Algorithme avancé

Valeran MAYTIE

mail : marc-atoine.weisserc@centralsupelec.fr

Contents

1 NP-Complétude

1.1 Séquençage ADN

Recomposition de séquence ADN Le but est de concaténer, dans le bon ordre, les différents fragments d'ADN afin de reconstituer le fragment d'ADN

Contraintes :

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

Problème compliqué, explose le temps de calcul

1.2 Sir Hamilton

1.2.1 Le voyage fermé autour du monde

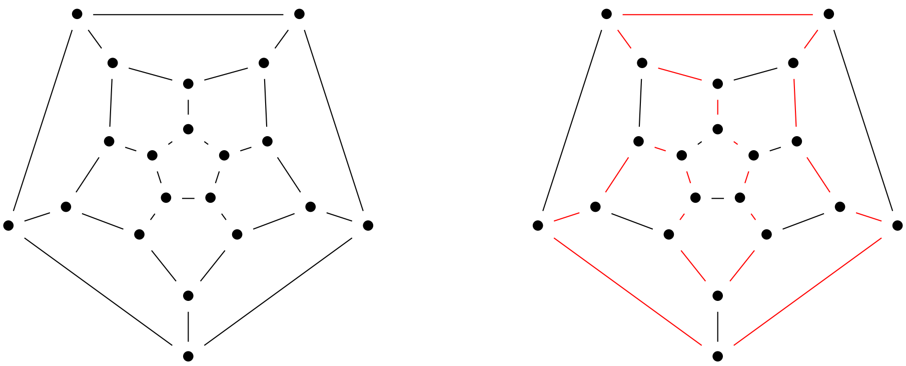


Figure 1: Cycle traversant 20 villes sans passer par les mêmes routes

1.2.2 Modélisation

Données : Un graphe $G = (V, E)$

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

1.2.3 Problème d'assemblage → Chemin hamiltonien

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

Exemple :

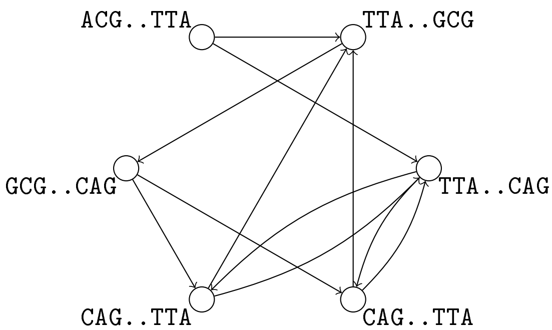


Figure 2: Exemple de réduction de séquencage ADN vers Chemin hamiltonien

1.3 La machine de Turing

1.3.1 Définition

Une définition formelle des machines de Turing :
(Q, A, B, Q_0, Q_f, T)
avec : $Q_0 \subseteq Q, Q_f \subseteq Q, T = Q \times (B \times \{\leftarrow, I, \rightarrow\} \times B)^k \times Q$

- Q :États
- A :Alphabet d'entré
- B :Alphabet de travail
- Q_0 :États initia
- Q_f :États finals
- T : Transition (k nombre de bandes)

Exemple :

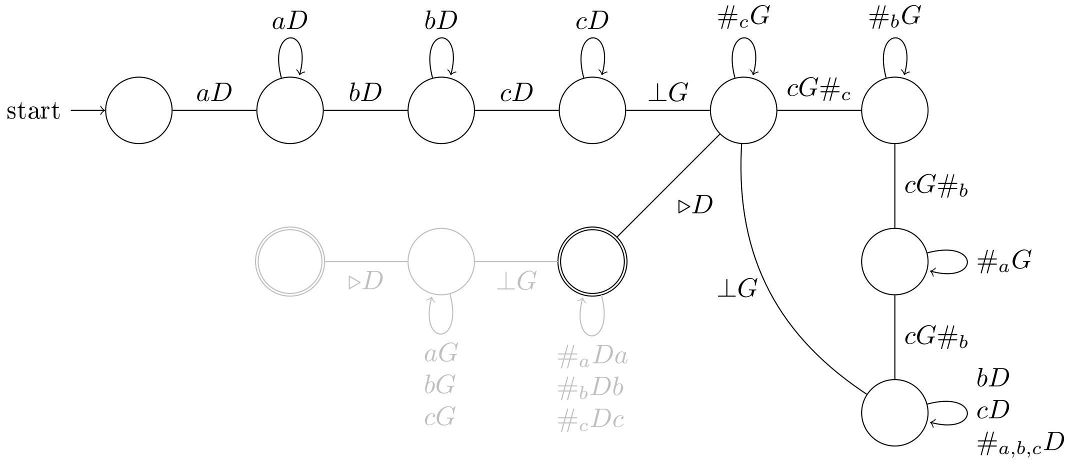


Figure 3: Machine de Turing reconnaissant $\{a^n b^n c^n | n \geq 1\}$

On peut exécuter cette machine sur une ou plusieurs bandes infinies ou bi-infinies, comme celle-ci :

$\triangleright$	a	a	a	b	b	b	c	c	c	...

1.3.2 Classe P

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

1.4 Classe NP

1.4.1 Définition formelle

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

$$I \in D^+ \Leftrightarrow \exists x, R(I, x)|x| \leq p(|I|)$$

Remarque : I est une instance positive et R est calculable en temp polynomial par une machine de Turing.

1.4.2 P et NP

On peut facilement montrer que $P \subseteq NP$ (il suffit de lancer l'algorithme polynomial sur l'entrée).

Par contre, nous ne savons toujours pas si NP est inclus dans P. C'est un problème ouvert depuis très longtemps (1.000.000$).

On ferra donc la conjecture suivante : $P \neq NP$.

1.4.3 Exemples, Problème du stable

Données :

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

Question : Existe-t-il un stable S tel que :

- $S \subseteq V$ avec $|S| \leq k$
- $\forall u, v \in S, \{u, v\} \notin E$

Stable est bien dans NP, il est facile de vérifier qu'un ensemble S est bien de cardinal plus petit que k, et qu'il n'y est pas d'arrête dans E qui relit deux éléments de S.

1.5 Réduction Polynomiale

1.5.1 Principe

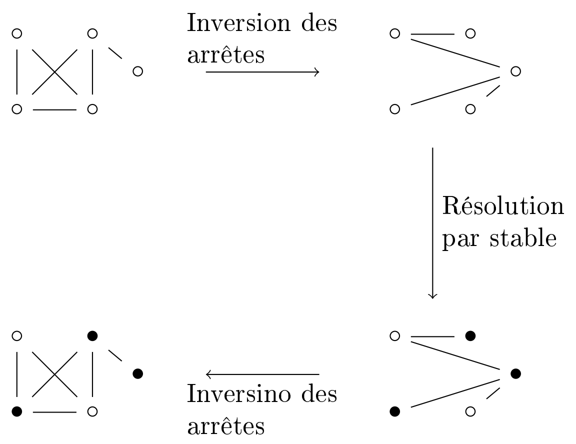
Problème de la clique :
Données :

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

Question : Existe-t-il une clique S telle que :

- $S \subseteq V$ avec $|S| \leq k$
- $\forall u, v \in S, \{u, v\} \in E$

On peut résoudre clique à l'aide de stable :



Cet algorithme nous montre que clique peut se résoudre à l'aide d'un algorithme par stable en temps polynomial (si stable est polynomial). On appelle ça une réduction polynomiale.

Soit $D_1 = (D_1^+, D_1^-)$ 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 $f : D_1 \to D_2$ telle que :

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

1.5.2 NP-Complétude

Un problème D est NP-difficile si $D' \leq D, \forall D' \in NP$
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 $P = NP$

1.5.3 Le problème SAT

Entrée : une formule sous Forme Normale Conjonctive

- 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 affection de valeurs aux variables telle que toutes les clauses soient satisfaites ?

Théorème de Cook-Levin, 1977 : SAT est NP-complet

1.5.4 SAT $\leq$ Stable

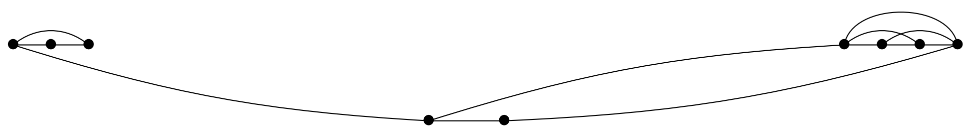
On sait que Stable est déjà un problème NP, on veut montrer qu'il est NP-difficile. On va donc faire une réduction polynômial de SAT vert stable (SAT $\leq$ Stable).

Algorithme :

1. On crée un sommet pour chaque occurrence de variable dans une clause.
2. On crée des liens entre les sommets d'une même clause.
3. Enfin on crée des liens entre les variables entre les occurrences positive et négative.

Exemple :

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



Preuve :

- SAT $\Rightarrow$ stable de taille k ou plus
  - Chaque clause est salifiable
  - On peut construire un stable de taille k en sélectionnant les littéraux vrais dans chaque clauses.
- Stable de taille k ou plus $\Rightarrow$ SAT
  - Chaque sommet du stable correspond à un littéral satifiant une clause différentes (étape 2 de l'algo).
  - Par construction, le stable ne contient pas deux sommets dont l'un correspond à une variable et à sa négation (étape 3 de l'algo).

1.5.5 SAT $\leq$ D-HAM

Voir schéma compliqué fait ici

1.6 Retour sur le Séquençage

Notre problème de raboutage ADN est donc compliqué (NP, la preuve de NP-difficile est facile à faire).

On veut une solution qui a "plus de sens" du point de vue de la biologie. Pour cella, on va rajouter des poids aux arrêts pour avoir des variations selon le temps de ressemblance sur le bout de l'ADN. On ne pourra plus utiliser les cycles hamiltonien, car il ne prend pas en compte les poids, à la place, on se basera sur le problème du voyageur de commerce.

2 Voyageur de commerce

2.1 Présentation du problème

Problème sur les graphes :

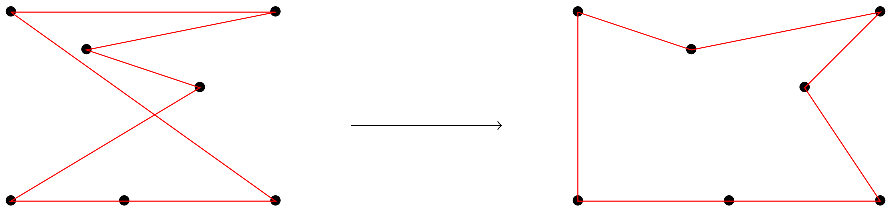


Figure 4: Exemple problème Voyageur de commerce

Données :

- $G = (V, E)$, complet
- $\omega : E \to \mathbb{N}$, respecte l'inégalité triangulaire
- $K \in \mathbb{N}$

Question : Existe-t-il un cycle hamiltonien de poids plut petit que K

Résolution ? :

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

2.2 Digressions nécessaires

2.2.1 Arbres couvrants de poids minimum

Données

- Un graphe non orienté $G = (V, E)$
- $\omega : E \to \mathbb{N}$, pondération sur les arrêtes

Questions : Construire un arbre contenant tous les sommets de V dont le poids minimum est minimum. Le poids d'un arbre étant la somme des poids de ces arrêtes

2.2.2 Graphe eulérien

Un cycle est eulérien est un cycle passant une et une seule fois par chaque arrête d'un graphe. Un graphe est eulérien s'il admet un tel cycle.
Un graphe est eulérien ssi le degré de chaque sommet est pair.

2.2.3 Couplage (Matching)

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

2.2.4 Set Cover

Données

- E un ensemble d'éléments
- S une famille de sous ensembles de E
- $K \in \mathbb{N}$

Questions : Existe-t-il un sous ensemble C de S de taille plus petit que K tel-que l'union de tous les éléments de C soit égale à E

2.3 Approximation

2.4 Inapproximation

3 Exemple : Centre médical pour des villes

Données :

- $G = (V, E)$
- $\omega : E \to \mathbb{N}$
- $D \in \mathbb{N}$, distance maximale (K-Centre)
- $K \in \mathbb{N}$, le nombre max de maison (Dominant)

Question : Existe-t-il un ensemble $M \subseteq V$ de centre tel-que toutes les villes soient à une distance plus petite que D d'un centre

3.1 K-Centre

On cherche à optimiser la distance (de gauche).
Trouver $M \subseteq V, |M| \leq K$ et $\max_{v \in V} \min_{m \in M} d(v, m)$ est minimum

3.2 Dominant "à peu près"

On cherche à minimiser la distance (de droite).
Trouver $M \subseteq, \forall v \in V, \min_{m \in M}(v, m) \leq D$ et $|M|$ est minimum