

Introduction

MTH6311

S. Le Digabel, École Polytechnique de Montréal

H2018

(v1)

Plan

1. Introduction
2. Exemples de problèmes
3. Algorithmes

1. Introduction

2. Exemples de problèmes

3. Algorithmes

Optimisation combinatoire

- ▶ L'**optimisation combinatoire** est un domaine qui étudie les problèmes de la forme

$$\min_{x \in \mathcal{X}} \{f(x) : x \in \Omega\}$$

où \mathcal{X} est un ensemble **discret** et fini et $\Omega \subseteq \mathcal{X}$ est l'ensemble des **solutions réalisables**. La **fonction objectif** f prend ses valeurs sur \mathcal{X} .

- ▶ En théorie, une solution optimale peut être obtenue en énumérant toutes les solutions réalisables et en conservant la meilleure.
- ▶ En pratique, ce procédé est trop long. Même l'énumération peut poser problème : comment formuler une solution réalisable ?

Optimisation combinatoire

- ▶ On veut donc résoudre ce problème en un temps raisonnable, que l'on va étudier grâce aux outils de la **théorie de la complexité**.
- ▶ On apprendra à distinguer les problèmes dits **faciles** des problèmes **difficiles**.
- ▶ Pour les faciles, une **résolution exacte** en un temps court est envisageable.
- ▶ Un grand nombre de problèmes sont difficiles. Des solutions exactes sont envisageables, mais dans un délai acceptable, on se contentera de **solutions approchées** obtenues par des méthodes **heuristiques**.
- ▶ Certaines propriétés seront toutefois obtenues (garantie de performance, écart à la solution optimale, etc.)

Notions

- ▶ Notions mathématiques de base.
- ▶ Modélisation.
- ▶ Théorie des graphes.
- ▶ Conception d'algorithmes et programmation.

1. Introduction

2. Exemples de problèmes

3. Algorithmes

Problème v.s. Instance

- ▶ Un **problème** correspond au modèle

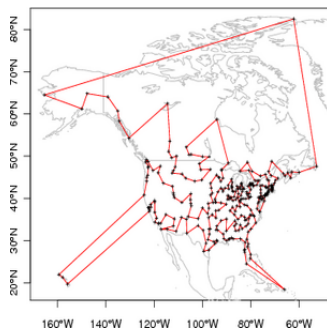
$$\min_{x \in \mathcal{X}} \{f_a(x) : x \in \Omega_a\}$$

dans lequel a est un ensemble de paramètres non déterminé (c'est le cas général).

- ▶ Une **instance** du problème est une formulation du modèle dans laquelle a est déterminé (c'est un cas particulier).
- ▶ En pratique, on conçoit souvent des algorithmes pour des problèmes, qu'on teste sur plusieurs instances.
- ▶ On peut aussi s'intéresser à une seule instance ou à une famille d'instances particulières. Dans ce cas on concevra une méthode plus spécialisée.

Voyageur de commerce (TSP)

- ▶ Un voyageur de commerce doit visiter un certain nombre de villes, et chaque ville une et une seule fois. Étant donné des distances entre chaque paire de villes, il doit minimiser la distance totale parcourue.



TSP : variation pour distribution de courrier

The Times and The Sunday Times e-paper - News Review - 6 Jan 2013 -... <http://epaper.thetimes.co.uk/epaper/services/OnlinePrintHandler.ashx?is...>

Postie-haste delivery

A German postman was taken to court for being too efficient. When the 53-year-old man from Rosenheim, near Munich, finished his round before everybody else, his colleagues reported him, suspecting he was throwing away mail. But he told a district court he had simply studied the quickest routes to speed up his round.

According to the Münchner Merkur newspaper, his supervisor told the court she knew of his time-saving methods. "I admit that some of them are possibly logical," she said, "but they could not be officially accepted as they did not fit the rules." The case was dismissed.

TSP (suite)

- ▶ On peut représenter ce problème par un graphe : chaque ville correspond à un sommet et chaque arête à une paire de villes pouvant être visitées l'une à la suite de l'autre.
- ▶ Le problème correspond à trouver un tour complet (circuit Hamiltonien) dans ce graphe qui minimise la somme des distances (f).
- ▶ L'ensemble \mathcal{X} correspond à un chemin dans le graphe, soit une liste de villes, et l'ensemble $\Omega \subset \mathcal{X}$ est l'ensemble des circuit Hamiltoniens.
- ▶ Le nombre de solutions pour n villes est de $(n - 1)!/2$. Si $n = 4$, il y a trois solutions possibles. Si $n = 30$, il y en a 4 420 880 996 869 850 977 271 808 000 000 !
- ▶ Voir liens sur le [site web](#).

Problème d'affectation

- ▶ n tâches à affecter à n machines de telle sorte que chaque machine soit affectée à une seule tâche et chaque tâche soit affectée à une seule machine.
- ▶ Le coût d'affecter la tâche j à la machine i est c_{ij} .
- ▶ L'objectif est de minimiser la somme des coûts
- ▶ Ω est l'ensemble de toutes les affectations possibles des n tâches aux n machines. Il y en a $n!$.
- ▶ Modèle en variables binaires :

$$\min_{x \in \mathcal{X} = \{0,1\}^{n^2}} f(x) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$$

sujet aux $2n$ contraintes $\left\{ \begin{array}{l} \sum_{j=1}^n x_{ij} = 1, i = 1, 2, \dots, n \\ \sum_{i=1}^n x_{ij} = 1, j = 1, 2, \dots, n \end{array} \right.$
 (c'est l'ensemble Ω).

Liste non exhaustive de problèmes

- ▶ Colorations de graphes.
- ▶ Routage.
- ▶ Satisfaisabilité.
- ▶ Classification.
- ▶ Horaires.
- ▶ Boîtes noires (*projet*).

1. Introduction

2. Exemples de problèmes

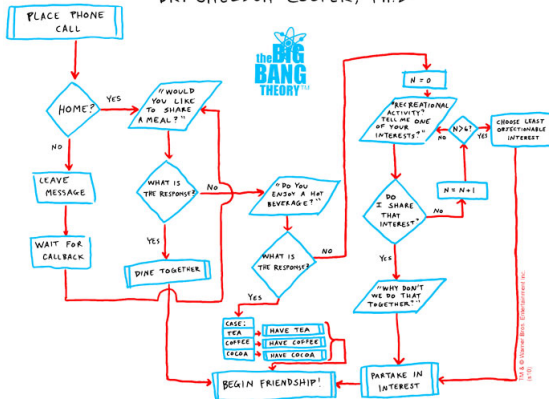
3. Algorithmes

Algorithmes

Un algorithme est la description d'une séquences d'instructions à entreprendre afin de résoudre un problème. Par exemple :

THE FRIENDSHIP ALGORITHM

DR. SHELDON COOPER, Ph.D



Il faut choisir un algorithme selon les exigences sur la **qualité de la solution** et le **temps de calcul**. Ces objectifs sont souvent contradictoires.

- ▶ **Algorithmes exacts** : garantissent une solution optimale globale mais peuvent être très longs si le problème est difficile.
 - ▶ Séparation et évaluation (*branch and bound*).
 - ▶ Programmation linéaire pour l'optimisation continue.
 - ▶ Programmation dynamique.
- ▶ **Algorithmes locaux** : garantissent une solution locale.
 - ▶ Méthodes de descente.
 - ▶ Gradient / Newton pour l'optimisation continue.
- ▶ **Métaheuristiques** : très peu de garanties sur la qualité de la solution, mais convergent rapidement.
 - ▶ Algorithme glouton.
 - ▶ Recherche tabou.
 - ▶ Recherche à voisinages variables (VNS).
 - ▶ Algorithmes génétiques.