Information, Calcul et Communication: Cours 6

Résumé de la dernière leçon:

- Tout problème n'est pas soluble par un algorithme !
  Alan Turing, 1936: "Le problème de l'arrêt est indécidable."
- Classes de complexité des problèmes:
  - La classe P des problèmes dits "faciles" à résoudre
  - La classe NP des problèmes dits "faciles" à vérifier
  - Les problèmes NP-complets: les plus difficiles de la classe NP

Plan de cette leçon:

- Les problèmes NP-difficiles
- Le problème du voyageur de commerce

Introduction: former un comité

Vous désirez former un comité pour organiser un voyage d'études.

Votre but est que le comité propose une destination pour ce voyage qui trouve un écho favorable auprès de la majorité de classe, et ce, le plus rapidement possible.

Qui allez-vous choisir pour former ce comité?

On peut voir ce problème comme:

une fonction (le temps passé à obtenir une majorité) à minimiser en fonction du choix du comité (un sous-ensemble S des gens de la classe)

Introduction: former un comité

Votre problème: la fonction à minimiser est complexe!

Par exemple, des choix a priori simples ne font pas l'affaire:

- le comité = tout le monde $\longrightarrow$ débats infinis...
- le comité = une seule personne $\longrightarrow$ difficile d'obtenir une majorité

Et il y a beaucoup de possibilités intermédiaires !
(plus précisément: 2^n s'il y a n personnes dans la classe)

Pire: Si on vous propose un comité (= un sous-ensemble $S \subset \{1, \ldots, n\}$), il n'est (a priori) pas facile de vérifier que ce comité est le meilleur !

Ce problème est NP-difficile.

Problèmes NP-difficiles et NP-complets

Définition (plus précise que la dernière fois)

- Un problème NP-difficile est un problème tel que tout problème de la classe NP peut être réduit en temps polynomial à ce problème.

- Un problème NP-complet est un problème NP-difficile qui appartient à la classe NP.

Dit plus simplement:

- Un problème NP-difficile est au moins aussi difficile à résoudre que tout problème appartenant à la classe NP.

- Un problème NP-complet fait partie des problèmes les plus difficiles à résoudre dans la classe NP.

Un autre exemple de problème NP-difficile

Le problème du voyageur de commerce (version "euclidienne")

Etant donné un ensemble de n villes sur une carte, trouver le chemin fermé le plus court passant une et une seule fois par chacune de ces villes.

Note: On suppose ici des voyages à vol d'oiseau entre chaque ville.

© Swiss Olympic Team

Le problème du voyageur de commerce : méthodes de résolution

- Comme on l'a déjà vu, il est impossible de tester toutes les possibilités afin de résoudre le problème, même pour des valeurs relativement faibles de n (ex : $n = 26 \longrightarrow n! \simeq n^n \cdot e^{-n} \cdot \sqrt{2\pi n} \simeq 4 \cdot 10^{26}$).

- On pourrait argumenter que parmi toutes ces possibilités, un grand nombre d'entre elles sont absurdes (comme par exemple : Genève - Lugano - Lausanne - St Gall - Sion ...) et ne méritent donc pas d'être testées ; il n'empêche que le choix reste très grand parmi les possibilités dites "raisonnables".

Essais avec complexité temporelle polynomiale en n

(A) 1. Partir d'une ville (choisie au hasard, peu importe)
2. Tant qu'il reste au moins une ville à explorer, se diriger vers la ville non explorée la plus proche
3. Revenir à la ville de départ

Cet algorithme a le mérite d'être simple à décrire (ainsi que d'être de complexité temporelle polynomiale en n), mais peut malheureusement amener à des parcours beaucoup trop longs par rapport à la solution optimale !

(B) 1. Choisir un chemin fermé $(v_1, v_2, \dots, v_n, v_1)$ au hasard

2. Pour i allant de 1 à n :

> permuter les villes v_i et v_{i+1} dans le chemin si cette permutation raccourcit la longueur totale du chemin

Le défaut de cet algorithme est que la plupart du temps, il ne trouve qu'un minimum local de la fonction à minimiser (ici, la longueur totale du chemin), et non le vrai minimum global. Pour pallier à ce problème, on peut relancer plusieurs fois l'algorithme, sans garantie de réussite toutefois.

(C) D'autres algorithmes, se basant encore plus sur le hasard que l'algorithme précédent, permettent de trouver des chemins de longueur satisfaisante en pratique, sans toutefois ne fournir aucune garantie quant à leur proximité de la solution optimale.

[Notez que le test de Miller-Rabin fait aussi un usage intensif du hasard pour tester si un grand nombre N est premier ou pas, et fonctionne très bien également en pratique.]

Algorithme de résolution avec garantie d'approximation (et complexité temporelle polynomiale en n)

Théorème

Soit L_{min} la longueur du chemin optimal.
Il existe un algorithme de complexité temporelle polynomiale en n permettant de trouver un chemin de longueur $L \leq 2L_{\text{min}}$.

Note: Vu qu'on ne connaît pas le chemin optimal (et donc pas non plus L_{min}), on est en droit de se demander à ce stade comment il est possible de garantir de trouver un chemin de longueur plus petite ou égale à $2L_{\text{min}}$. Nous allons voir comment cela est possible !

Première étape : arbre couvrant minimal

Etant donné des villes sur une carte, on cherche à les relier par un arbre dont la longueur totale des branches est minimale.

Exemple :

Il est possible de trouver un tel arbre (T) en temps polynomial en n :

1. Commencer avec $T = \{$une ville prise au hasard$\}$ (= racine de l'arbre)
2. chercher parmi les villes restantes la ville v qui soit la plus proche d'une des villes $w \in T$: rajouter v et le lien v-w à T
3. recommencer en 2 jusqu'à ce que T contienne toutes les villes

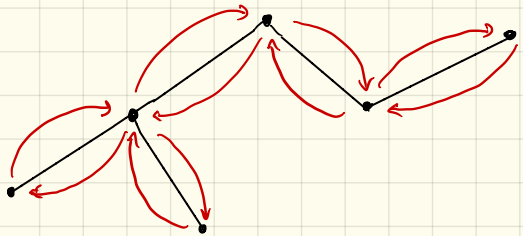
Remarque importante:

> Si L_T désigne la somme des longueurs des branches de l'arbre couvrant minimal, et L_{min} désigne la longueur du trajet optimal qui passe une fois par chaque ville, alors $L_{min} \geq L_T$.

Supposons en effet que ce ne soit pas le cas, i.e., que $L_{min} < L_T$. Dans ce cas, on pourrait couper en un endroit le trajet de longueur optimale et obtenir ainsi un "arbre" dont la longueur totale serait $\leq L_{min} < L_T$. Ce qui serait une contradiction, car T est l'arbre couvrant minimal. On en déduit que $L_{min} \geq L_T$. #

Deuxième étape : parcours le long de l'arbre

Illustration :



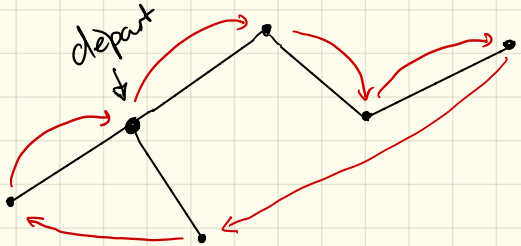
la longueur du trajet rouge vaut $2L_T \leq 2L_{min}$.

Presque ce qu'on veut, mais le trajet rouge n'est pas un trajet qui passe une fois par chaque ville...

Troisième et dernière étape : prendre des raccourcis !

On modifie le trajet rouge en suivant la règle : dès qu'une ville a été déjà visitée, on prend un raccourci vers la ville suivante dans l'arbre.

Illustration :

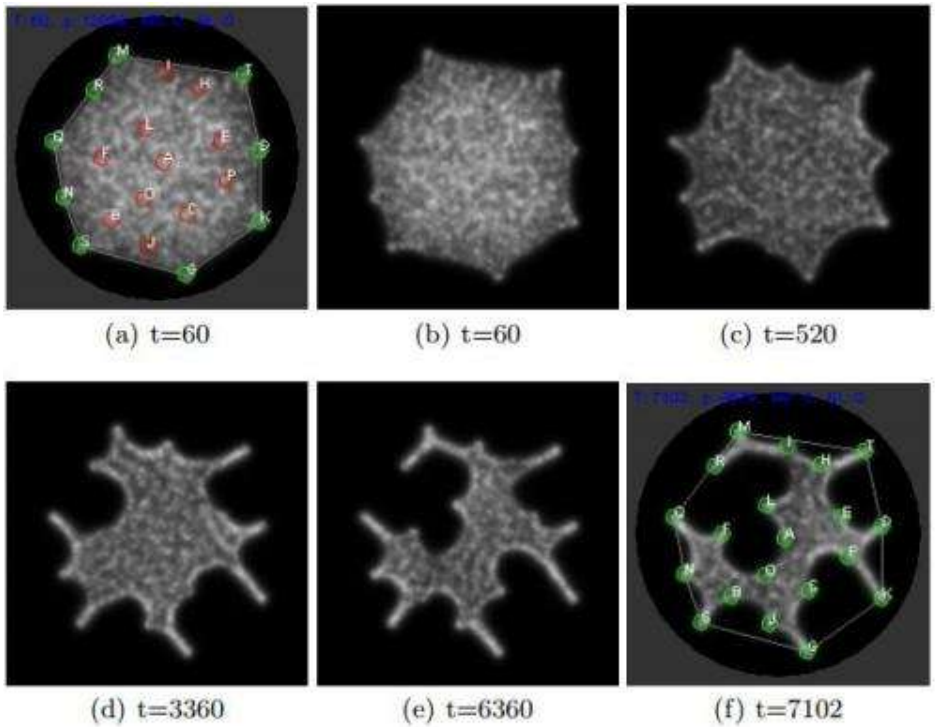


Ce nouveau trajet rouge n'est pas plus long que le précédent (donc $\leq 2L_{min}$), mais il remplit cette fois la condition de passer une fois dans chaque ville. #

Encore mieux. . . (pour la version euclidienne du problème)

Depuis 2010, on sait que pour tout $\varepsilon > 0$, il existe un algorithme de complexité temporelle polynomiale en n permettant de trouver un chemin de longueur $L \leq (1 + \varepsilon)\, L_{\text{min}}$.

Pour finir: en voilà un qui résoud le problème tout seul !



Ref: "Computation of the Travelling Salesman Problem by a Shrinking Blob"

Il s'agit d'un "blob" (plus savamment, un *physarum polycephalum*), organisme unicellulaire capable d'optimiser sa structure interne pour accéder à de la nourriture.