Problem Solving
Introduction

Marie Pelleau
marie.pelleau@univ-cotedazur.fr

Acknowledgements

- Wikipedia
- Jean-Charles Régin
- Olivier Bournez, LIX
- Christine Solnon, Université Lyon
- Anne Benoit, ENS Lyon
- Roman Barták, Charles University

Course outline

Lectures

1. Greedy Algorithms
2. Local Search
3. Constraint Programming

Knowledge control

- Mid-term exam
- Final exam

References

- T. Cormen, C. Leiserson, R. Rivest, **Introduction à l'algorithmique**, Dunod
- D. Knuth, **The Art of Computer Programming**
- M. Gondran et M. Minoux, **Graphes et Algorithmes**
- Other books as per your preference: do not hesitate to consult several

Problem

- A problem is a **general question**: shortest path between two points, timetable
- It is described by data and a question
- Answering this question is solving the problem
- In computer science, we seek a general answer, *i.e.*, an algorithm that works in all cases
- **Instance**: A specific set of data, *e.g.*, the shortest path between Nice and Nantes.

Problem

- Some problems are easy: Sorting numbers, Reversing a string
- Others are difficult: Traveling Salesman Problem (TSP)

Traveling Salesman Problem (TSP)

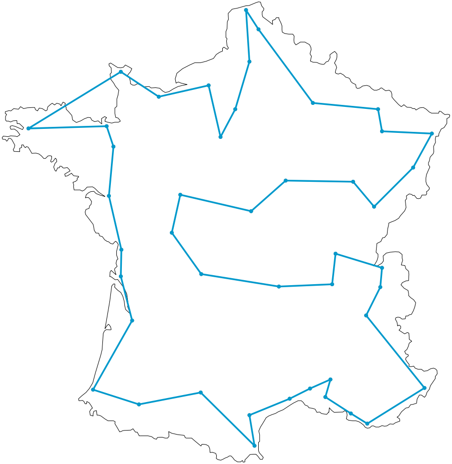
Description

- **Data**: A list of cities and pairwise distances
- **Question**: Find the shortest tour that visits each city exactly once

Mathematical Formulation

Given a complete weighted graph, find a Hamiltonian cycle of minimum weight

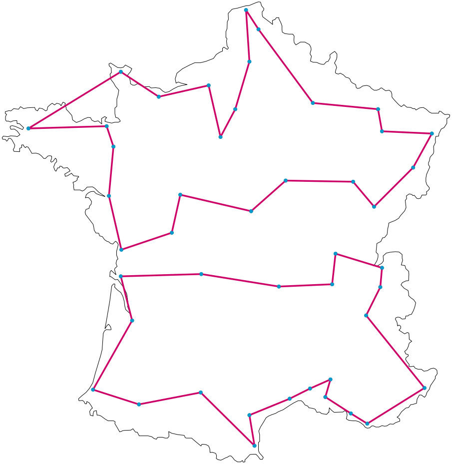
TSP



Description

- All cities are visited exactly once

TSP



Description

- All cities are visited exactly once
- Only one tour (no sub-tours)

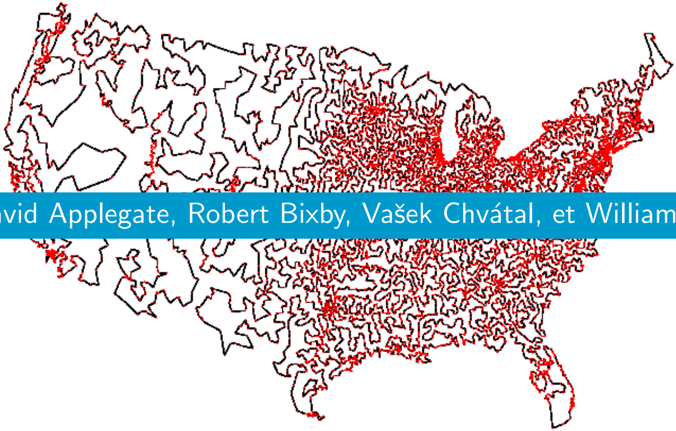
TSP

Description

- Some problems are equivalent to TSP
  - scheduling problems: find the order in which to build objects
- The "pure" version of TSP is rare, in practice often, we encounter variations:
  - Non-Euclidean
  - Asymmetric
- These variations do not make the problem easier
- Common applications
  - Vehicle routing (time windows, pickup and delivery, *etc.*)

TSP

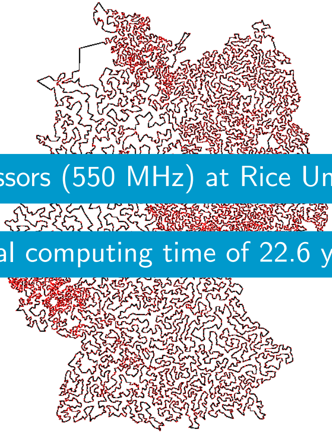
USA 13 509 cities, solved in 1998



By David Applegate, Robert Bixby, Vašek Chvátal, et William Cook

TSP

Germany 15 112 cities, solved in 2001



Network of 110 processors (550 MHz) at Rice University and Princeton

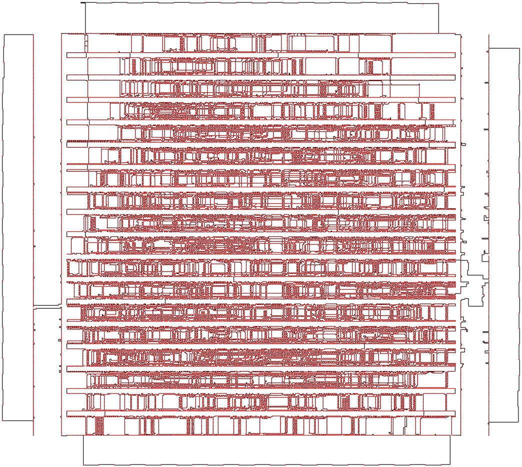
Total computing time of 22.6 years

TSP

Sweden 24 978 cities, solved in 2004

TSP

Microchip 85 900 "cities" solved in 2006



TSP

- Ther exist several solvers
- The most well-known solver is Concorde by William Cook (free)

- TSP solvers are typically dedicated to solving the pure problem
- TSP solvers may not handle even slight variations (asymmetric, additional constraints, *etc.*)

Algorithms

- Not all algorithms are the same, they vary in efficiency, differentiated by:
  - Computation time: slow vs fast
  - Memory usage: low vs high
- We discuss complexity in terms of time (speed) and space (memory used).

Algorithms Complexity

Purpose

- To gauge the difficulty of problems
- To estimate the computation time or space required to solve a problem

- This allows for comparing algorithms
- Expressed as a function of data size and amount

Problem

Not all problems are the same, they vary according to the algortihms used to solve them

- Easy problem: we know an efficient algorithm to solve it
- Difficult problem: we do not know (yet?) an efficient algorithm to solve it
- Undecidable problem: no algorithm exists to solve it

NP-Completeness

THE major current issue in computer science

P vs NP

- Easy problems have polynomial algorithms
- Difficult problems have no known polynomial algorithm
- **Key question**: Does a polynomial algorithm always exist?

NP-Completeness

For some problems, we do not know if there exists polynomial algorithm, we only know exponential algorithms: 2^n

21 NP-Complete Problems by Karp

- Hitting-set: Set covering problem
- Knapsack
- Subset sum
- Bin packing
- Graph coloring
- Maximum clique

Remark

NP-complete problems are all equivalent and resemble each other. They can be transformed into one another

Hitting-set: Set covering

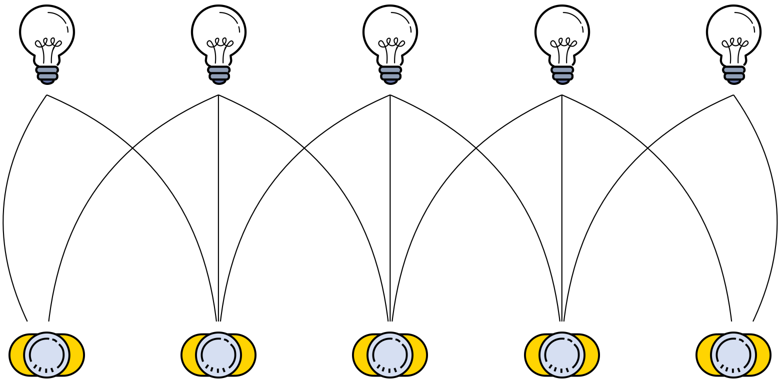
Description

Light bulbs and switches

- A switch is connected to certain bulbs
- When a switch is pressed, all connected bulbs are lit
- **Question**: What is the minimum number of switches needed to light all bulbs?
- We want a general answer that works for all instances

Hitting-set: Set covering

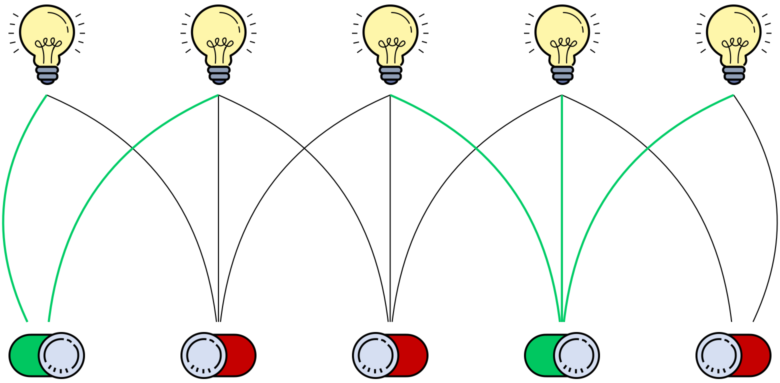
Example



n switches, 2 choices per switch $\Rightarrow 2^n$

Hitting-set: Set covering

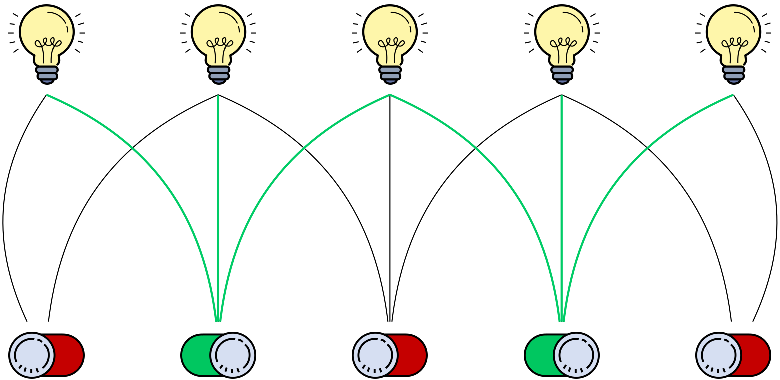
Example



n switches, 2 choices per switch $\Rightarrow 2^n$

Hitting-set: Set covering

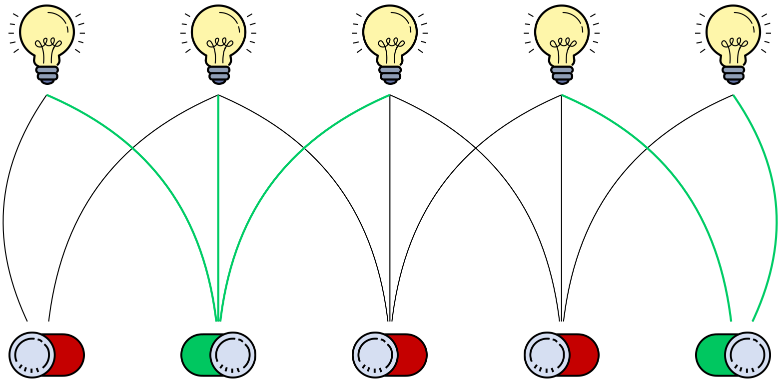
Example



n switches, 2 choices per switch $\Rightarrow 2^n$

Hitting-set: Set covering

Example



n switches, 2 choices per switch $\Rightarrow 2^n$

Graph Vertex Coloring

Description

Vertices of a graph are colored such that no two adjacent vertices share the same color

Principle of contraction and connection

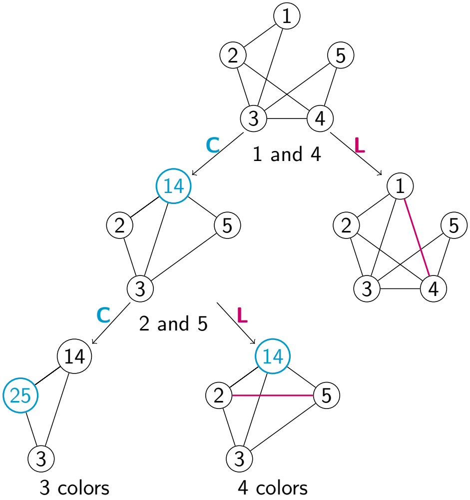
Coloring a complete graph (a clique) with n vertices requires n colors

- Given 2 non-adjacent vertices a and b
- Either they have the same color $\Rightarrow$ Contraction (C)
- Or they have the different color$\Rightarrow$ Connection (L)
- Reaching a clique $\Rightarrow$ the number of vertices gives the number of colors

p possible edges, 2 choices per edge (same color or different color) $\Rightarrow 2^p$

Graph Vertex Coloring

Example



Easy vs Difficult

- Given a matrix
- For each row and each column, the number of 1's is known
- Define precisely the 0's and 1's of this matrix

The difference can be subtle

- Pure problem: easy
- connectivity is introduced: difficult
- Convexity is introduced: difficult
- connectivity and convexity are introduced : easy

Decision Problem

A decision problem is a mathematically defined question about given parameters that requires a **yes or no** answer

Example

- Given a set of cities and a distance d, is there a path visiting all cities with a total length less than d?
- Can a graph be colored with k colors?

Optimization Problem

- An Optimization Problem involves finding the **best solution** among feasible options
- An Optimization Problems has an objective function (min or max)
- An **optimal** solution minimizes (or maximizes) the objective function among all feasible solutions

Example

- Shortest path visiting all the cities?
- Minimum number of colors to color a graph vertices?

Optimization Problem

It is important to distinguish between

- An optimal solution
- Proving that a solution is optimal (optimality proof)

Be careful not to overgeneralize

- Finding optimality and proving it can be slow or fast
- One can be fast and not the other

Decision vs Optimization

Every optimization problem has a corresponding decision problem asking if a solution exists with a particular value

Exemple

Finding the shortest path between s and t with a cost c

- Decision Problem: is there a path with cost c?
- Optimality proof: is there a path with cost less than c?

Decision vs Optimization

Optimization problems are often solved by solving a sequence of decision problems

- We find a feasible solution with cost k
- We ask if there is a solution with cost $< k$ and repeat the process
- At the end, we prove optimality, because the last search does not find a solution.

Hard Problems

- We do not know if hard problems can be quickly solved
- Currently, hard problems cannot be efficiently solved (in polynomial time)
- Current solutions may be inefficient (exponential time)

Approaches Covered in the Course: how to find solutions to problems

- Using heuristics (inexact but fast)
- Complete enumeration of combinations (exact but slow)