



Algorithmique avancée

1 – Complexité et approximabilité

Arpad Rimmel, Marc-Antoine Weisser

CentraleSupélec – M1 MPRI

Saison 2020-21



Organisation du cours

- 1 *NP*-complétude, approximation et inapproximation
- 2 Méthodologie, modélisation, évaluation
- 3 Lien entre espace et temps
- 4 TP 1
- 5 Méta-heuristiques
- 6 TP 2
- 7 TP 3



Première partie I

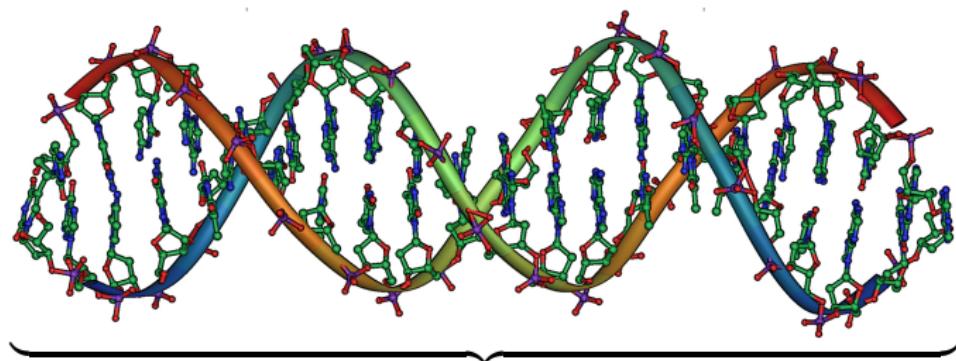
Séquençage ADN



Plan

- 1 Séquençage ADN
- 2 Sir Hamilton
- 3 La machine de Turing
- 4 La classe *NP*
- 5 Réduction Polynomiale
- 6 Retour sur le séquençage

Séquençage ADN



10 millions d'acides aminés

GGCGAATTG GGCCCGACGT CGCATGCTCC TCTAGACTCG AGGAATTCGG
TACCCCCGGGT TCGAAATCGA TAAGCTTGGA TCCGGAGAGC TCCCAACGCG
TTGGATGCAT AGCTTGAGTA TTCTATAGTG TCACCTAAAT AGCTTGGCGT
AATCATGGTC ATAGCTGTT CCTGTGTGAA ATTGTTATCC GCTCACAAATT
CCACACAACA TACGAGCCGG AAGCATAAAG TGTAAGCCT GGGGTGCCATA
ATGAGTGAGC TAACTCACAT TAATTGCGTT GCGCTCACTG CCCGCTTCC
AGTCGGGAAA CCTGTCGTGC CAGCTGCATT AATGAATCGG



Séquençage ADN

Fragments d'ADN

- Impossible de lire un brin ADN sans le fragmenter.
- Quelques centaines de bases par fragment.

GGCGAATTG GG [...] CGT CGCATG TTGGATGCAT
AGCTTGAGTA TC [...] AAT AGCTTGGCGT AATCATGGTC
ATAGCTGTT CCTGTGTGAA ATTGTTAT CCACACAACA
TACGAGCCGG AAGCATAAAAG TGTAAA ...



Séquençage ADN

Fragments d'ADN

- Impossible de lire un brin ADN sans le fragmenter.
- Quelques centaines de bases par fragment.

Problème

Comment reconstituer la séquence entière ?



Séquençage ADN

GGGCGAATTG [...] CGCCGT
TTGGATGCAT [...] TCGTGTGAAT AGCTTGGCGT
AATCATGGTC [...] CCTGTGTGAA ATTGTTAT
TATCACAAACA [...] AAGCATAAAG TGTAAA
CGTGTGTGAA [...] GCTCACAAATT
CGTCACAAACA [...] AAGCATAAAG TGTAAAGCCT GGGGTGCCTA
CTAAGTGAGC [...] TAATTGCGTT GCGCTCACTG CCCGCTTCC

...



Séquençage ADN

GGGCGAATTG [. . .] CGCCGT
TTGGATGCAT [. . .] TCGTGTGAAT AGCTTGGCGT
AATCATGGTC [. . .] CCTGTGTGAA ATTGTTAT
TATCACAAACA [. . .] AAGCATAAAG TGTAAA
CGTGTGTGAA [. . .] GCTCACAAATT
CGTCACAACA [. . .] AAGCATAAAG TGTAAAGCCT GGGGTGCCTA
CTAAGTGAGC [. . .] TAATTGCGTT GCGCTCACTG CCCGCTTCC

...



Séquençage ADN

GGGCGAATTG [. . .] CGCCGT
TTGGATGCAT [. . .] TCGTGTGAAT AGCTTGGCGT
ATTCATGGTC [. . .] CCTGTGTGAA ATTGTTAT
TATCACACAACA [. . .] AAGCATAAAAG TGAAA
CGTGTGTGAA [. . .] GCTCACACATT
CGTCACAACA [. . .] AAGCATAAAAG TGTAAAGCCT GGGGTGCCTA
CTAAGTGAGC [. . .] TAATTGCAGTT GCGCTCACTG CCCGCTTCC

...



Séquençage ADN

Assemblage des fragments

Concaténer, **dans le bon ordre**, les différents fragments d'ADN afin de reconstituer le brin ADN complet.

Contraintes

- La terminaison de chaque fragment doit être compatible avec le commencement du suivant.
- Chaque fragment doit apparaître une et une seule fois.



Séquençage ADN

Conceptuellement, un problème simple

- En pratique, plus de 1000 fragments à assembler et un très grand nombre de séquences à reconstituer.
- L'assistance informatique est indispensable.

Conséquences

Nécessité de construire un modèle mathématique pour formaliser le problème à résoudre, l'étudier et proposer des algorithmes pour le résoudre.



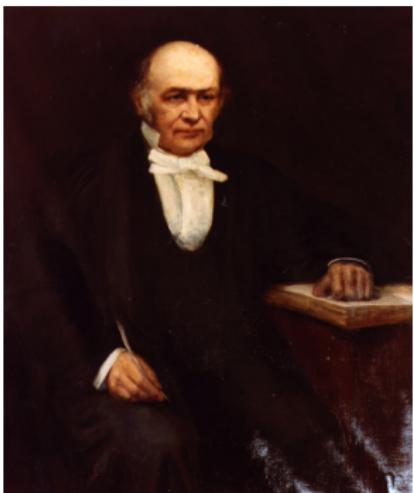
Plan

- 1 Séquençage ADN
- 2 Sir Hamilton
- 3 La machine de Turing
- 4 La classe *NP*
- 5 Réduction Polynomiale
- 6 Retour sur le séquençage

Sir William Rowan Hamilton (1805–1865)

Physicien, astronome et mathématicien Irlandais

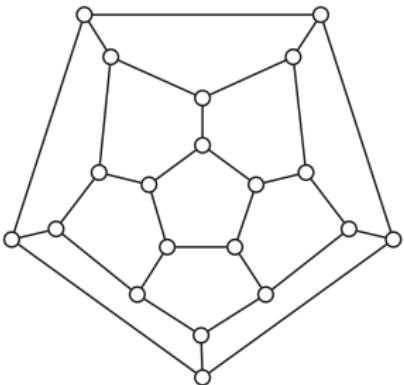
- Mécanique
- Optique
- Algèbre



Le voyage fermé autour du monde

Jeu inventé par Hamilton en 1859

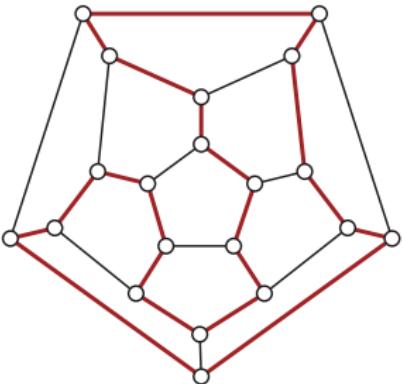
- 20 villes sont réparties sur le globe terrestre (représenté par un dodécaèdre) ;
- Il s'agit de passer une et une seule fois pas chacune des villes en utilisant uniquement les arêtes.



Le voyage fermé autour du monde

Jeu inventé par Hamilton en 1859

- 20 villes sont réparties sur le globe terrestre (représenté par un dodécaèdre) ;
- Il s'agit de passer une et une seule fois pas chacune des villes en utilisant uniquement les arêtes.





Problème du cycle hamiltonien

Modélisation du problème

Données : Un graphe $G = (V, E)$.

Question : Existe-t-il un cycle passant une et une seule fois par chaque sommet du graphe G ?

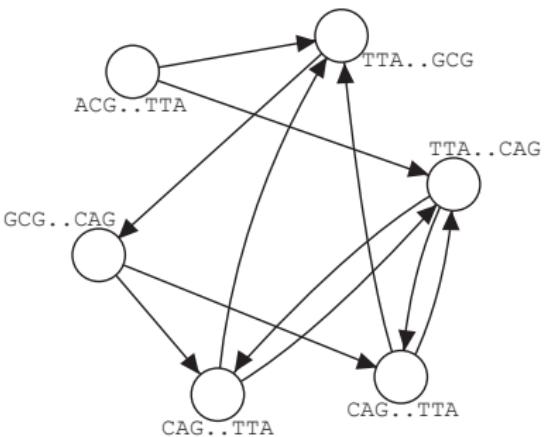
Voir aussi chaîne, chemin et circuit hamiltonien ainsi que de graphe hamiltonien.

Le problème de l'assemblage est directement modélisable avec le problème du chemin hamiltonien.

Modélisation du problème d'assemblage

Problème d'assemblage → Chemin hamiltonien

- Chaque sommet représente un fraguement ADN
- Il existe un arc entre deux sommets u et v si le fraguement ADN u se termine comme v commence.



Résolution du problème

Algorithme récursif simple $A(G, p)$

Entrées:

G , un graphe orienté
 p , un chemin

Algorithme:

- 1: **si** $V(G) = \emptyset$ **alors**
- 2: **renvoie** p
- 3: **pour tout** $v \in V(G)$ **faire**
- 4: **si** $A(G - v, p + v) = \emptyset$ **alors**
- 5: **renvoie** \emptyset
- 6: **sinon**
- 7: **renvoie** $p + v$
- 8: **renvoie** \emptyset



Résolution du problème

Quelle est la qualité de cet algorithme ?

Complexité exponentiel en la taille du graphe : $\mathcal{O}(n!)$

- Est-ce optimal ?
- Y-a-t'il un algorithme polynomial ?

Avant de coder cet algorithme, comment savoir s'il a des chances de fonctionner pour notre utilisation ?



Plan

- 1 Séquençage ADN
- 2 Sir Hamilton
- 3 La machine de Turing
 - Définition
 - Classe *P*
- 4 La classe *NP*
- 5 Réduction Polynomiale
- 6 Retour sur le séquençage



Abstraction d'un ordinateur : la Machine de Turing, 1936

Ressources

- ① fichier de données comme entrée
- ② instructions du programme
- ③ mémoire
- ④ registres et pile d'exécution
- ⑤ fichier de données comme sortie



Abstraction d'un ordinateur : la Machine de Turing, 1936

Ressources

- ① fichier de données comme entrée
- ② instructions du programme
- ③ mémoire
- ④ registres et pile d'exécution
- ⑤ fichier de données comme sortie

Machine de Turing

- La *Machine de Turing* est une **formalisation** de cette structure.

Abstraction d'un ordinateur : la Machine de Turing, 1936

Ressources



Formalisation

- ① fichier de données comme entrée
- ② instructions du programme
- ③ mémoire
- ④ registres et pile d'exécution
- ⑤ fichier de données comme sortie

ruban d'entrée

Machine de Turing

- La *Machine de Turing* est une **formalisation** de cette structure.

Abstraction d'un ordinateur : la Machine de Turing, 1936

Ressources



Formalisation

- ① fichier de données comme entrée
- ② instructions du programme
- ③ mémoire
- ④ registres et pile d'exécution
- ⑤ fichier de données comme sortie

ruban d'entrée
table d'actions

Machine de Turing

- La *Machine de Turing* est une **formalisation** de cette structure.

Abstraction d'un ordinateur : la Machine de Turing, 1936

Ressources



Formalisation

- ① fichier de données comme entrée
- ② instructions du programme
- ③ mémoire
- ④ registres et pile d'exécution
- ⑤ fichier de données comme sortie

ruban d'entrée
table d'actions
ruban de travail

Machine de Turing

- La *Machine de Turing* est une **formalisation** de cette structure.

Abstraction d'un ordinateur : la Machine de Turing, 1936

Ressources



Formalisation

- ① fichier de données comme entrée
- ② instructions du programme
- ③ mémoire
- ④ registres et pile d'exécution
- ⑤ fichier de données comme sortie

ruban d'entrée
table d'actions
ruban de travail
registre d'état

Machine de Turing

- La *Machine de Turing* est une **formalisation** de cette structure.

Abstraction d'un ordinateur : la Machine de Turing, 1936

Ressources



Formalisation

- ① fichier de données comme entrée
- ② instructions du programme
- ③ mémoire
- ④ registres et pile d'exécution
- ⑤ fichier de données comme sortie

- ruban d'entrée
- table d'actions
- ruban de travail
- registre d'état
- ruban de sortie

Machine de Turing

- La *Machine de Turing* est une **formalisation** de cette structure.

Abstraction d'un ordinateur : la Machine de Turing, 1936

Ressources



Formalisation

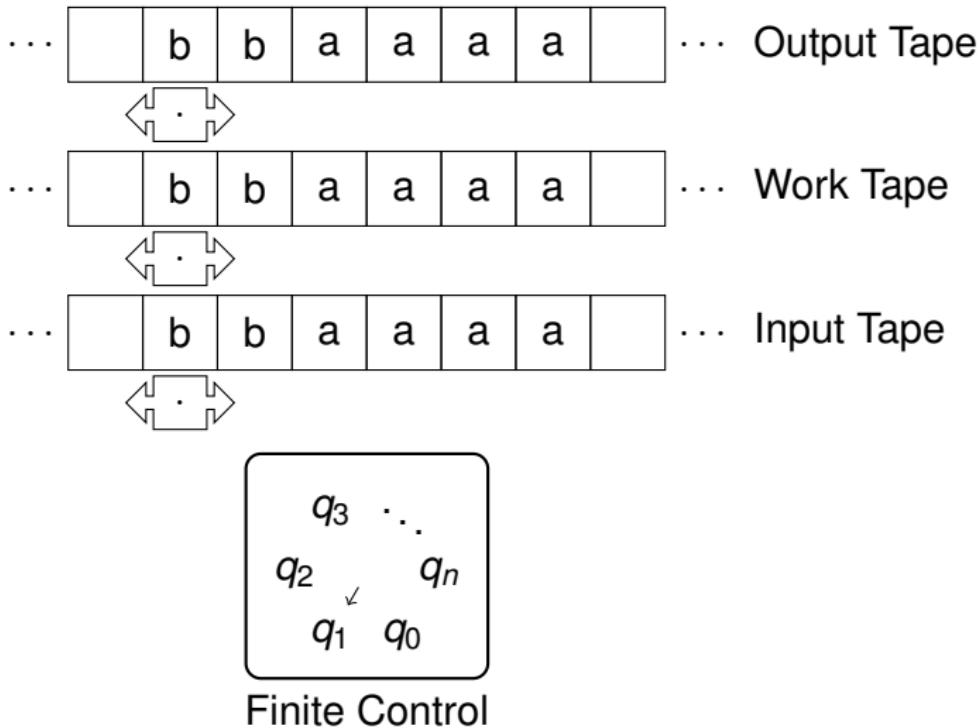
- ① fichier de données comme entrée
- ② instructions du programme
- ③ mémoire
- ④ registres et pile d'exécution
- ⑤ fichier de données comme sortie

- ruban d'entrée
- table d'actions
- ruban de travail
- registre d'état
- ruban de sortie

Machine de Turing

- La *Machine de Turing* est une **formalisation** de cette structure.
- Il existe plusieurs définitions/variantes de la Machine de Turing (nombre de rubans, alphabet...).

Illustration d'une Machine de Turing



Détails des éléments d'une Machine de Turing

Chaque machine possède :

- un **registre** d'état qui mémorise l'état courant de la machine de Turing (le nombre d'états possibles est fini) ;
- trois **rubans** d'entrée, de travail et de sortie divisés en cases pouvant stocker des symboles (alphabet fini) ;
- trois **têtes** pouvant lire ou écrire sur les rubans et se déplacer d'une case vers la gauche ou vers la droite ;

Détails des éléments d'une Machine de Turing

Chaque machine possède :

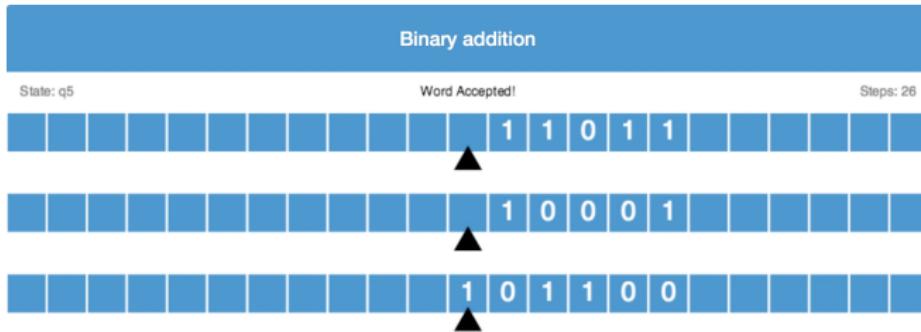
- un **registre** d'état qui mémorise l'état courant de la machine de Turing (le nombre d'états possibles est fini) ;
- trois **rubans** d'entrée, de travail et de sortie divisés en cases pouvant stocker des symboles (alphabet fini) ;
- trois **têtes** pouvant lire ou écrire sur les rubans et se déplacer d'une case vers la gauche ou vers la droite ;
- une table d'**actions** qui en fonction :
 - de l'état courant
 - des symboles lus sur les rubans

indique :

- quel symbole écrire sur chaque ruban
- quel déplacement appliquer aux têtes de lecture (gauche/droite)
- quel est le nouvel état.

Démonstration du fonctionnement d'une Machine de Turing

- Addition de 27 et 17 en binaire : 11011#10001
- Simulateur de MT disponible en ligne :
<https://turingmachinesimulator.com/>



La thèse de Church Turing

- D'autres modèles de calcul existent. Jusqu'à présent, tous ont pu être simulés par une Machine de Turing.



La thèse de Church Turing

- D'autres modèles de calcul existent. Jusqu'à présent, tous ont pu être simulés par une Machine de Turing.

Thèse de Church Turing

Tout système de calcul physique (basé sur du silicium, de l'ADN, des neurones ou toute autre technologie extra-terrestre) peut être simulé par une Machine de Turing.

- Cette thèse n'est pas un théorème, mais elle est largement acceptée par la communauté scientifique.
- Elle implique que ce qui est calculable ne dépend pas du modèle de calcul.



Limites de ce modèle

Tout ne peut pas être calculé par une Machine de Turing.

Problème indécidable

Il existe des fonctions qui ne sont pas calculables par les Machines de Turing.

Exemple, le problème de l'arrêt (Turing, Church, 1936)

Il n'existe pas de MT M qui prend en entrée une MT M' quelconque et détermine si M' s'arrête ou non.

- Savoir si un programme se termine est indécidable.



Temps d'exécution d'une Machine de Turing

Comment mesurer le temps d'exécution d'une MT ?

- Chaque **action** de la machine est une **étape** (lecture, écriture sur un ruban, décalage d'une tête de lecture).

Temps d'exécution d'une Machine de Turing

Comment mesurer le temps d'exécution d'une MT ?

- Chaque **action** de la machine est une **étape** (lecture, écriture sur un ruban, décalage d'une tête de lecture).
- Tout calcul nécessite de lire une entrée dans son ensemble, on mesure donc le temps d'exécution en fonction de la taille de l'entrée.
 - Pour une entrée x on note $|x|$ la taille de x .



Temps d'exécution d'une Machine de Turing

Comment mesurer le temps d'exécution d'une MT ?

- Chaque **action** de la machine est une **étape** (lecture, écriture sur un ruban, décalage d'une tête de lecture).
- Tout calcul nécessite de lire une entrée dans son ensemble, on mesure donc le temps d'exécution en fonction de la taille de l'entrée.
 - Pour une entrée x on note $|x|$ la taille de x .
- On dit qu'une MT calcule une fonction f en temps $T(n)$ si le calcul de toute entrée x tel que $n = |x|$ nécessite au plus $T(|x|)$ étapes.



La classe P

Définition de P

La classe P est l'ensemble des problèmes qui peuvent être résolus par des Machines de Turing en temps **polynomial** $\text{poly}(n)$.



La classe P

Définition de P

La classe P est l'ensemble des problèmes qui peuvent être résolus par des Machines de Turing en temps **polynomial** $\text{poly}(n)$.

Machine simple vs machine complexe

- Toute fonction f calculable par une MT M avec k rubans et un alphabet Γ en un temps $T(n)$ peut être calculé par une MT \tilde{M} avec un seul ruban et un alphabet binaire en un temps $\text{poly}(T(n))$.
- La classe P est donc indépendante du modèle de machine de Turing que l'on considère.



Questions philosophiques

Importance de la classe P

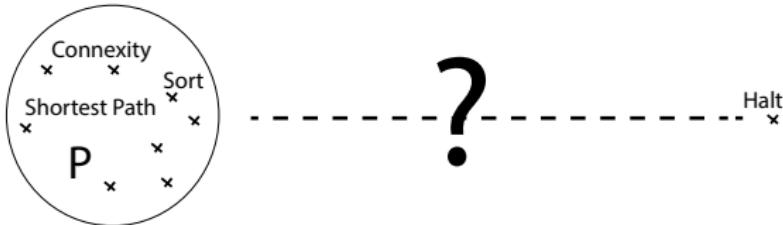
- La classe P capture les problèmes **accessibles**.

Critiques concernant la classe P

- Bien que l'on puisse douter de l'efficacité d'un algorithme avec une complexité de $\mathcal{O}(n^{100})$, en pratique la complexité des algorithmes de résolution des problèmes de P ne dépasse que rarement $\mathcal{O}(n^5)$.
- La complexité au pire est trop stricte.
- D'autres modèles de calcul devraient être envisagés (quantique, avec utilisation d'aléa).

Enjeu

- La classe P est la classe de problèmes pour lesquels il existe des algorithmes efficaces.
- Il existe une classe de problèmes pour lesquels il n'existe pas d'algorithme.
- Qui y a-t-il entre ces deux classes ?
- Où se place le problème du cycle hamiltonien ?





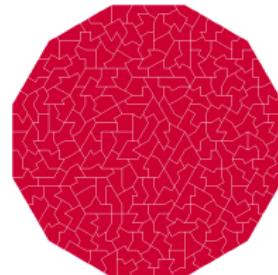
Plan

- 1 Séquençage ADN
- 2 Sir Hamilton
- 3 La machine de Turing
- 4 La classe NP
 - Intuition
 - Définition formelle
 - P et NP
 - Exemples
- 5 Réduction Polynomiale

Eternity

Le puzzle Eternity

- Puzzle de 209 pièces, sorti en juin 1999, associé à une prime de £1'000'000.
- Deux mathématiciens de Cambridge ont remporté le prix en octobre 2000.



Vérifier vs Résoudre

- Dans ce puzzle, il est très simple de vérifier que la solution est correcte alors qu'il est extrêmement difficile de la trouver.
- Cette partie du cours traite de la classe des problèmes NP -complets qui présentent une propriété analogue.

Les problèmes de décision

Définition

Un problème de décision partitionne l'ensemble D d'instances en deux sous-ensembles :

- D^+ d'instances positives (pour lesquelles la réponse est oui) ;
- D^- d'instances négatives (pour lesquelles la réponse est non).

Résoudre un tel problème consiste à déterminer, étant donné $I \in D$, si $I \in D^+$ (ou bien $I \in D^-$).



Classe NP : intuition

Intuition

L'ensemble des **problèmes de décision** pour lesquels :
chaque **instance** positive $I \in D^+$ admet une **solution** S que l'on peut
vérifier par un algorithme en temps polynomial



Classe NP : intuition

Intuition

L'ensemble des **problèmes de décision** pour lesquels :
chaque **instance** positive $I \in D^+$ admet une **solution** S que l'on peut
vérifier par un algorithme en temps polynomial

- L'algorithme de **vérification** prend en entrée une solution et répond **oui** au problème de décision



Classe NP : intuition

Intuition

L'ensemble des **problèmes de décision** pour lesquels :
chaque **instance** positive $I \in D^+$ admet une **solution** S que l'on peut
vérifier par un algorithme en temps polynomial

- L'algorithme de **vérification** prend en entrée une solution et répond **oui** au problème de décision
- L'algorithme de **vérification** est polynomial
- Aucune contrainte sur l'algorithme de **Résolution** : il peut être exponentiel !

(qui réponds oui/non au problème initial, sans connaître la solution)



Classe NP : intuition

Intuition

L'ensemble des **problèmes de décision** pour lesquels :
chaque **instance** positive $I \in D^+$ admet une **solution** S que l'on peut
vérifier par un algorithme en temps polynomial

- L'algorithme de **vérification** prend en entrée une solution et répond **oui** au problème de décision
- L'algorithme de **vérification** est polynomial
- Aucune contrainte sur l'algorithme de **Résolution** : il peut être exponentiel !

(qui réponds oui/non au problème initial, sans connaître la solution)

Pour illustrer !

On peut comprendre et vérifier la preuve d'un théorème, même si c'est bien plus difficile de la trouver soi-même.

▶ skip formal definition



La Classe NP : définition formelle

Définition de la classe NP

Un problème de décision appartient à NP s'il existe une relation binaire polynomiale R et un polynôme p tels que :

$$I \in D^+ \Leftrightarrow \exists x \text{ t.q. } R(I, x) \text{ et } |x| \leq p(|I|)$$

Remarques

- x est un *certificat* prouvant que l'instance I est positive.
- R est calculable, en temps polynomial, par une machine de Turing.
- **On ne considère que les instances positives.**
- NP signifie "Nondeterministic Polynomial Turing Machine", ce nom provient de la définition originelle de la classe NP .



Relations entre **P** et **NP**

$$P \stackrel{?}{\subseteq} NP$$

$$P \stackrel{?}{=} NP$$

$$P \stackrel{?}{\supseteq} NP$$



Relations entre **P** et **NP**

Trivial !

$$P \subseteq NP$$

- L'algorithme de résolution est un algorithme de vérification :
 - il produit un certificat (une solution) et répond oui au problème de décision

Relations entre **P** et **NP**

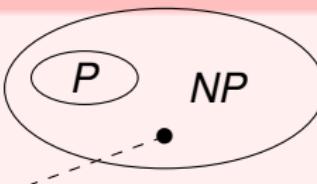
Trivial !

$$P \subseteq NP$$

- L'algorithme de résolution est un algorithme de vérification :
 - il produit un certificat (une solution) et répond oui au problème de décision

Conjecture !

$$P \subsetneq NP$$



- Trouver un problème **NP** et prouver qu'il n'appartient pas à **P**
- **P ≠ NP** est la conjecture la plus célèbre en informatique



Exemple de problème dans NP

1/2

Problème du Stable

Instance :

- $G = (V, E)$ un graphe ;
- $k \in \mathbb{N}$

Question : existe-t-il un **stable** S (independent set) tel que :

- $S \subseteq V$ (sous-graphe) avec $|S| \geq k$
- $\forall u, v \in S. \quad \{u, v\} \notin E$ (le sous-graphe induit ne contient pas d'arête)

Stable est dans NP

On peut vérifier en temps polynomial par rapport à la taille de G , si une solution $S \subseteq V$ (certificat) est un stable de taille supérieure ou égale à k



Exemple de problème dans NP

2/2

Nombre composé

Instance :

- $X \in \mathbb{N}$

Question : X est-il un nombre composé, c'est-à-dire, non premier ?

Le problème du nombre composé est dans NP

- Il existe un certificat de taille polynomiale (n, m) avec $n \leq m < x$.
- On vérifie en temps polynomial (par rapport à la taille de x) que $x = n \times m$.



Liste de problèmes dans NP

Problèmes dans $P \subseteq NP$

Plus court chemin, arbre couvrant de poids minimum, flot maximum

Problèmes dans NP

Stable, sac à dos, sudoku



Liste de problèmes dans NP

Problèmes dans $P \subseteq NP$

Plus court chemin, arbre couvrant de poids minimum, flot maximum

Problèmes dans NP

Stable, sac à dos, sudoku

- En pratique, pour les problèmes NP de la seconde liste, on n'a jamais trouvé d'algorithmes de résolution dans P .
- La plupart des scientifiques pensent que $P \neq NP$



**Que faire des problèmes dans NP
qui ne semblent pas être dans P ?**

Peut-on classer les problèmes par difficulté ?



Plan

- 1 Séquençage ADN
- 2 Sir Hamilton
- 3 La machine de Turing
- 4 La classe *NP*
- 5 Réduction Polynomiale
 - Principe
 - *NP*-complétude
 - SAT
 - $\text{SAT} \leq \text{Stable}$
 - $\text{SAT} \leq \text{D-HAM}$

Problème de la clique

Problème de la clique

Instance :

- $G = (V, E)$ un graphe ;
- $k \in \mathbb{N}$

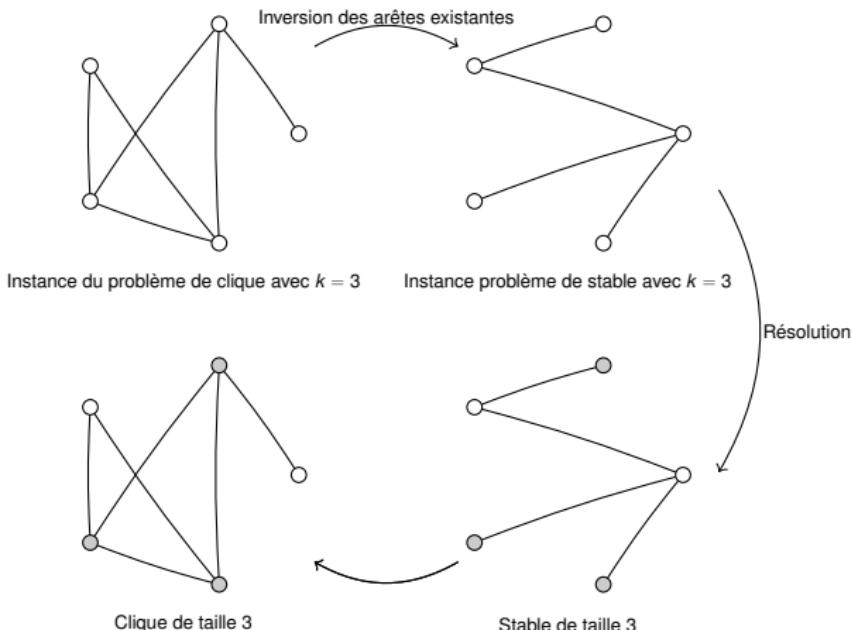
Question : existe-t-il une **clique** S telle que :

- $S \subseteq V$ (sous-graphe) avec $|S| \geq k$
- $\forall u, v \in S. \quad \{u, v\} \in E$ (le sous-graphe induit est **complet**)

Intuitivement, ce problème appartient à la classe *NP*.

Résoudre Clique par Stable

1/2





Résoudre Clique par Stable

2/2

Que montre l'algorithme précédent ?

- La transformation précédente peut-être calculée en temps polynomial donc :
si l'on savait résoudre le problème du stable en temps polynomial, alors on saurait résoudre le problème de la clique également en temps polynomial !



Résoudre Clique par Stable

2/2

Que montre l'algorithme précédent ?

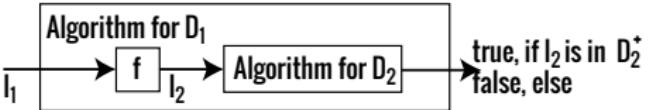
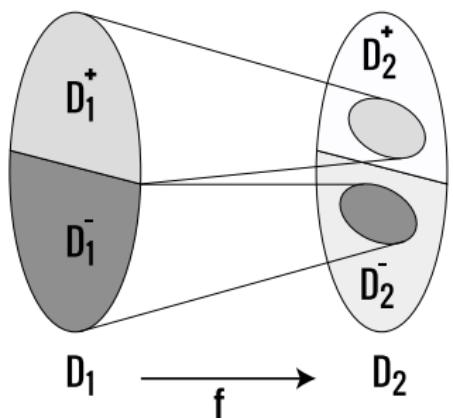
- La transformation précédente peut-être calculée en temps polynomial donc :
si l'on savait résoudre le problème du stable en temps polynomial, alors on saurait résoudre le problème de la clique également en temps polynomial!
- La notion sous-jacente est la **réduction polynomiale**.

Réduction Polynomiale $D_1 \leq D_2$

Soient $D_1 = (D_1^+, D_1^-)$ et $D_2 = (D_2^+, D_2^-)$ deux problèmes de décision.

On dit que D_1 se réduit à D_2 sous une réduction de Karp ($D_1 \leq_K D_2$) s'il existe une fonction $f : D_1 \rightarrow D_2$ telle que :

- $I \in D_1^+ \iff f(I) \in D_2^+$;
- f est calculable en temps polynomial.



NP-complet

NP-difficile

Un problème D est *NP-difficile* si $D' \leq D, \forall D' \in NP$.

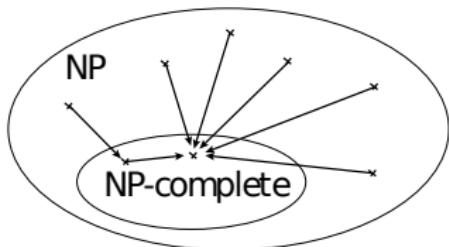
NP-complet

Un problème D est *NP-complet* si D est *NP-difficile* et $D \in NP$.

Théorèmes

- Si $D \leq D'$ et $D' \leq D''$ alors $D \leq D''$.
- Si D est *NP-difficile* et $D \in P$ alors $P = NP$.
- Si D est *NP-complet* alors $D \in P$ ssi $P = NP$.

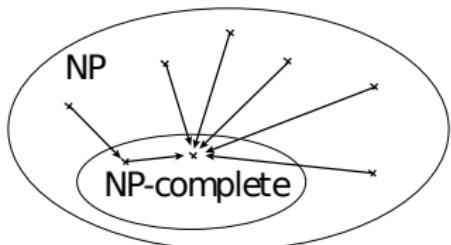
Bilan



Corollaire

On peut réduire tout problème de *NP* dans un problème *NP*-complet.

Bilan



Corollaire

On peut réduire tout problème de *NP* dans un problème *NP*-complet.

Existe-t-il un problème *NP*-complet ?

Satisfaction de formules booléennes

Le problème SAT

Entrée : une formule sous **Forme Normale Conjonctive** (FNC)

- Un ensemble U de variables
- Une collection C de clauses disjonctives de littéraux, où les littéraux sont une variable ou la négation d'une variable.

$$(U_1 \vee U_2 \vee \neg U_3) \wedge (\neg U_1 \vee \neg U_4) \wedge (U_1 \vee U_2 \vee \neg U_5 \vee U_4)$$

Satisfaction de formules booléennes

Le problème SAT

Entrée : une formule sous **Forme Normale Conjonctive** (FNC)

- Un ensemble U de variables
- Une collection C de clauses disjonctives de littéraux, où les littéraux sont une variable ou la négation d'une variable.

Question : Existe-t-il une affectation de valeurs aux variables telle que toutes les clauses soient satisfaites ?

$$(U_1 \vee U_2 \vee \neg U_3) \wedge (\neg U_1 \vee \neg U_4) \wedge (U_1 \vee U_2 \vee \neg U_5 \vee U_4)$$

Satisfaction de formules booléennes

Le problème SAT

Entrée : une formule sous **Forme Normale Conjonctive** (FNC)

- Un ensemble U de variables
- Une collection C de clauses disjonctives de littéraux, où les littéraux sont une variable ou la négation d'une variable.

Question : Existe-t-il une affectation de valeurs aux variables telle que toutes les clauses soient satisfaites ?

$$(U_1 \vee U_2 \vee \neg U_3) \wedge (\neg U_1 \vee \neg U_4) \wedge (U_1 \vee U_2 \vee \neg U_5 \vee U_4)$$

Un premier ensemble de solutions...

Satisfaction de formules booléennes

Le problème SAT

Entrée : une formule sous **Forme Normale Conjonctive** (FNC)

- Un ensemble U de variables
- Une collection C de clauses disjonctives de littéraux, où les littéraux sont une variable ou la négation d'une variable.

Question : Existe-t-il une affectation de valeurs aux variables telle que toutes les clauses soient satisfaites ?

$$(U_1 \vee U_2 \vee \neg U_3) \wedge (\neg U_1 \vee \neg U_4) \wedge (U_1 \vee U_2 \vee \neg U_5 \vee U_4)$$

Un autre ensemble de solutions...



Théorème de Cook-Levin, 1971

Théorème

- SAT est *NP*-complet



Théorème de Cook-Levin, 1971

Théorème

- SAT est NP -complet

Idée de la preuve

- Montrer que SAT est dans NP est trivial.
- On admettra qu'il est NP -difficile :
étant donné un problème $D \in NP$ et une machine de Turing M qui résout D , pour toute instance I de D il est possible de construire en temps polynomial une formule $SAT \varphi(I)$ qui sera vraie ssi M vérifie I .



Existe-t-il d'autres problèmes *NP*-Complets ?



Prouver que Stable est *NP*-complet

Problème du Stable

(rappel)

Instance :

- $G = (V, E)$ un graphe ;
- $k \in \mathbb{N}$

Question : existe-t-il un stable S de taille au moins k ?

Comment montrer que Stable est *NP*-complet ?

Prouver que Stable est *NP*-complet

Problème du Stable

(rappel)

Instance :

- $G = (V, E)$ un graphe ;
- $k \in \mathbb{N}$

Question : existe-t-il un stable S de taille au moins k ?

Comment montrer que Stable est *NP*-complet ?

- Montrer qu'il est dans *NP* (fait)

Prouver que Stable est *NP*-complet

Problème du Stable

(rappel)

Instance :

- $G = (V, E)$ un graphe ;
- $k \in \mathbb{N}$

Question : existe-t-il un stable S de taille au moins k ?

Comment montrer que Stable est *NP*-complet ?

- Montrer qu'il est dans *NP* (fait)
- Montrer qu'il est *NP*-difficile ?

Prouver que Stable est *NP*-complet

Problème du Stable

(rappel)

Instance :

- $G = (V, E)$ un graphe ;
- $k \in \mathbb{N}$

Question : existe-t-il un stable S de taille au moins k ?

Comment montrer que Stable est *NP*-complet ?

- Montrer qu'il est dans *NP* (fait)
- Montrer qu'il est *NP*-difficile ?

*Pour cela, on peut se limiter à réduire un problème *NP*-complet à Stable car la réduction polynomiale est transitive.*



SAT \leq Stable

Considérons les k clauses d'une FNC :

$$(U_1 \vee U_2 \vee \neg U_3) \quad \wedge \quad (\neg U_1 \vee \neg U_4) \quad \wedge \quad (U_1 \vee U_2 \vee \neg U_5 \vee U_4)$$

Méthode

$SAT \leq Stable$

Considérons les k clauses d'une FNC :

$$(U_1 \vee U_2 \vee \neg U_3) \quad \wedge \quad (\neg U_1 \vee \neg U_4) \quad \wedge \quad (U_1 \vee U_2 \vee \neg U_5 \vee U_4)$$



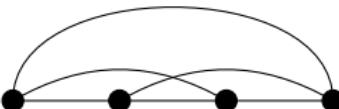
Méthode

- 1 sommet pour chaque occurrence de variable dans une clause

$SAT \leq Stable$

Considérons les k clauses d'une FNC :

$$(U_1 \vee U_2 \vee \neg U_3) \quad \wedge \quad (\neg U_1 \vee \neg U_4) \quad \wedge \quad (U_1 \vee U_2 \vee \neg U_5 \vee U_4)$$



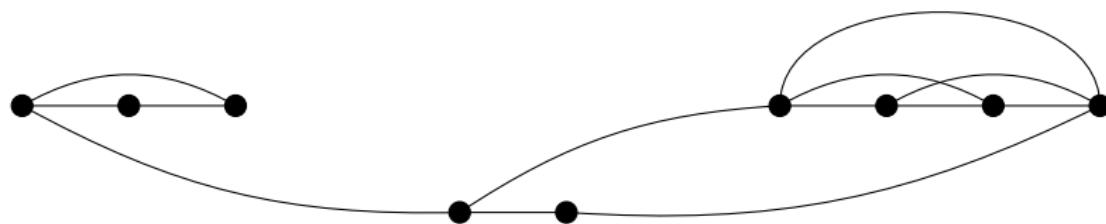
Méthode

- ➊ 1 sommet pour chaque occurrence de variable dans une clause
- ➋ Relier entre eux les sommets d'une même clause

$SAT \leq Stable$

Considérons les k clauses d'une FNC :

$$(U_1 \vee U_2 \vee \neg U_3) \quad \wedge \quad (\neg U_1 \vee \neg U_4) \quad \wedge \quad (U_1 \vee U_2 \vee \neg U_5 \vee U_4)$$



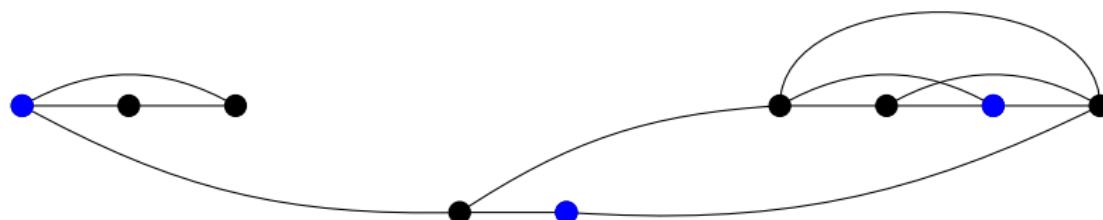
Méthode

- ➊ 1 sommet pour chaque occurrence de variable dans une clause
- ➋ Relier entre eux les sommets d'une même clause
- ➌ Relier les occurrences positives et les occurrences négatives

$SAT \leq Stable$

Considérons les k clauses d'une FNC :

$$(U_1 \vee U_2 \vee \neg U_3) \wedge (\neg U_1 \vee \neg U_4) \wedge (U_1 \vee U_2 \vee \neg U_5 \vee U_4)$$



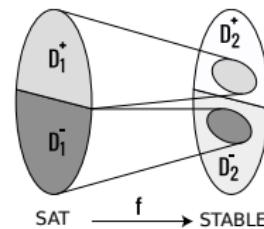
- satisfiable \Rightarrow stable de taille k ou plus
 - chaque clause est satisfaite.
 - on peut construire un stable de taille k en sélectionnant un littéral vrai dans chaque clause.
- stable de taille k \Rightarrow satisfiable
 - chaque sommet du stable correspond à un littéral satisfaisant une clause différente.
 - par construction, le stable ne contient pas deux sommets dont l'un correspondant à une variable et l'autre à sa négation.

$SAT \leq Stable$

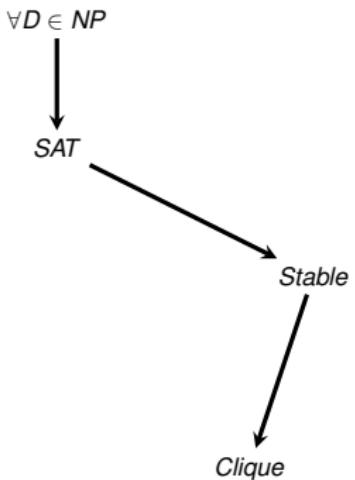
Conclusion

Nous avons défini une réduction f qui pour toute instance I_{SAT} de SAT :

- crée une instance de Stable
 $I_{Stable} = f(I_{SAT})$
 - I_{SAT} est positive $\Rightarrow I_{Stable}$ est positive
 - I_{SAT} est négative $\Rightarrow I_{Stable}$ est négative
 - car I_{Stable} est positive $\Rightarrow I_{SAT}$ est positive
 - f est **polynomiale** en temps
- $SAT \leq Stable$



Le réseau des réductions entre les problèmes *NP*-complets



▶ skip next reduction

Le problème du circuit hamiltonien

D-HAM

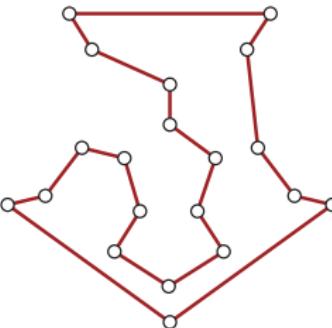
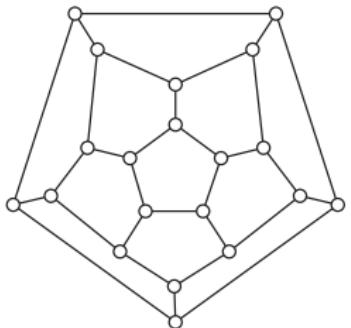
(Directed HAM)

Instance :

- $G = (V, A)$ un graphe orienté

Question : existe-t-il un circuit hamiltonien, c'est-à-dire passant une et une seule fois par tous les sommets du graphe ?

Dans cet exemple, tel que présenté à l'origine par Lord Hamilton, le graphe est non orienté.



D-HAM est *NP*-complet

Premièrement : $D\text{-HAM} \in NP$

Étant donné une instance de D-HAM (un graphe $G = (V, A)$) et un circuit C , il est possible de vérifier en temps polynomial si C est un circuit hamiltonien ou non. Le problème D-HAM appartient donc à la classe *NP*.

D-HAM est *NP*-complet

Premièrement : $D\text{-HAM} \in NP$

Étant donné une instance de D-HAM (un graphe $G = (V, A)$) et un circuit C , il est possible de vérifier en temps polynomial si C est un circuit hamiltonien ou non. Le problème D-HAM appartient donc à la classe *NP*.

Deuxièmement : une réduction polynomiale $SAT \leq D\text{-HAM}$

Comment réduire SAT en D-HAM ?

SAT \leq D-HAM

Soit I une instance de SAT avec les variables x_1, \dots, x_n et les clauses C_1, \dots, C_k .

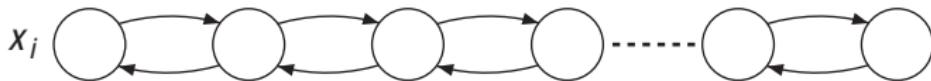
$$(x_1 \vee x_2) \wedge (\overline{x_3} \vee x_4 \vee \overline{x_5}) \wedge \dots \wedge (\overline{x_1})$$

Idée de la réduction

- Créer des structures à base de graphe pour représenter les variables et les clauses.
- Organiser les structures ensemble pour encoder la formule.
- Montrer que la structure admet un circuit hamiltonien ssi la formule est satisfiable.

SAT \leq D-HAM

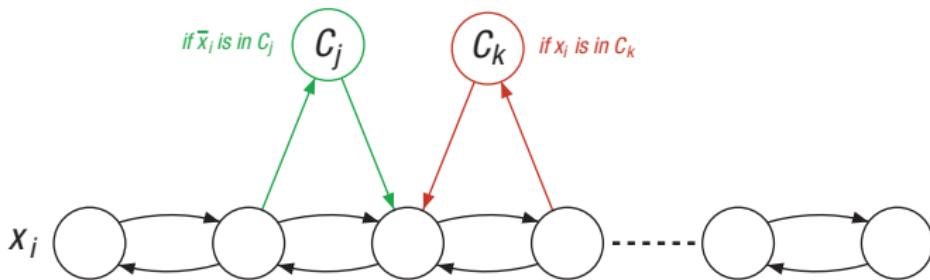
Pour chaque variable x_i , on crée une structure de la forme suivante.



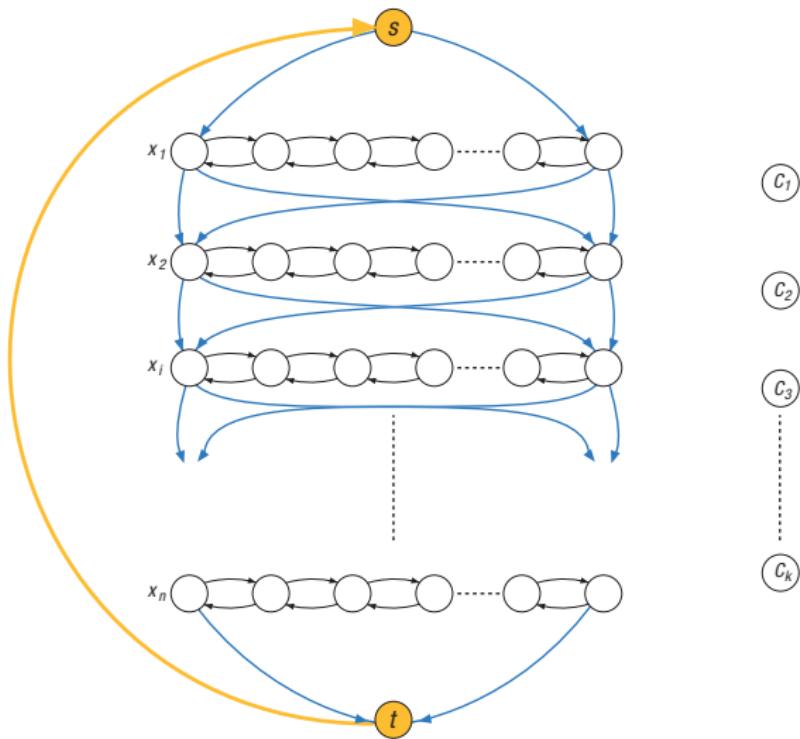
Elle devra être traversée par le circuit hamiltonien. Par convention, si elle est traversée de gauche à droite, on affectera *faux* à la variable correspondante, sinon on affectera *vrai* à la variable.

$SAT \leq$ D-HAM

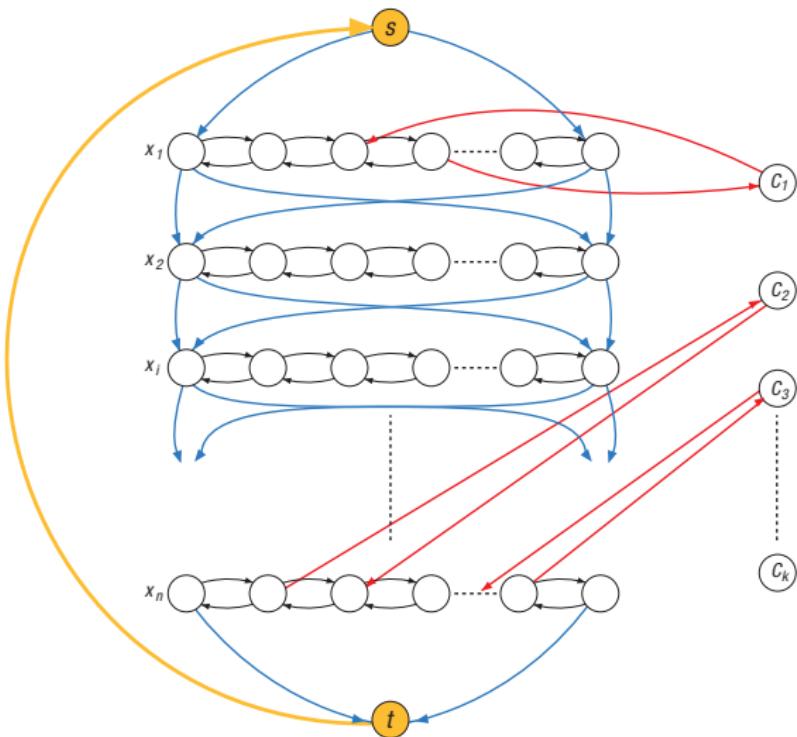
On ajoute un nouveau nœud pour chaque clause et on le relie aux structures existantes.



Pour chaque variables, les structures doivent être suffisamment longues pour placer les clauses (au moins $3+k$ noeuds). Le nombre total de noeuds ($3n+kn$) reste polynomial en fonction de la taille de la formule SAT.

$SAT \leq D\text{-HAM}$ 

$SAT \leq D\text{-HAM}$



$SAT \leq D\text{-HAM}$

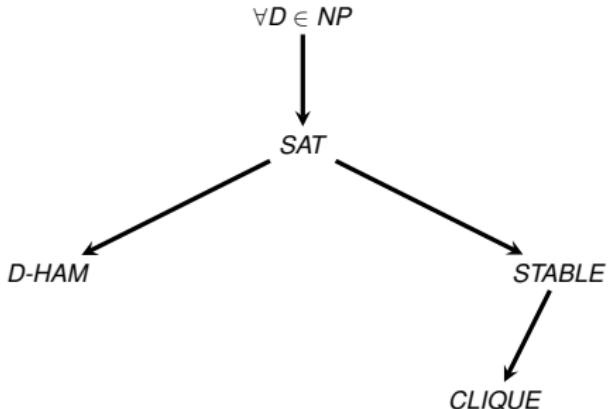
Supposons qu'il existe un circuit hamiltonien

- Ce circuit hamiltonien encode les affectations pour les variables (en fonction de la direction dans laquelle chaque structure est traversée).
- Pour que le circuit soit hamiltonien, il doit visiter toutes les clauses.
- On ne peut visiter les clauses qu'en les satisfaisant (en mettant un de ses termes à *vrai*).
- Donc s'il existe un circuit hamiltonien, il existe une affectation qui satisfait la formule.

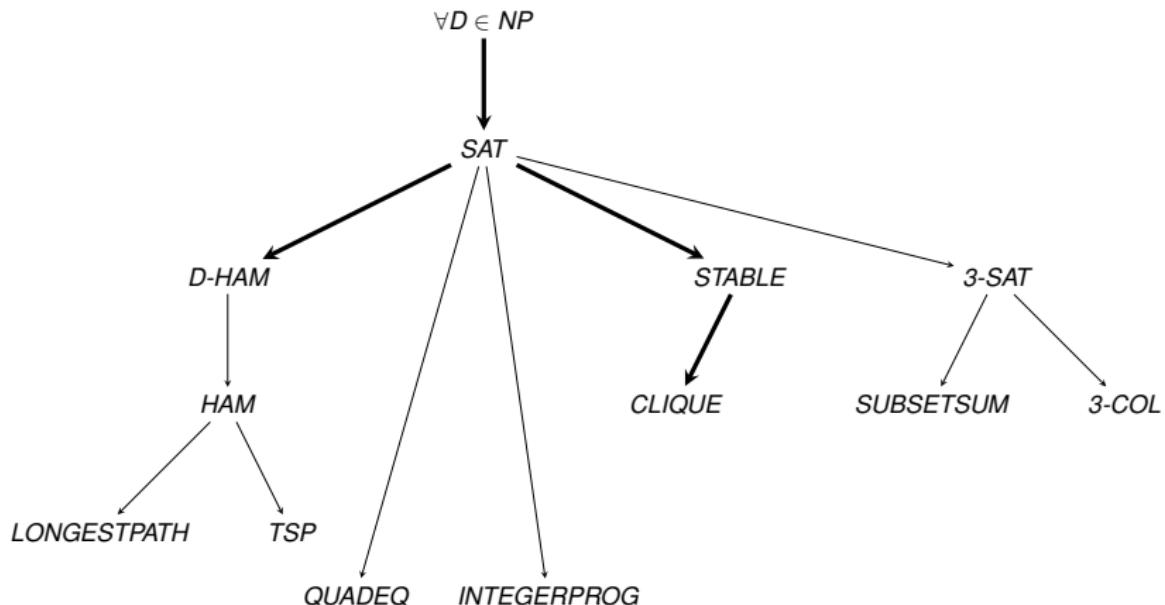
Réiproque

Une affectation décrit un parcours (attention à ne pas repasser par des clauses déjà satisfaites)

Le réseau des réductions entre les problèmes *NP*-complets



Le réseau des réductions entre les problèmes *NP*-complets





Plan

- 1 Séquençage ADN
- 2 Sir Hamilton
- 3 La machine de Turing
- 4 La classe *NP*
- 5 Réduction Polynomiale
- 6 Retour sur le séquençage

Raboutage ADN

Complexité du problème

Ces résultats montrent qu'il n'existe pas d'algorithme qui, pour toute instance, fonctionne en temps polynomial.

Unicité de la solution

- Dans certain cas, il peut exister plusieurs solutions au problème du chemin hamiltonien.
- Comment choisir celle qui le “plus de sens” du point de vue de la biologie ?

Taille du recouvrement

Meilleur solution

Soient deux fragments ADN s_1 et s_2 . Plus le nombre de bases en commun entre la fin de s_1 et le début de s_2 est grand, plus la probabilité que s_2 suive s_1 est grande.

Graphe orienté complet pondéré

Construction d'un graphe complet $G = (V, A, \omega)$ avec $\omega : V \times V \rightarrow \mathbb{Z}$ où $\omega(x, y)$ est l'opposé du nombre de base en commun entre x et y .

Problème du voyageur de commerce

Circuit Hamiltonien pondéré

Dans le graphe pondéré $G = (V, A, \omega)$ le problème est toujours de trouver un chemin hamiltonien, mais dont le poids est minimum (où le poids d'un circuit est la somme du poids de ces arcs).

Problème du Voyageur de Commerce (VC)

Données : un graphe $G = (V, A, \omega)$ complet, $\omega : V \times V \rightarrow \mathbb{Z}$ et $K \in \mathbb{Z}$

Question : Existe-t-il un circuit hamiltonien de poids inférieur à K dans G ?

Complexité du Voyageur de Commerce

VC est-il plus difficile que CIR-HAM ?

- Contrairement à CIR-HAM dans VC on considère des graphes complets (il existe donc toujours un circuit hamiltonien)
- VC est clairement dans la classe *NP*. Étant donné un circuit hamiltonien, on peut tester en temps polynomial si son poids est inférieur à K ou non.
- VC est complet pour cette classe car l'on peut réduire un problème *NP*-Complet (CIR-HAM) au problème VC.
- Les deux problèmes appartiennent donc à la même classe de complexité.

Méthode exacte de résolution du voyageur de commerce

- force brute, complexité en $\mathcal{O}(n!)$, on ne dépasse pas les 20 nœuds ;
- algorithme de branch and bound ~ 50 nœuds ;
- programmation linéaire ~ 200 nœuds ;
- branch and bound adapter au problème spécifique plusieurs milliers de nœuds.



Deuxième partie II

Voyageur de commerce



Plan

7 Voyageur de commerce

8 Digressions nécessaires

9 Approximation

10 Inapproximation

Fraiseuse à commande numérique

Ceci est une fraiseuse

- Permet de percer des trous
- Coûte 50 000 €

Fonctionnement

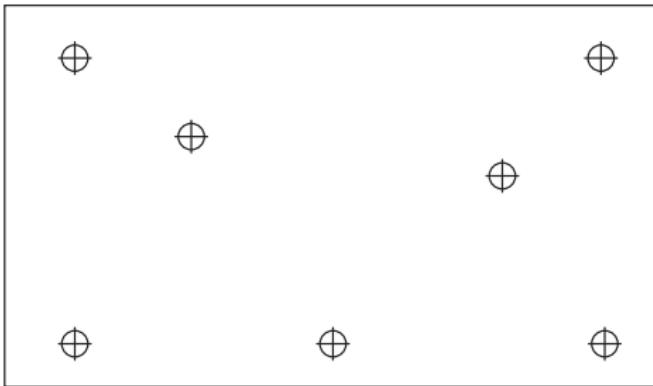
- Prend des plaques
- Perce des trous en suivant un programme établi



Comment rentabiliser la fraiseuse

Rentabilisation de la machine

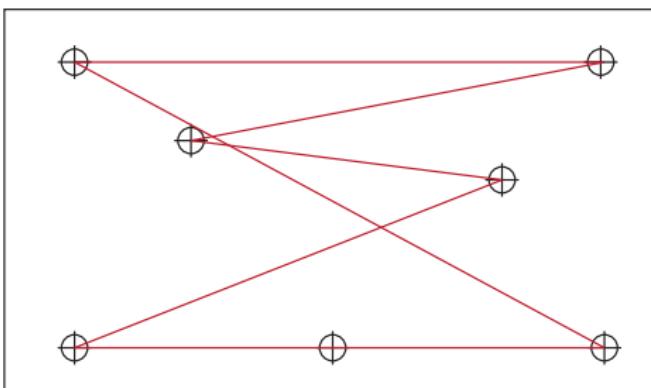
- Percer un maximum de plaques en un minimum de temps
- Le temps nécessaire pour perforez un trou est fixe
- Optimiser le trajet entre les trous afin de minimiser le temps de traitement d'une plaque



Comment rentabiliser la fraiseuse

Rentabilisation de la machine

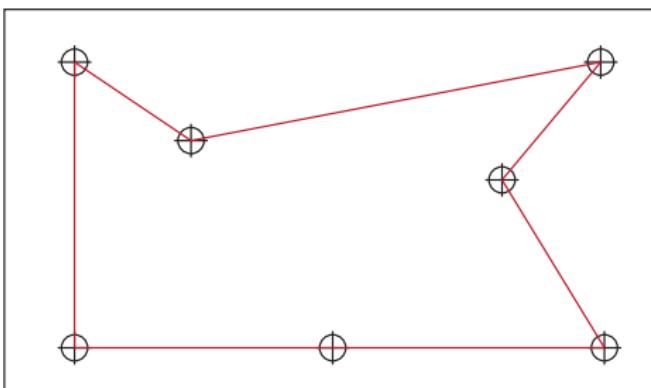
- Percer un maximum de plaques en un minimum de temps
- Le temps nécessaire pour perforez un trou est fixe
- Optimiser le trajet entre les trous afin de minimiser le temps de traitement d'une plaque



Comment rentabiliser la fraiseuse

Rentabilisation de la machine

- Percer un maximum de plaques en un minimum de temps
- Le temps nécessaire pour perforez un trou est fixe
- Optimiser le trajet entre les trous afin de minimiser le temps de traitement d'une plaque





Voyageur de commerce

Problèmes similaires

- Voyageur de commerce
- Tournées de véhicule
- Conception de processeur
- ...

Voyageur de commerce métrique

Contrairement au séquençage ADN, le support n'est pas un graphe mais un plan, les distances respectent l'inégalité triangulaire.



Modélisation

Voyageur de commerce métrique

Données :

- Un graphe non orienté $G = (V, E)$ complet
- Une pondération sur les arêtes $\omega : E \rightarrow \mathbb{N}$ telle que $\forall u, v, w \in V, \omega(u, w) < \omega(u, v) + \omega(v, w)$ (inégalité triangulaire)
- $K \in \mathbb{N}$

Question : Existe-t-il un cycle passant une et une seule fois par chaque sommet du graphe G de poids inférieur à K ?

Le problème de minimisation associé consiste à minimiser la taille du cycle



Remarques

Fonction de pondération ω

Une seule contrainte sur la fonction ω , elle doit respecter l'inégalité triangulaire. Suivant l'application on peut la définir différemment :

- distance euclidienne
- distance de manhattan (déplacements diagonaux interdits)
- distance maximum (déplacements dans les deux dimensions indépendants)

NP-Complétude

- La preuve de *NP*-complétude du problème de l'hamiltonicité d'un graphe fonctionne avec ce problème.



Résolution ?

Deux approches

- Méthodes exactes telles que le branch and bound (non polynomial)
- Heuristiques polynomiales

On s'intéresse aux heuristiques



Plan

7 Voyageur de commerce

8 Digressions nécessaires

9 Approximation

10 Inapproximation



Arbres couvrants de poids minimum

Problème de l'arbre couvrant de poids minimum

Sous forme de problème d'optimisation

Données :

- Un graphe non orienté $G = (V, E)$ complet
- Une pondération sur les arêtes $\omega : E \rightarrow \mathbb{N}$

Question : Construire un arbre contenant tous les sommets de V dont le poids est minimum. Le poids d'un arbre étant la somme des poids de ses arêtes.

Complexité

Ce problème est polynomial.



Algorithme de Kruskal (1956)

Algorithme

Entrées:

$G = (V, E)$, un graphe non-orienté

$\omega : E \rightarrow \mathbb{N}$

Algorithme:

1: $E' \leftarrow \emptyset$

2: **pour tout** $|E'| < |V| - 1$ **faire**

3: Soit $e \in E - E'$ une arête telle que $(V, E' + \{e\})$ soit acyclique et $\omega(e)$
 soit minimum

4: $E' \leftarrow E' + \{e\}$

5: **renvoie** (V, E')



Leonhard Euler (1707–1783)

Mathématicien et physicien suisse

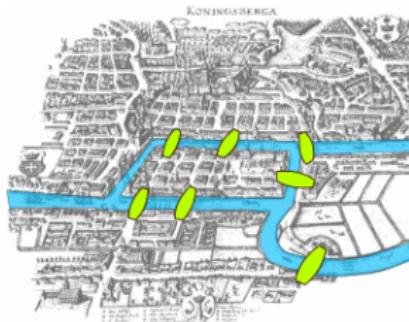
Un des premiers à travailler sur la théorie des graphes



Les sept ponts de Königsberg

Problème historique

- La ville de Königsberg est traversée par une rivière
- Il y a sept ponts pour franchir la rivière
- Est-il possible de faire un circuit passant une et une seule fois par chaque pont ?





Graphe eulérien

Cycle eulérien

Un cycle eulérien est un cycle passant une et une seule fois par chaque arête d'un graphe. Un graphe est dit eulérien s'il admet un tel cycle.

Propriété

Un graphe est eulérienssi le degré de chaque sommet est pair.

Construire un cycle eulérien dans un graphe eulérien est polynomial



Couplage (Matching)

Définition

Un couplage d'un graphe $G = (V, E)$ est un sous-ensemble d'arêtes $M \subseteq E$ tel que $\forall a, b \in M$, a et b n'ont pas de sommets en commun.

Couplage parfait

Un couplage est parfait si tous les sommets du graphe sont couvert par le couplage.

Couplage (Matching)

Couplage parfait de poids minimum

Soient $G = (V, E)$ un graphe et $\omega : E \rightarrow \mathbb{N}$ une fonction de pondération sur les arêtes. $M \subseteq E$ est un couplage parfait de poids minimum ssi M couvre tous les sommets du graphe et M minimise

$$\sum_{e \in M} \omega(e)$$

Complexité

Le problème du couplage parfait minimum est polynomial.



Plan

- 7 Voyageur de commerce
- 8 Digressions nécessaires
- 9 Approximation
- 10 Inapproximation



Algorithme de résolution pour VC

Algorithme

Entrées:

$G = (V, E)$, un graphe non-orienté, $\omega : E \rightarrow \mathbb{N}$

Algorithme:

- 1: Construire T un arbre couvrant de poids minimum de G
- 2: Dupliquer toutes les arêtes de T pour obtenir un graphe eulérien G'
- 3: Construire C' un cycle eulérien dans G'
- 4: **renvoie** C le cycle passant par tous les sommets de G dans l'ordre de leur première occurrence dans C'

Algorithme de résolution pour VC

Algorithme

Entrées:

$G = (V, E)$, un graphe non-orienté, $\omega : E \rightarrow \mathbb{N}$

Algorithme:

- 1: Construire T un arbre couvrant de poids minimum de G
- 2: Dupliquer toutes les arêtes de T pour obtenir un graphe eulérien G'
- 3: Construire C' un cycle eulérien dans G'
- 4: **renvoie** C le cycle passant par tous les sommets de G dans l'ordre de leur première occurrence dans C'





Algorithme de résolution pour VC

Algorithme

Entrées:

$G = (V, E)$, un graphe non-orienté, $\omega : E \rightarrow \mathbb{N}$

Algorithme:

- 1: Construire T un arbre couvrant de poids minimum de G
- 2: Dupliquer toutes les arêtes de T pour obtenir un graphe eulérien G'
- 3: Construire C' un cycle eulérien dans G'
- 4: **renvoie** C le cycle passant par tous les sommets de G dans l'ordre de leur première occurrence dans C'

Performence

- Soit C^* une solution optimale, $\omega(T) \leq \omega(C^*)$
- G' contient les arêtes de T en double donc $\omega(C') = 2\omega(T)$
- Par inégalité triangulaire, $\omega(C) \leq \omega(C')$ donc

$$\omega(C) \leq \omega(C') = 2\omega(T) \leq 2\omega(C^*)$$

Approximation



2-approximation

Quelque soit l'instance, l'algorithme produit une solution au plus 2 fois l'optimal. C'est un algorithme de 2-approximation.



Définitions

rapport d'approximation

Soient \mathcal{P} un problème d'optimisation et f la fonction d'évaluation des solutions de \mathcal{P} . Soient I une instance de \mathcal{P} , S une solution réalisable de I et S^* une solution optimale. Le rapport d'approximation de S sur I est

$$\rho(I, S) = \frac{f(S)}{f(S^*)}$$

ρ -approximation

Soient \mathcal{P} un problème d'optimisation et ρ une fonction $\rho : \mathcal{I} \rightarrow \mathbb{R}^+$. Un algorithme approché \mathcal{A} pour \mathcal{P} est un algorithme fournissant, pour toute instance I de \mathcal{P} , une solution réalisable $\mathcal{A}(I)$ telle que le rapport d'approximation de $\mathcal{A}(I)$ sur I est meilleur que $\rho(I)$.



Améliorer le facteur d'approximation

Algorithme précédent est une 2-approximation

- À cause de la construction du cycle eulérien
- Comment construire un cycle eulérien moins couteux ?

De l'arbre couvrant au cycle eulérien

- Un graphe est eulérien \Leftrightarrow tout sommet a un degré pair.
- Ajouter des arêtes aux sommets de degré impair de l'arbre.
- Le nombre de sommets de degré impair est pair.
- Chercher un couplage parfait de coût minimum entre sommets de degré impair.



Algorithme amélioré de résolution pour VC

Algorithme amélioré

Entrées:

$G = (V, E)$, un graphe non-orienté, $\omega : E \rightarrow \mathbb{N}$

Algorithme:

- 1: Construire T un arbre couvrant de poids minimum de G
- 2: Construire un couplage parfait M de poids minimum du sous-graphe induit par les sommets de degré impair de T .
- 3: Ajouter M à T pour obtenir le graphe eulérien G'
- 4: Construire C' un cycle eulérien dans G'
- 5: **renvoie** C le cycle passant par tous les sommets de G dans l'ordre de leur première occurrence dans C'



Algorithme amélioré : rapport d'approximation

Minorants

Pour borner les rapports d'approximation, on utilise des minorants. Pour le premier algorithme le minorant est

$$\omega(T) \leq \omega(C^*)$$

Pour l'algorithme amélioré le minorant est

$$\omega(M) \leq \omega(C^*)/2$$

Algorithme amélioré : rapport d'approximation

Preuve

- Soit V' les sommets couverts par M
- Soit C_M le circuit obtenu à partir de C en court-circuitant les sommets de $V \setminus V'$
- $\omega(C_M) \leq \omega(C)$ (inégalité triangulaire)
- $|V'|$ est pair donc C_M est l'union de deux couplages parfaits de V'
- Notons M_1 le moins cher des deux
- $\omega(M_1) \leq \omega(C_M)/2 \leq \omega(C^*)/2$
- Le couplage minimum a un poids inférieur à $\omega(c^*)/2$



Algorithme amélioré : rapport d'approximation

Théorème

L'algorithme amélioré a un rapport d'approximation de $\frac{3}{2}$. En effet :

$$\omega(T) \leq \omega(T) + \omega(M) \leq \omega(C^*) + \frac{1}{2}\omega(C^*) = \frac{3}{2}\omega(C^*)$$

Peut-on faire mieux ?

Ce qu'on sait et ce qu'on ne sait pas

Entre autres...

- On ne sait pas s'il existe un meilleur algorithme d'approximation.
- Si le poids est $\omega : E \rightarrow \{1, 2\}$ alors il existe une $7/6$ -approximation.
- Si ω est une fonction asymétrique alors on ne connaît pas de c -approximation avec c une constante.



Plan

7 Voyageur de commerce

8 Digressions nécessaires

9 Approximation

10 Inapproximation



Inapproximation

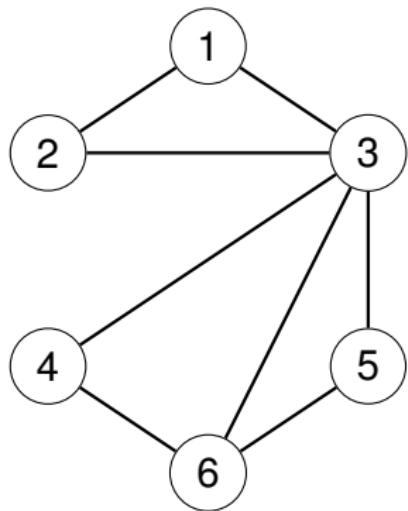
Algorithme pour l'hamiltonicité

On peut déterminer si un graphe G est hamiltonien en utilisant un algorithme pour approximer le problème du voyageur de commerce.

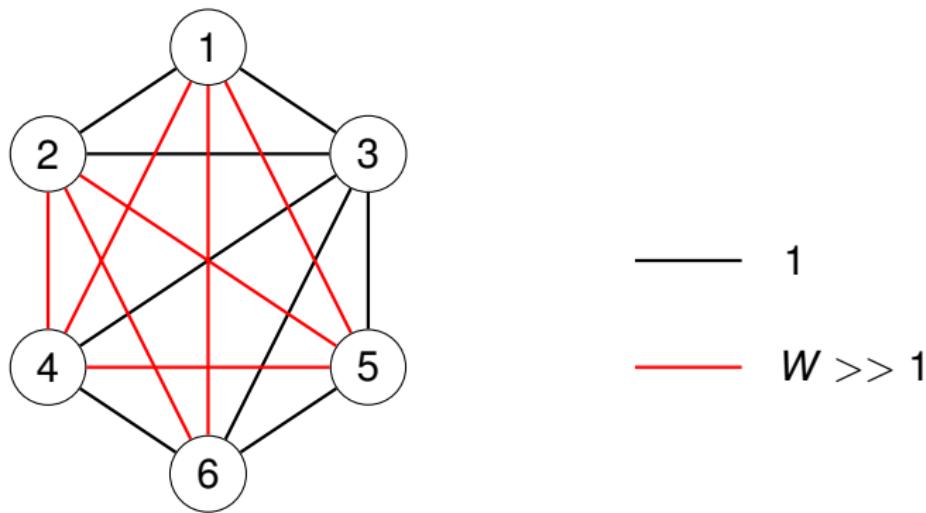
Supposons qu'on dispose d'une c -approximation pour le voyageur de commerce.

- 1 On complète le graphe G avec toutes les arêtes manquantes.
- 2 On donne un poids de 1 pour chaque arrête initiale de G et un poids de $W \gg 1$ pour les nouvelles arêtes.

Exemple de transformation du graphe



Exemple de transformation du graphe





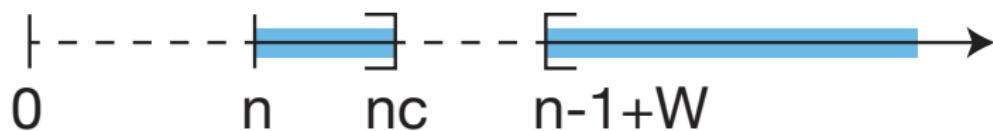
Poids d'une solution optimale

Deux types de solutions

- S'il existe dans G un cycle hamiltonien alors
 - le poids d'une solution optimale au voyageur de commerce est n (le nombre de sommets de G) ;
 - une c -approximation donnera une solution d'au plus $n \times c$.
- Sinon,
 - le poids d'une solution optimale est au moins $n - 1 + W$;
 - une c -approximation donnera une solution d'au moins $n - 1 + W$.

Gap

Quelque soit c , le rapport d'approximation, il existe un W suffisamment grand, qui introduit un trou, "gap", séparant les instances positives des négatives.



$$nc < n - 1 + W$$

$$n(c - 1) + 1 < W$$



Inapproximation

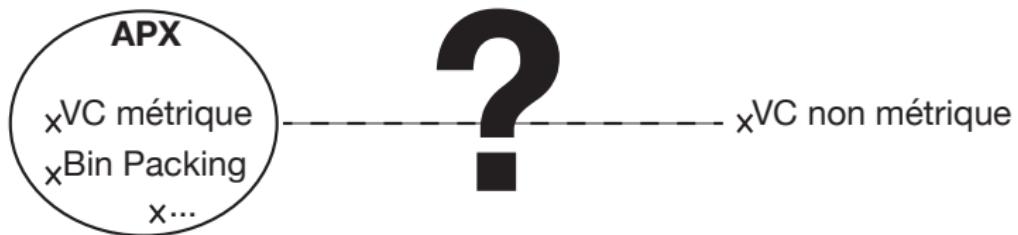
Lemme

S'il existe une algorithme permettant d'approximer le problème du voyageur de commerce non métrique alors il est possible de déterminer si un graphe est hamiltonien en temps polynomial.

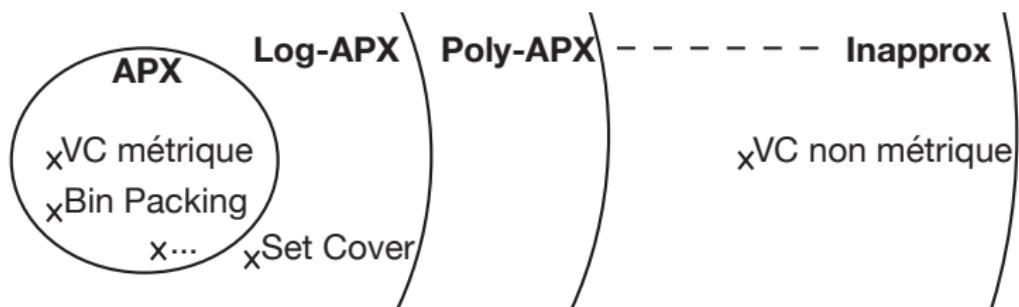
Théorème

Si $P \neq NP$, alors il n'existe pas d'algorithme d'approximation pour le problème du Voyageur de Commerce non métrique.

Entre approximable et inapproximable ?



Entre approximable et inapproximable ?



Couverture par des ensembles

Set Cover

Étant donnée une collection C de sous-ensembles d'un ensemble finis S , trouver un $C' \subseteq C$, de taille minimum, tel que chaque élément de S appartienne à au moins un membre de C' .

S • • • • • • • • • • •

Couverture par des ensembles

Set Cover

Étant donnée une collection C de sous-ensembles d'un ensemble finis S , trouver un $C' \subseteq C$, de taille minimum, tel que chaque élément de S appartienne à au moins un membre de C' .



Couverture par des ensembles

Set Cover

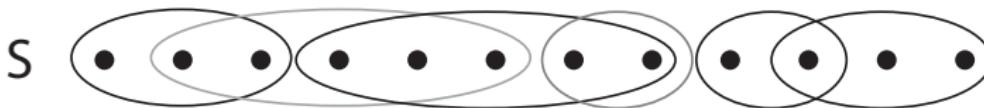
Étant donnée une collection C de sous-ensembles d'un ensemble finis S , trouver un $C' \subseteq C$, de taille minimum, tel que chaque élément de S appartienne à au moins un membre de C' .



Couverture par des ensembles

Set Cover

Étant donnée une collection C de sous-ensembles d'un ensemble finis S , trouver un $C' \subseteq C$, de taille minimum, tel que chaque élément de S appartienne à au moins un membre de C' .



$\log(S)$ -approximation

En choisissant de manière gloutonne le sous-ensemble couvrant le plus grand nombre d'éléments non-couverts, on peut construire une $\log(S)$ -approximation.

Conclusion

Ce que l'on a vu

- La modélisation de problèmes classiques (Stable, Clique, Hamiltonien, SAT, voyageur de commerce).
- Comment aborder ces problèmes *NP*-complets avec des l'introduction du concept d'algorithme d'approximation.
- Un algorithme d'approximation.
- Un résultat d'inapproximation.



Conclusion

Ce que l'on a vu

- La modélisation de problèmes classiques (Stable, Clique, Hamiltonien, SAT, voyageur de commerce).
- Comment aborder ces problèmes *NP*-complets avec des l'introduction du concept d'algorithme d'approximation.
- Un algorithme d'approximation.
- Un résultat d'inapproximation.

Ce que l'on va voir

- Comment formaliser notre démarche.
- Qu'il est possible, dans certain cas, des approximations encore plus efficace.
- Que l'on peut affiner les classes de complexité.