Preparation CAPES

Difficult problems on graphs
(notions of) NP completeness

Laure Gonnord

http://laure.gonnord.org/pro/teaching/

Laure.Gonnord@univ-lyon1.fr

Université Claude Bernard Lyon1

2017

Lyon 1

Difficult problems on graphs, NP-completeness

Eulerian and Hamiltonian paths

Eulerian/Hamiltonian Path

- An Eulerian path : each edge is seen only once.
- An Hamiltonien path : each node ...

Hamiltonian Tour, one solution

- Extend a path until not possible
- Backtrack one time, and try with another neighbour
- And so on.

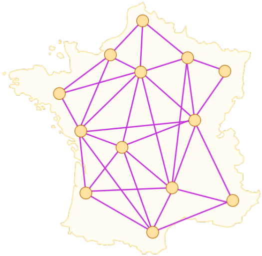
▶ Worst case complexity time : exponential in n

Application 1 : Knight's Tour

Is it possible to make a Knight's tour ? « The knight is placed on the empty board and, moving according to the rules of chess, must visit each square exactly once.»

▶ A variant of the Hamiltonian tour that can be solved in polynomial time.

The Travelling Salesman Problem



with valuated (non oriented) graph.

- Find an hamiltonian cycle of minimum weight

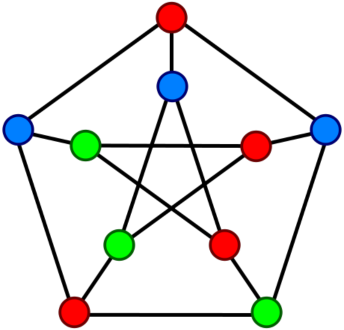
Travelling Salesman Problem - 2

A (non-polynomial) algorithm for this problem :

- Transform into a linear programming problem with integers
- Solve it with the simplex !

► This algorithm is **exponential in the size of the graph**

Graph Coloring Problem



Color with the minimal number of colors !

- Application to the **register allocation** in compilers.
- The sudoku problem (9-coloring of a 81-vertices graph)

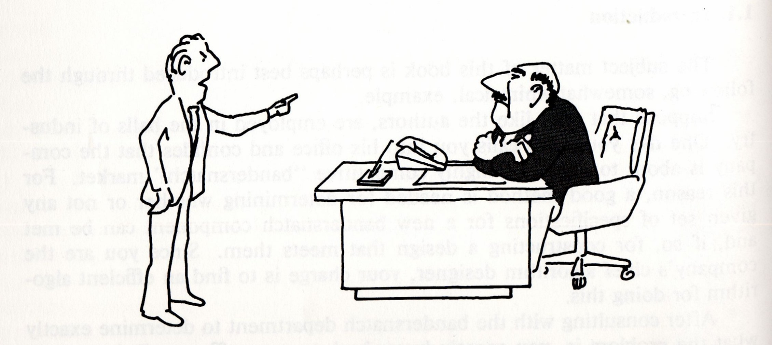
Graph Coloring Problem - 2

Are **you** able to design a polynomial algorithm ?

"I can't find an efficient algorithm, I guess I'm just too dumb."

Graph Coloring Problem - 3

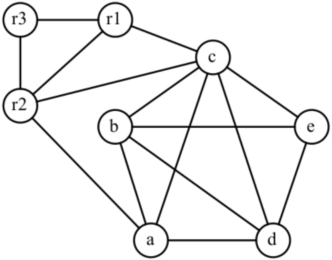
We do not know any polynomial algorithm for this problem.



"I can't find an efficient algorithm, because no such algorithm is possible!"

Graph Coloring Problem - A Polynomial algorithm

An algorithm to color a graph (but without optimising the number) with $\leqslant K$ colors.
Running example :



Kempe's simplification algorithm 1/2

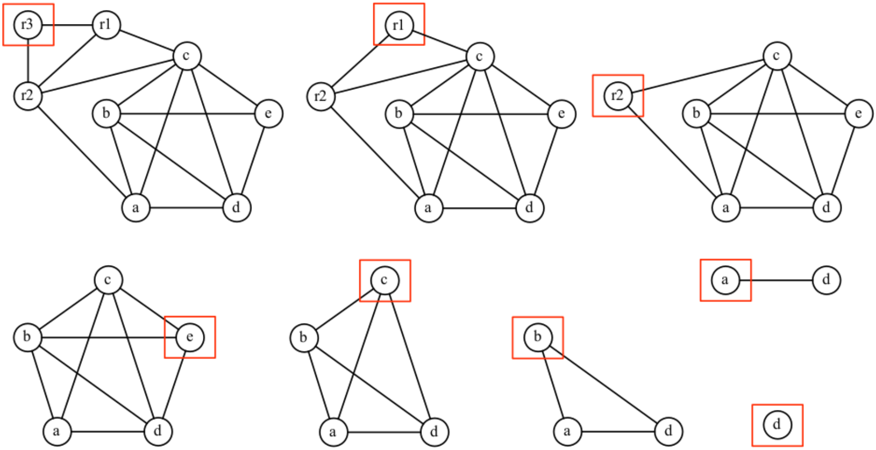
On the interference graph (without coalesce edges) :

Proposition (Kempe 1879)

Suppose the graph contains a node m with fewer than K neighbours. Then if $G' = G \setminus \{m\}$ can be colored, then G can be colored as well.

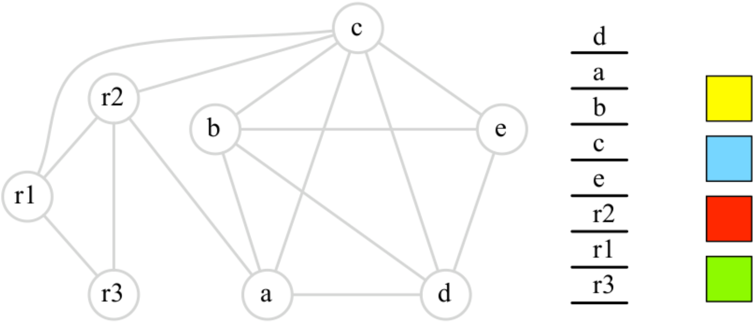
- Pick a low degree node, and remove it, and continue until remove all (the graph is K-colorable) or …

Kempe’s simplification algorithm 2/2

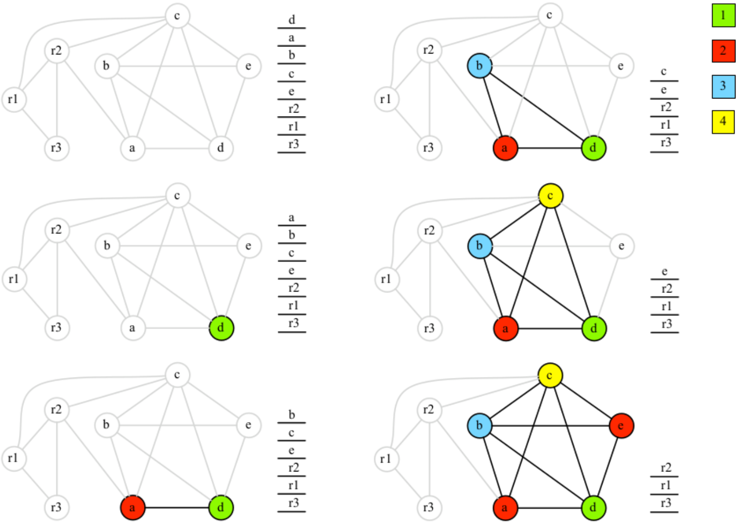


Let's color !

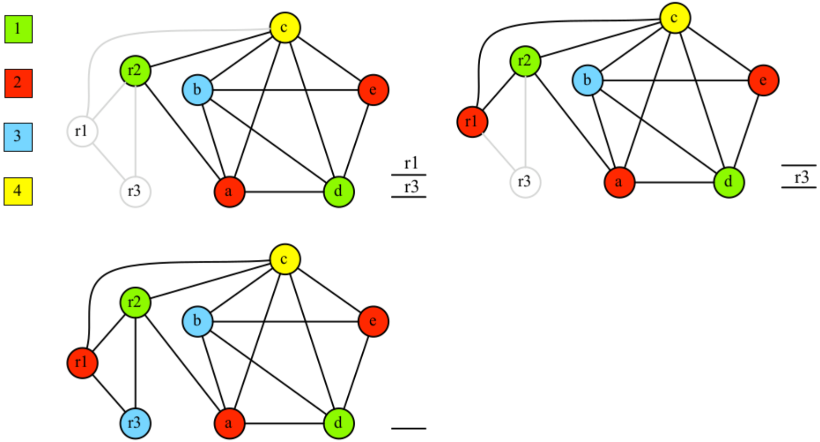
- We assign colors to the nodes greedily, in the reverse order in which nodes are removed from the graph.
- The color of the next node is the first color that is available, *i.e.* not used by any neighbour.



Greedy coloring example 1/2



Greedy coloring example 2/2



- see the Python implementation !

Complexity

The complexity of an implementation / an algorithm is linked to Turing machines :

- The number of steps in the execution of a TM gives the **complexity in time**,
- The number of seen squares in the execution of a TM gives the **complexity in space**.

We can replace the TM by “a C implementation”.

P Problems

P

A **problem** (or a language) belongs to P if there exists a **deterministic** Turing Machine that gives the result in **polynomial time**.

- there is a polynom p such that for all entry x, the Turing machine stops (with the good result) in less than $p(|x|)$ steps.

Example : $L = a^n b^n c^n$, the TM of the previous course checks any $a^i b^i c^i$ in $O(n)$ steps

P Problems

- Integer multiplication
- Eulerian tour (each edge once)
- Linear programming (recall that polynomial diophantine equations are **undecidible**.)
- Primality test

http://www.cs.uu.nl/groups/AD/compl-diaz.pdf

NP Problems

NP

A **problem** (or a language) belongs to NP if there exists a **deterministic** Turing Machine that checks the result in **polynomial time**.

Example : there exists a TM that is able to check if a given path is an hamiltonien tour, in polynomial time.

NP Problems - Examples

- Factoring
- 3-SAT (formulae in CNF with 3 litterals on each term)
- Hamiltonian tour
- 3-colorability

http://www.cs.uu.nl/groups/AD/compl-diaz.pdf

NP vs P - some facts

- If a problem belongs to NP, then there exists an exponential brute-force deterministic algorithm to solve it (easy to prove)
- $P \subseteq NP$ (trivial)

▶ P = NP ? P $\neq$ NP ? Open question ! Problem of the millenium !
Is finding more difficult that verifying ?

NP-complete problems - 1

Given a **problem** :

- if we find a polynomial algorithm ▶ P
- if we find a polynomial time checking algorithm ▶ NP, but we want to know more :

"it it really hard to solve it ?"
"is it worth looking for a better algorithm ?"

NP-complete problems - 2

The key notion is the notion of polynomial reduction

Polynomial reduction of problems

Given two problems P_1 and P_2, P_1 reduces polynomially to P_2 if there exists an f such that :

- f is polynomial
- x_1 solution of P_1 iff $f(x_1)$ solution of P_2.

Example : Hamiltonian circuit is polynomially reductible to TSP.

NP-complete problems - 3

The most difficult problems in NP. All problems in NPC are **computationally equivalent** in the sense that, if one problem is easy, all the problems in the class are easy. Therefore, if one problem in NPC turns out to be in P, then

P=NP

NP-complete problems - 4

Reminder : we want to characterise the most difficult problems in NP.

NP-complete

A problem is said to be NP-complete if :

- It belongs to NP
- All other problems in NP **reduce polynomially** to it.

In other words, if we solve any NP-complete in polynomial time, we have proved $P = NP$ (and we become rich)

NP-complete problems - 4

Problem : how to prove that a problem is NP-complete ?

- The very first one is hard to prove
- Then, we use the following result :

If two problems / languages L_1 and L_2 belong to NP and L_1 is NP-complete, and L_1 reduces polynomially to L_2, then L_2 is NP-complete.

▶ Pick one NP complete problem and try to reduce it to **your** problem.

NP-complete problems - The historical example

- SATISFIABILITY is NP-complete (Cook, 1971)
- SAT reduces polynomially to 3SAT
- 3SAT reduces polynomially to Vertex Cover
- Vertex Cover reduces polynomially to Hamiltonian circuit

► "So"Hamiltonian circuit, then TSP are also **NP-complete** problems

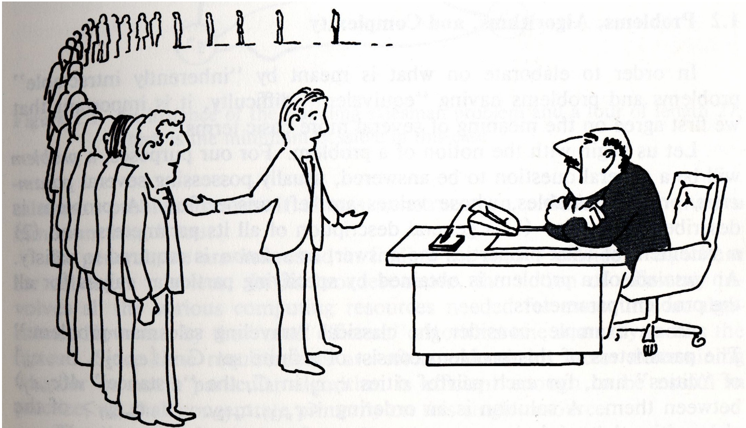
Some NP-complete (common) problems

Some classical examples :

- Hamiltonian cycle and TSP
- Graph Coloring Problem
- Vertex cover
- Knapsack (integer)
- Flow shop problem
- Sudoku

The book : Computers and Intractability : A Guide to the Theory of NP-Completeness. A short list on the french webpage : `http://fr.wikipedia.org/wiki/Liste_de_probl%C3%A8mes_NP-complets`

Conclusion - 1



"I can't find an efficient algorithm, but neither can all these famous people."

Conclusion - 2

Knowing that a given problem is NP complete is just the beginning :

- handle less general classes of inputs
- compute less precise solutions

More docs on graphs

- In french : `http://laure.gonnord.org/site-ens/mim/graphes/cours/cours_graphes.pdf` or type “cours de Théorie des graphes” in Google
- In english : an interactive tutorial here : `http://primes.utm.edu/cgi-bin/caldwell/tutor/graph/intro.`
- (en) The theorical book “Graph Theory”, R. Diestel (electronic version : `http://diestel-graph-theory.com/basic.html`)
- (en) e-book for algorithms : `http://code.google.com/p/graph-theory-algorithms-book/`

More docs on complexity

The Book : computers and intractability, a guide to the theory of NP completeness, by Garey/Johnson