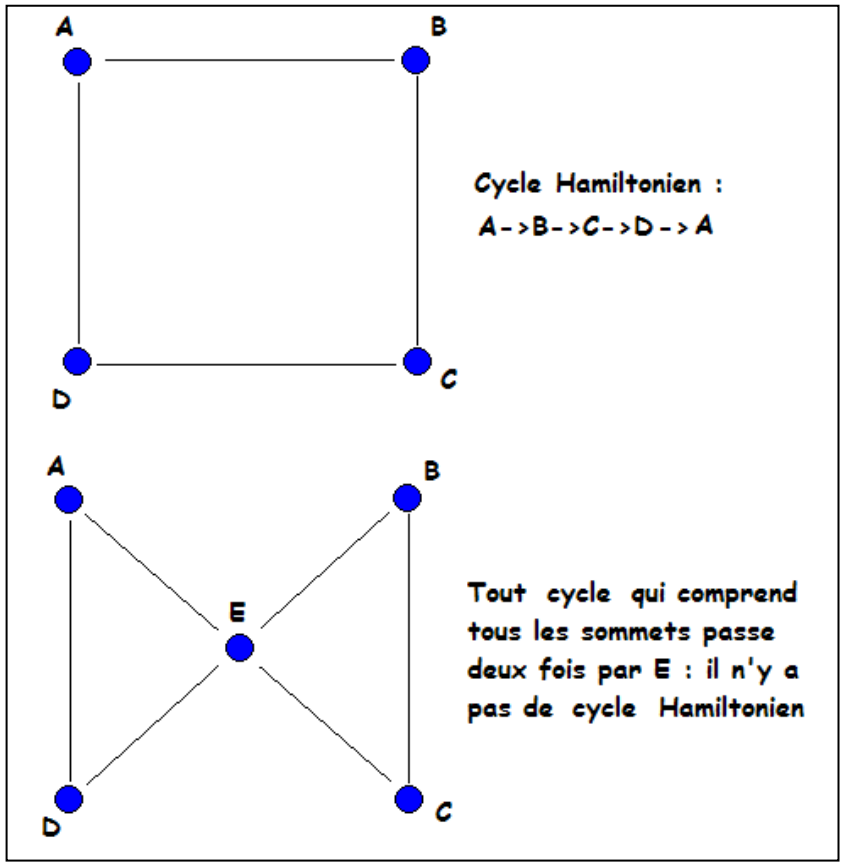
La réduction polynomiale

- Pour relier des problèmes entre eux, on procède par réduction.
- Un problème P1 est réductible à un problème P2 si il existe un algorithme résolvant P1 qui utilise un algorithme résolvant P2.
- Si l'algorithme résolvant P1 est polynomial, considérant les appels à l'algorithme résolvant P2 comme de complexité constante, la réduction est dites polynomiale. On dit que P1 est polynomialement réductible à P2 et l'on note **P1 α P2**.

Exemple de réduction polynomiale

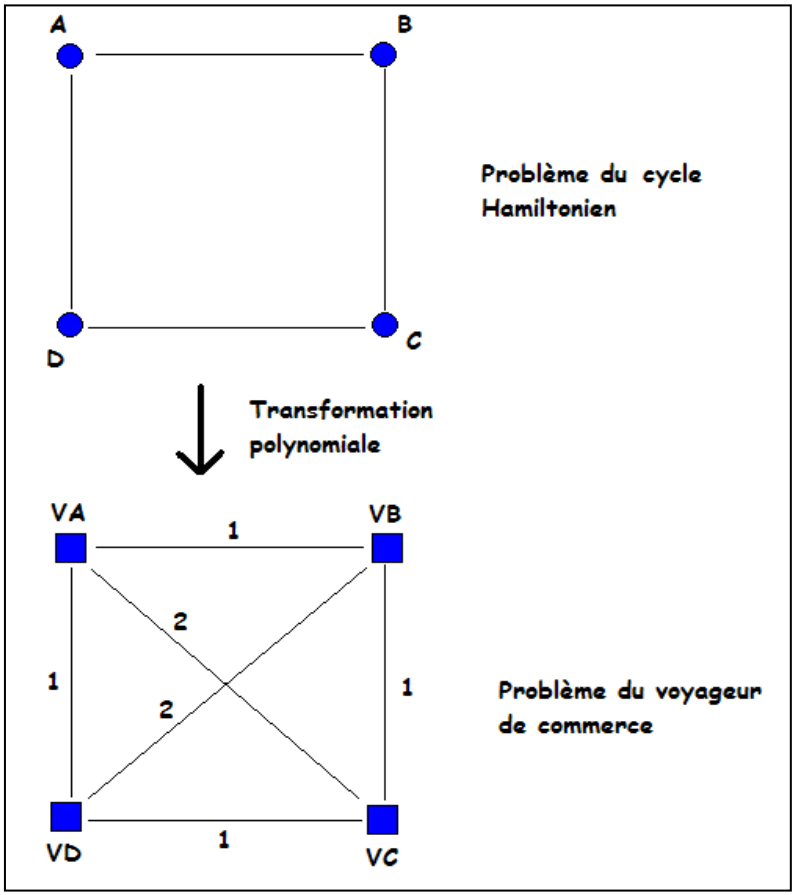
- Problème du cycle hamiltonien : étant donné un graphe, trouver un cycle qui comprend tous les sommets (un cycle est un chemin $(S_1, S_2, \ldots, S_k)$ tel que $S_1 = S_k$ et les autres sommets n'apparaissent qu'une fois dans le chemin).

- Le problème du cycle hamiltonien est polynomialement réductible au problème du voyageur de commerce



Exemple de réduction polynomiale

1. A chaque sommet on associe une ville.
2. Si deux sommets sont reliés par une arête la distance inter-ville correspondante est 1, 2 sinon.
3. Exécuter un algorithme résolvant le problème du voyageur de commerce avec L=n.
4. Si une solution est trouvée, c'est un cycle hamiltonien.



Transitivité de la réduction polynomiale

- On a la propriété suivante :

si

$P1 \alpha P2$ et $P2 \alpha P3$

alors

$P1 \alpha P3$

Transitivité de la réduction polynomiale

Preuve :

A3 : algorithme résolvant P3
A2 : algorithme polynomial résolvant P2 utilisant A3
A1 : algorithme polynomial résolvant P1 utilisant A2

A1 est de complexité polynomiale A2 étant considéré comme temps constant donc le nombre d'appels de A2 est polynomial.

A2 est de complexité polynomiale A3 étant considéré comme temps constant donc le nombre d'appels de A3 est polynomial.

On transforme A1 en un algorithme polynomial A1' appelant A3 un nombre polynomial
de fois (polynôme x polynôme → polynôme) donc la complexité de A3 étant considérée constante A1' est de complexité polynomiale et P1 est polynomialement réductible à P3.

Problèmes $\mathcal{NP}$-difficiles, $\mathcal{NP}$-complets

- Définition : un problème est $\mathcal{NP}$-difficile si tout problème de $\mathcal{NP}$ lui est polynomialement réductible.
- Question : existe-t-il des problèmes $\mathcal{NP}$-difficile ?
- Réponse : oui, c'est l'objet du théorème de Cook.

Existence de problèmes $\mathcal{NP}$-difficile

- Théorème de Cook (1971) : le problème de la satisfabilité d'une expression booléenne est $\mathcal{NP}$-difficile.

- Conséquence : si on trouve un algorithme de complexité polynomiale résolvant SAT, alors tout problème de $\mathcal{NP}$ est dans $\mathcal{P}$, et comme $\mathcal{P} \subset \mathcal{NP}$ alors $\mathcal{P} = \mathcal{NP}$.

Problèmes $\mathcal{NP}$-complet

- Définition : un problème $\mathcal{NP}$-difficile de $\mathcal{NP}$ s'appelle un problème $\mathcal{NP}$-complet.
- SAT est un problème $\mathcal{NP}$-complet.
- Si on connaît un algorithme de complexité polynomiale résolvant un problème $\mathcal{NP}$-complet alors $\mathcal{P} = \mathcal{NP}$.

Exemples de problèmes $\mathcal{NP}$-complet

- Problème du stable : trouver dans un graphe fini un ensemble de m sommets non reliés (m donné).
- Problème du sac à dos
- Problème du cycle Hamiltonien.
- Problème du voyageur de commerce.

Exemples de problèmes $\mathcal{NP}$-complet

- Problème du 3-coloriage : colorier les sommets d'un graphe avec 3 couleurs de sorte que deux sommets adjacents n'aient jamais la même couleur (toujours faisable avec 4 couleurs : théorème des 4 couleurs).

- Problème de la partition : séparer un ensemble d'entiers en deux sous-ensembles dont les sommes partielles sont égales.

Le problème fondamental de la complexité

- $\mathcal{P} = \mathcal{NP}$??
- Tout le monde s'accorde à penser que la réponse est NON.

Exemple de réduction pour la $\mathcal{NP}$-complétude

- Le problème 3-SAT : étant donné une expression booléenne en forme normale conjonctive comportant exactement 3 variables par clause, dire si cette expression est satisfaisable.
- Une telle expression est de la forme $E_1 \wedge \ldots \wedge E_i \wedge \ldots \wedge E_k$ où les E_i sont de la forme $E_i = x \vee y \vee z$
- Proposition : le problème 3-SAT est $\mathcal{NP}$-complet.
- Preuve : par réduction polynomiale (SAT α 3-SAT).

Preuve de la $\mathcal{NP}$-complétude de 3-SAT

- 3-SAT est clairement dans $\mathcal{NP}$ puisque SAT l'est.
- Supposons que l'on montre que SAT α 3-SAT. Si p est un problème quelconque de $\mathcal{NP}$ alors on a :

  p α SAT α 3-SAT et donc p α 3-SAT par transitivité.

  3-SAT est donc $\mathcal{NP}$-difficile et par suite $\mathcal{NP}$-complet.
- Il reste à prouver que SAT α 3-SAT.

Preuve de la $\mathcal{NP}$-complétude de 3-SAT

- On va montrer que l'on peut transformer en temps polynomial une expression booléenne quelconque $F(x_1,\ldots,x_n)$:
  - En une expression booléenne sous forme normale conjonctive $G(x_1,\ldots,x_n,y_1,\ldots y_m)$ comportant exactement 3 variables par clause.
  - Qui lui est équivalente du point de vue de la satisfabilité, c'est-à-dire que si l'on peut attribuer aux variables $x_1,\ldots,x_n$ des valeurs de vérité qui rendent F vraie, on peut attribuer aux variables $y_1,\ldots y_m$ des valeurs tel que G soit vraie pour les valeurs attribuées aux variables $x_1,\ldots,x_n,y_1,\ldots y_m$ et inversement, si l'on peut attribuer des valeurs de vérité aux variables $x_1,\ldots,x_n,y_1,\ldots,y_m$ qui rendent G vraie, F est vraie pour les valeurs attribuées aux variables $x_1,\ldots,x_n$
- On peut montrer que toute expression booléenne peut se mettre en temps polynomial sous forme normale conjonctive (pas fait ici).

Preuve de la $\mathcal{NP}$-complétude de 3-SAT

- On étudie les trois cas suivants pour lesquels E_i ne comporte pas exactement 3 variables :

1. $E_i = x_1$
2. $E_i = x_1 \vee x_2$
3. $E_i = x_1 \vee x_2 \vee \ldots \vee x_j \vee \ldots \vee x_{l-1} \vee x_l$ avec $l \geq 4$

Preuve de la $\mathcal{NP}$-complétude de 3-SAT

- $E_i = x_1$ est remplacée par :

  $E'_i = (x_1 \vee y_1 \vee y_2) \wedge (x_1 \vee y_1 \vee \neg y_2) \wedge (x_1 \vee \neg y_1 \vee y_2) \wedge (x_1 \vee \neg y_1 \vee \neg y_2)$

  où y_1 et y_2 sont deux variables supplémentaires.

- Il est facile de vérifier que E_i et E'_i sont équivalentes : E_i est vraie si x_1 est vrai. Quel que soient les valeurs attribuées à y_1, y_2 , E'_i est alors vraie. Inversement, si E'_i est vraie, comme quel que soient les valeurs attribuées à y_1, y_2 l'une des expressions $y_1 \vee y_2$, $y_1 \vee \neg y_2$, $\neg y_1 \vee y_2$, $\neg y_1 \vee \neg y_2$ est fausse, x_1 est nécessairement vrai et par conséquent E_i est vraie.

Preuve de la $\mathcal{NP}$-complétude de 3-SAT

- $E_i = x_1 \vee x_2$ est remplacée par :

  $E'_i = (x_1 \vee x_2 \vee y) \quad \wedge \quad (x_1 \vee x_2 \vee \neg y)$

  où y est une variable supplémentaire.

- Il est facile de vérifier que E_i et E'_i sont équivalentes : E_i est vraie si x_1 ou x_2 est vrai. Dans ce cas quel que soit la valeur de y, E'_i est vraie. Si E'_i est vraie $x_1 \vee x_2$ est obligatoirement vrai sinon une ou l'autre des clauses $x_1 \vee x_2 \vee y$ et $x_1 \vee x_2 \vee \neg y$ est fausse et donc E'_i est fausse. E_i est donc vraie.

Preuve de la $\mathcal{NP}$-complétude de 3-SAT

- $E_i = x_1 \vee x_2 \vee \ldots \vee x_j \vee \ldots \vee x_{l-1} \vee x_l$ avec $l > 3$ est remplacée par :

$$E'_i = (x_1 \vee x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee y_2) \wedge (\neg y_2 \vee x_4 \vee y_3) \wedge \ldots$$
$$\wedge (\neg y_{j-2} \vee x_j \vee y_{j-1}) \wedge \ldots$$
$$\wedge (\neg y_{l-4} \vee x_{l-2} \vee y_{l-3}) \wedge (\neg y_{l-3} \vee x_{l-1} \vee x_l)$$

où $y_1, y_2, \ldots, y_{l-3}$ sont des variables supplémentaires.

Preuve de la $\mathcal{NP}$-complétude de 3-SAT

- E_i et E'_i sont équivalentes :
- E_i est vraie si l'un au moins des $x_1,\ldots,x_l$ est vrai.
  - Si c' est x_1 ou x_2, en attribuant aux variables $y_1, y_2, \ldots, y_{l-3}$ la valeur faux, E'_i est vraie.
  - Sinon supposons que cela soit x_j avec j>2. En attribuant la valeur vrai aux variables $y_1, y_2, \ldots, y_{j-2}$ et faux aux variables $y_{j-1}, y_j, \ldots, y_{l-3}$, E'_i est vraie.
- Supposons que E'_i soit vraie et que tous les $x_1,\ldots,x_n$ soient faux. E'_i peut alors s'écrire :

$$E'_i = y_1 \wedge (\neg y_1 \vee y_2) \wedge (\neg y_2 \vee y_3) \wedge \ldots \wedge (\neg y_{j-2} \vee y_{j-1}) \wedge \ldots \wedge (\neg y_{l-4} \vee y_{l-3}) \wedge \neg y_{l-3}$$

Alors y_1 est vrai, donc y_2 est vrai, donc y_3 est vrai, …, donc y_{l-3} est vrai, donc $\neg y_{l-3}$ est faux et finalement E'_i est fausse. Donc l'un au moins des $x_1,\ldots,x_n$ est vrai et E_i est vraie.

Preuve de la $\mathcal{NP}$-complétude de 3-SAT

- En utilisant un algorithme A résolvant 3-SAT, on développe alors un algorithme B résolvant SAT :

  1. Transformer la formule $F(x_1,\ldots,x_n)$ sous forme normale conjonctive F'.
  2. Transformer la formule F' $(x_1,\ldots,x_n)$ sous forme normale conjonctive avec des clauses de 3 variables exactement $G(x_1,\ldots,x_n, y_1,\ldots y_m)$.
  3. Appliquer A à G pour déterminer si G est satisfaisable. Si G l'est, F aussi pour les mêmes valeurs des variables $x_1,\ldots,x_n$.

- En considérant A de complexité constante, comme les deux transformations sont de complexité polynomiale, B est de complexité polynomiale et donc SAT est polynomialement réductible à 3-SAT.

Que faire en face d'un problème $\mathcal{NP}$-complet ?

- Utiliser des heuristiques : cela permet de ne pas explorer tous les choix possibles et donc en pratique de rendre les temps de calcul plus raisonnables.
- Accepter une solution approchée au problème : on trouve une solution mais c'est une approximation.

Conclusion

- Les machines ont des limites théoriques : elle ne peuvent résoudre TOUS les problèmes.
- Elles en résolvent quand même pas mal !