leo.baty@enpc.fr

$\Rightarrow$ batyleo.github.io/teaching/RO2023/

1. Operation Research: Introduction

27 Septembre 2023

Problems

Voyageur de commerce (optimization)

- Instance : n villes, coordonnées (x_i, y_i), $i \in [n]$
- Question : tour de coût minimum

Problem

- Input/instance
- Question

Two types of problems :

- **Decision problems** : the answer to the question is Yes or No
- **Optimization problems** :

$$\min_x c(x) \text{ s.t. } x \in \mathcal{X} \quad (\mathcal{P})$$

Voyageur de commerce (décision)

- Instance : n villes, $M \in \mathbb{R}_+$
- Question : existe-t-il un tour de longueur $\leq M$

Algorithms

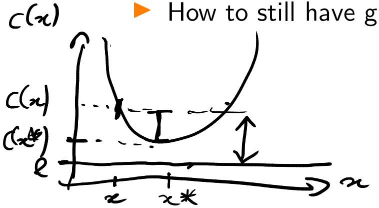
Algorithm f, instance I
- exact $f(I) = x^*$
- approx: $\forall I, f(I) = x, \quad c(x) \leq \alpha c(x^*)$

Algorithm

Sequence of elementary operations that can be implemented on a computer

Types of algorithms for optimization problems :

- **Exact** algorithms : compute an optimal solution
- **Approximation** algorithms : compute a solution with guarantee
- **Heuristic** algorithms : compute a solution with no guarantee
  - How to still have guarantees ?



Algorithms

Algorithm

Sequence of elementary operations that can be implemented on a computer

Types of algorithms for optimization problems :

- **Exact** algorithms : compute an optimal solution
- **Approximation** algorithms : compute a solution with guarantee
- **Heuristic** algorithms : compute a solution with no guarantee
  - How to still have guarantees ? $\rightarrow$ compute lower bound

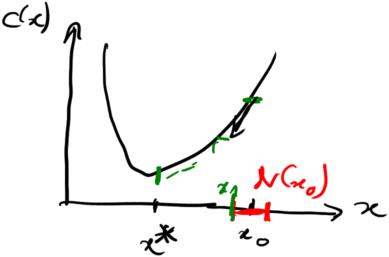
Heuristic example : local descent

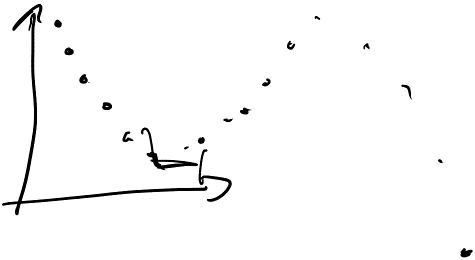
$$(P) : \begin{cases} \min_x & c(x) \\ \text{s.t.} & x \in \mathcal{X} \end{cases}$$

Iterative algorithm : current solution $x_k \in \mathcal{X}$

1. Compute solutions in the neighborhood $\mathcal{N}(x_k)$ of x_k.
2. Select $x_{k+1} \in \mathcal{N}(x_k)$ such that $c(x_{k+1}) < c(x_k)$
3. Stop when we cannot improve anymore.

Designing good local search heuristics will be useful in the project.





Time complexity

Time complexity $f(n)$ of an algorithm : number of elementary operation that must be realized if the input is of size n.

Example

1. Find maximum of n integers ? $O(n)$
2. Sorting n integers ? $O(n \log n)$
3. Multiplication of two matrices of size $n \times n$? $O(n^3)$

Time complexity

Time complexity $f(n)$ of an algorithm : number of elementary operation that must be realized if the input is of size n.

Example

1. Find maximum of n integers ? $\rightarrow \mathcal{O}(n)$
2. Sorting n integers ? $\rightarrow \mathcal{O}(n \log n)$
3. Multiplication of two matrices of size $n \times n$? $\rightarrow \mathcal{O}(n^3)$

Time complexity

Time complexity $f(n)$ of an algorithm : number of elementary operation that must be realized if the input is of size n.

Example

1. Find maximum of n integers ? $\rightarrow \mathcal{O}(n)$
2. Sorting n integers ? $\rightarrow \mathcal{O}(n \log n)$
3. Multiplication of two matrices of size $n \times n$? $\rightarrow \mathcal{O}(n^3)$

"Clean" definition of algorithm, size of input, time complexity

- Requires formalizing what is an algorithm on a computer
- See textbook for more details
- Informal understanding is enough for this lecture

Polynomial vs exponential algorithm

- **Polynomial** algorithm : time complexity in $\mathcal{O}(n^a)$
- Otherwise : **exponential** algorithm

Polynomial vs exponential algorithm

- **Polynomial** algorithm : time complexity in $\mathcal{O}(n^a)$
- Otherwise : **exponential** algorithm

Time complexity	**Size n**			
	10	20	50	60
n	$0.01\,\mu$s	$0.02\,\mu$s	$0.05\,\mu$s	$0.06\,\mu$s
n^2	$0.1\,\mu$s	$0.4\,\mu$s	$2.5\,\mu$s	$3.6\,\mu$s
n^3	$1\,\mu$s	$8\,\mu$s	$125\,\mu$s	$216\,\mu$s
n^5	0.1 ms	3.2 ms	312.5 ms	777.6 ms
2^n	$1\,\mu$s	1 ms	13 days	36.5 years

Table – Comparision of different time complexity functions on a computer executing 1 billion operations per second

A question of computational time ?

Let $\mathcal{A}$ be an algorithm solving a problem $\mathcal{P}$ in 2^n operations. We have a computer that solved $\mathcal{P}$ with $\mathcal{A}$ in 1 hour for instances of size up to $n = 438$.

With a computer 1000 times faster, instances of up to which size can we solve in 1 hour ?

$2^{10} \simeq 1024$

A question of computer speed

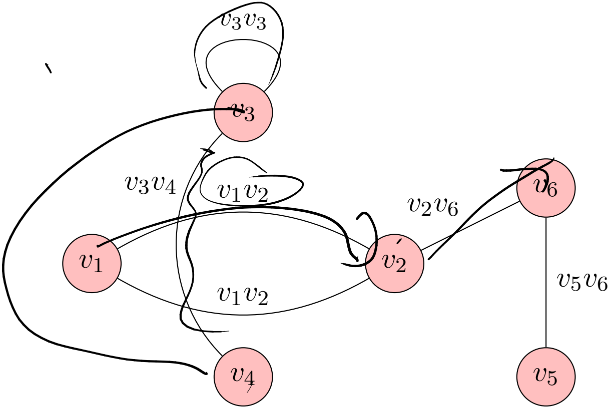
Complexity function	Present day computer	Computer $100\times$ faster	Computer $1000\times$ faster
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.64N_3$	$10N_3$
n^5	N_4	$2.5N_4$	$3.98N_4$
2^n	N_5	$N_5 + 6.64$	$N_5 + 9.97$
3^n	N_6	$N_6 + 6.29$	$N_6 + 6.29$

Table – Size of the largest instance that can be solved in 1 hour

Graphs

Graph : $G = (V, E)$

- V : set of **vertices**
- E : set of **edges** (unordered pairs of vertices)



Paths

Path : sequence of nodes connected by edges

- **Simple** : no edge is crossed twice
- **Cycle** : no vertex is visited twice
- **Elementary** : start vertex = end vertex
- **Eulerian** : crosses all edges exactly once
- **Hamiltonian** : visits all vertices exactly once

Modeling with cycles

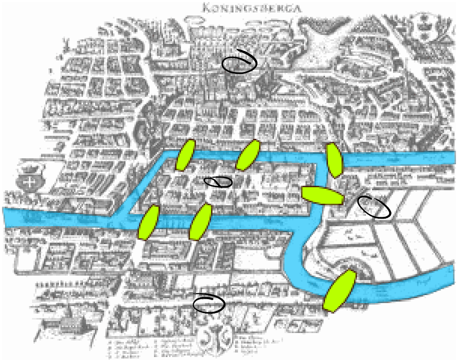
Give examples of real-life problems whose solutions are Hamiltonian/Eulerian cycles.

Modeling with cycles

Give examples of real-life problems whose solutions are Hamiltonian/Eulerian cycles.

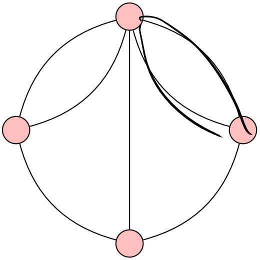
- Traveling Salesman (Hamiltonian cycle)
- Post office (Eulerian cycle)

Example : Königsberg bridges (1736) - Euler



Is it possible to go through all the bridges without crossing the same bridge twice ?

Example : Königsberg bridges (1736) - Euler

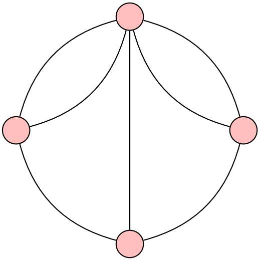


Is it possible to go through all the bridges without crossing the same bridge twice ?

Decision problem : Eulerian cycle

- **Instance**. Graph G
- **Question**. Is G Eulerian ?

Example : Königsberg bridges (1736) - Euler



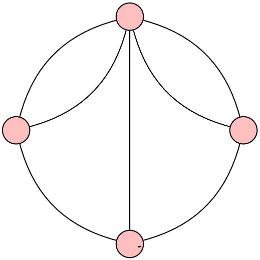
Is it possible to go through all the bridges without crossing the same bridge twice ?

Decision problem : Eulerian cycle

- **Instance**. Graph G
- **Question**. Is G Eulerian ?

Theorem : A graph is Eulerian $\Leftrightarrow$ all its vertices have even degree

Example : Königsberg bridges (1736) - Euler



Is it possible to go through all the bridges without crossing the same bridge twice ?

$O(|V| + |E|)$

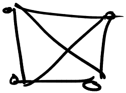
Decision problem : Eulerian cycle

- **Instance**. Graph G
- **Question**. Is G Eulerian ?

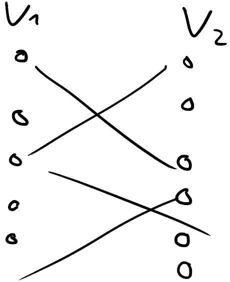
Theorem : A graph is Eulerian $\Leftrightarrow$ all its vertices have even degree

Algorithmic complexity of testing if an Eulerian cycle exists ?

Types of graph



- **Simple** graph : no duplicate edges and no self loops
- **Complete** graph : simple graph where every pair of vertices is an edge
- **Bipartite** graph : vertices partitioned into two subsets, such that there is no edge between two vertices of the same subset



Complete graphs, bipartite graphs

- K_n : complete graph with n vertices $\frac{n(n-1)}{2} = \binom{n}{2}$
  - How many edges in K_n ?
- $K_{m,n}$: bipartite complete graph with m and n vertices
  - How many edges in $K_{m,n}$? $m\,n$

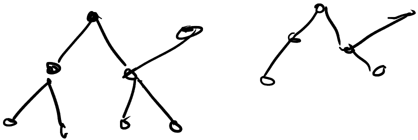
Complete graphs, bipartite graphs

- K_n : complete graph with n vertices
  - How many edges in K_n ? $\rightarrow \frac{n(n-1)}{2}$
- $K_{m,n}$: bipartite complete graph with m and n vertices
  - How many edges in $K_{m,n}$? $\rightarrow mn$

Types of graph

Connexe

- **Connected** : there is a path between any pair of vertices
- **Forest** : no cycle
- **Tree** : connected forest

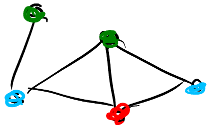


Coloring problem

- **Coloring** : $c : V \to \mathbb{N}$
- Proper coloring : for any edge (u, v), $c(u) \neq c(v)$
- **Chromatic number** $\chi(G)$: minimum number of colors in a coloring

Optimisation problem : graph coloring

- **Instance**. A graph G
- **Question**. Compute $\chi(G)$



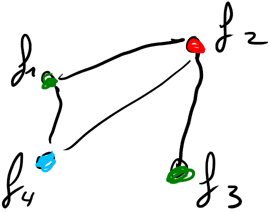
Example of problem that can be modeled with coloring

A set F of formations must be given to employees of a firm. Each employee i must follow a subset F_i of formations. The firm wants to find the minimum number of formation time slots it must schedule so that each employee can attend its formations. Model this problem as a coloring problem.

$$\begin{cases} V = F \\ E = \{(u,v) \mid \exists i : u \in F_i, v \in F_i\} \end{cases}$$

$i_1 : f_1, f_2, f_4$

$i_2 : f_2, f_3$

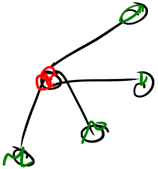


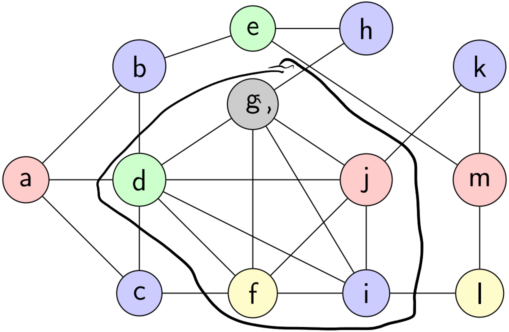
Example of problem that can be modeled with coloring

A set F of formations must be given to employees of a firm. Each employee i must follow a subset F_i of formations. The firm wants to find the minimum number of formation time slots it must schedule so that each employee can attend its formations. Model this problem as a coloring problem.

- Vertices : formations
- Edges : (f_i, f_j) if an employee needs to follow both f_i and f_j
- Each color = 1 time slot

Example



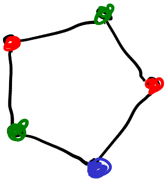


Can we color this graph with fewer than five colors ?

Clique property

- **Clique** : complete subgraph
- cardinal of a clique $\leq$ number of colors in a proper coloring
- $\omega(G)$: maximum cardinality of a complete subgraph of G

Theorem. $\omega(G) \leq \chi(G)$

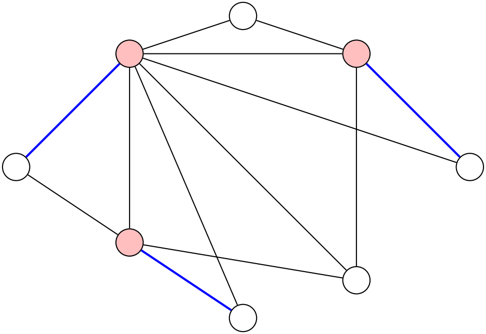


Matching and covers

Couplage

- **Matching** : set of edges two by two disjoints.
- **Vertex cover** : Set of vertices S such that each edge contains a vertex in S.

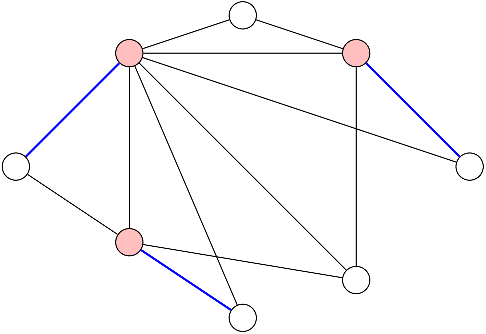
Couverture



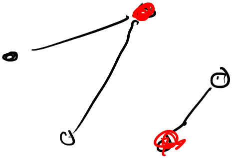
Matching and covers

- $\tau(G)$: minimum cardinality of a cover
- $\nu(G)$: maximum cardinality of a matching

Theorem. $\nu(G) \leq \tau(G)$



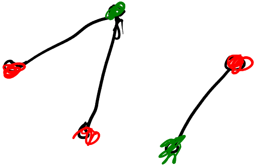
Examples



You are the security guard in a bar. You know which pair of clients will fight if they are both admitted. You want to choose a minimum number of clients to exclude from the bar to avoid any fight.

couverture minimum.

You want to do the seating plan in such a way that guests who fight are not at the same table. How to minimize the number of tables used ? coloration



Examples

You are the security guard in a bar. You know which pair of clients will fight if they are both admitted. You want to choose a minimum number of clients to exclude from the bar to avoid any fight.
$\rightarrow$ Min vertex cover problem

You want to do the seating plan in such a way that guests who fight are not at the same table. How to minimize the number of tables used ?

Examples

You are the security guard in a bar. You know which pair of clients will fight if they are both admitted. You want to choose a minimum number of clients to exclude from the bar to avoid any fight.
$\rightarrow$ Min vertex cover problem

You want to do the seating plan in such a way that guests who fight are not at the same table. How to minimize the number of tables used ?
$\rightarrow$ Coloring problem

Mixed Integer Linear Programming

Programme linéaire en nombres entiers (PLNE)

Mixed Integer Linear Program

$$\begin{cases} \min & c^\top x = \sum_i c_i x_i \\ \text{s.t.} & Ax \leq b \\ & x \in \underline{\mathbb{Z}^p} \times \mathbb{R}^{n-p} \end{cases}$$

with $c \in \mathbb{R}^n$, and $A \in \mathbb{R}^{m \times n}$.

One of the most used framework in Operations Research

- Wide modelling power
- Efficient open source and commercial solvers

$\implies$ very useful in the industry

Commerciaux :
Gurobi
CPLEX

Open source :
HiGHS, GLPK

Example : dinner party

Do the seating plan of a party in such a way that

- guests likely to fight are not at the same table
- guests have as many friends as possible at their table

Model this problem as a mixed integer linear program.

- Variables de décision :
- Fonction objectif :
- Contraintes :

- n invités
- T tables, de taille M.
- $\forall i,j \in [n]$, $f_{ij} = 1$ si i,j amis (0 sinon)

  $e_{ij} = 1$ si i,j ennemis (0 sinon)

- $\forall i \in [n], \forall t \in [T]\ x_{it} \in \{0,1\} = \mathbb{1}(i \text{ sur la table } t)$
- $y_{ijt} \in \{0,1\} = \mathbb{1}(i,j \text{ sur la table } t)$
- Objectif : $\max\limits_{x,y} \sum\limits_{t \in T} \sum\limits_{i,j} f_{ij}\, y_{ijt}$

s.c.

- $\sum\limits_i x_{it} \leq M_t \quad \forall t \in T$
- $\sum\limits_{ij} e_{ij}\, y_{ijt} = 0 \quad \forall t$ | $x_{it} + x_{jt} \leq 1$ (si $e_{ij} = 1$) $\forall t, \forall i,j$
- $\begin{cases} y_{ijt} \leq x_{it} \\ y_{ijt} \leq x_{jt} \end{cases}$ ou $2y_{ijt} \leq x_{it} + x_{jt}$

Input instance :

- Guests $i \in [\![1, G]\!]$, tables $t \in [\![1, T]\!]$
- $e_{ij} = \mathbb{1}(i \text{ will fight } j)$
- $f_{ij} = \mathbb{1}(i \text{ and } j \text{ are friends})$

Input instance :

- Guests $i \in [\![1, G]\!]$, tables $t \in [\![1, T]\!]$
- $e_{ij} = \mathbb{1}(i \text{ will fight } j)$
- $f_{ij} = \mathbb{1}(i \text{ and } j \text{ are friends})$

Decision variables :

- x_{it} equals 1 if i affected to table t, 0 otherwise
- y_{ijt} equals 1 if both i and j affected to t, 0 otherwise

Input instance :

- Guests $i \in [\![1, G]\!]$, tables $t \in [\![1, T]\!]$
- $e_{ij} = \mathbb{1}(i \text{ will fight } j)$
- $f_{ij} = \mathbb{1}(i \text{ and } j \text{ are friends})$

Decision variables :

- x_{it} equals 1 if i affected to table t, 0 otherwise
- y_{ijt} equals 1 if both i and j affected to t, 0 otherwise

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{t=1}^{T} f_{ij} y_{ijt} \\
\text{s.t.} \quad & \sum_{t=1}^{T} x_{it} = 1, && \forall i \\
& x_{it} + x_{jt} \leq 2 - e_{ij}, && \forall i, j, t \\
& y_{ijt} \leq x_{it},\ y_{ijt} \leq x_{jt} && \forall i, j, t \\
& x_{it} \in \{0, 1\},\ y_{ijt} \in \{0, 1\}, && \forall i, j, t
\end{aligned}
$$