# Unit 1. Introduction to Operational Research

Jordi Villà i Freixa

Universitat de Vic - Universitat Central de Catalunya
Study Abroad. Operations Research

*jordi.villa@uvic.cat*

8/02, 2024

# Preliminary

This course is strongly based on the monography on Operations Research by Carter, Price and Rabadi [1], and in material obtained from different sources (quoted when needed through the slides).

# Learning outcomes

- Learn about the origins and applications of Operations Research
- Understand system modelling principles
- Understand algorithm efficiency and problem complexity
- Contrast between the optimality and practicality
- Learn about software for operations Research
- Introduction to the Python/Colab environment

# Summary

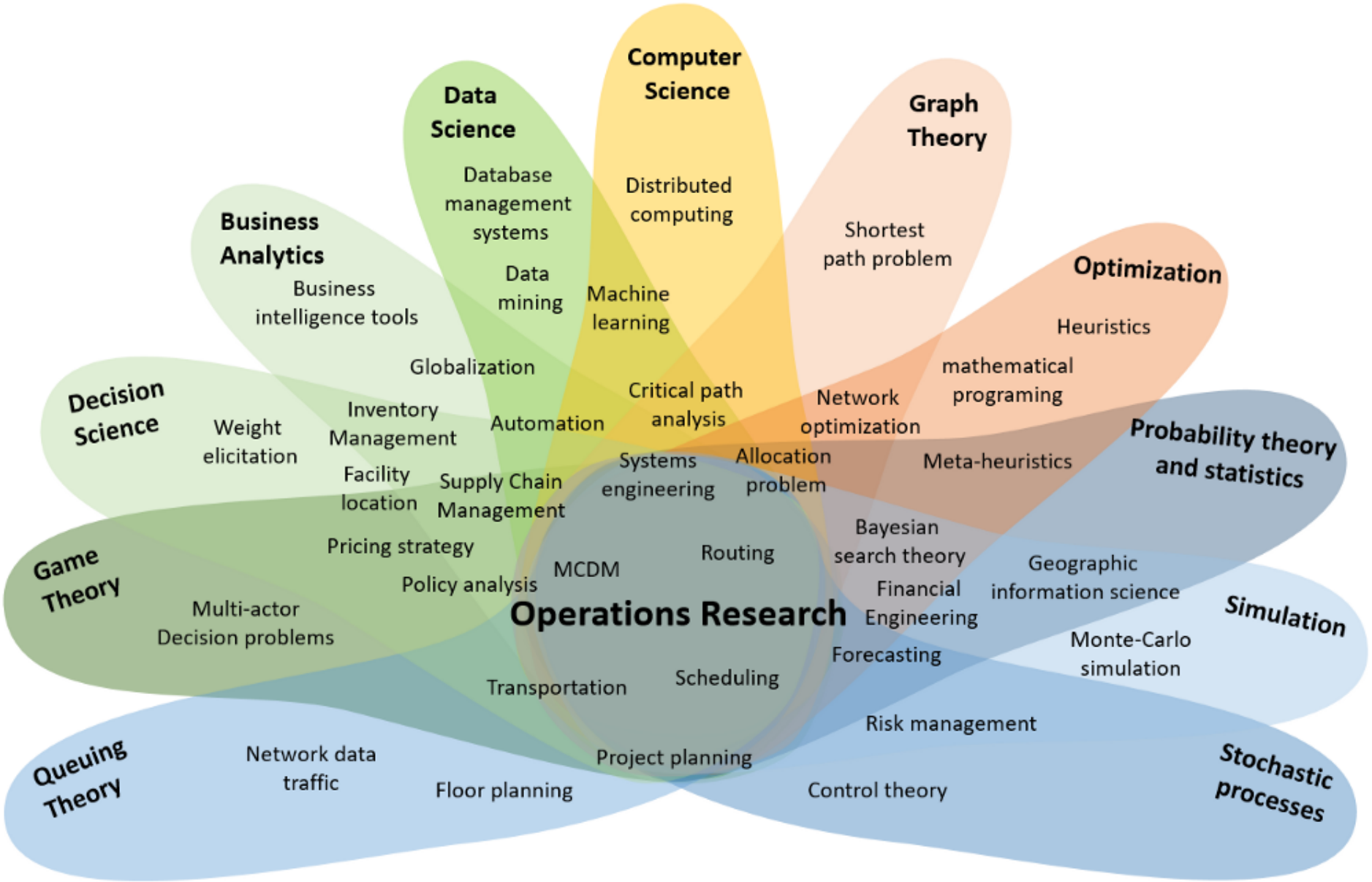
## Operations research (OR)...

The use of quantitative methods to assist analysis and decision-makers in designing, analysizing and improving the performance or operation of systems.

## ...or

Applied math field, where mathematical tools and operators aren't used to investigate mathematics further but rather to analyse and solve problems within the OR domain by designing innovative solution approaches.

Link to a big picture view of OR

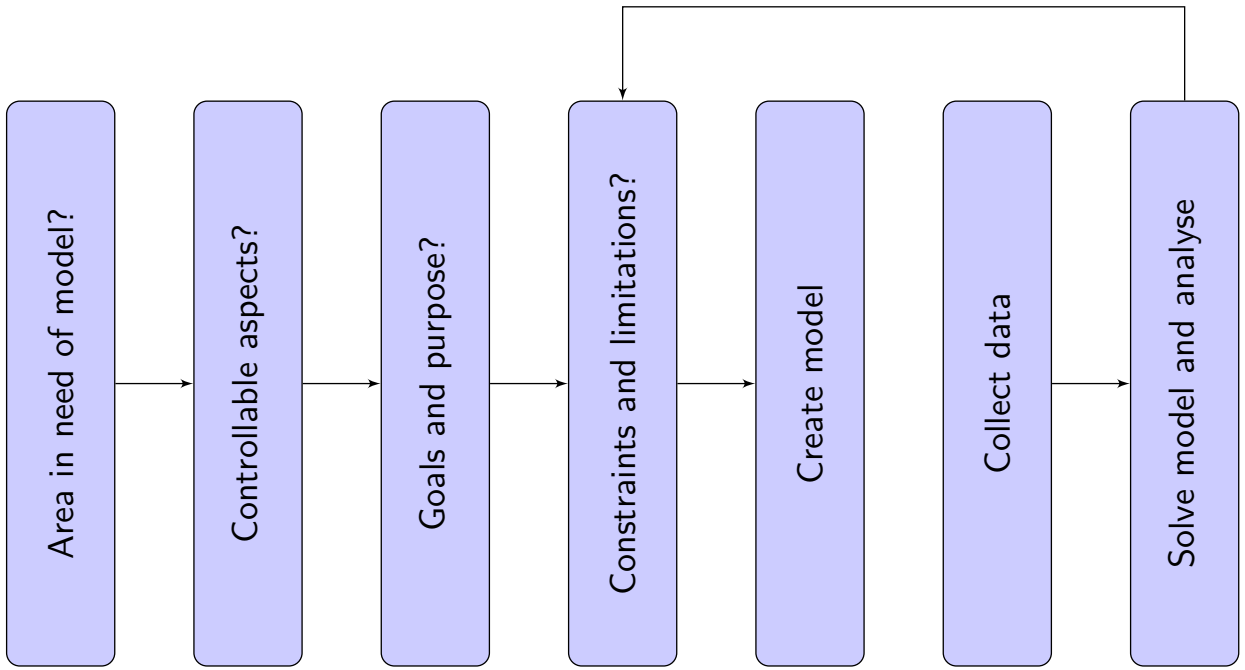

Computer Science
Data Science
Graph Theory
Business Analytics
Optimization
Decision Science
Probability theory and statistics
Game Theory
Simulation
Queuing Theory
Stochastic processes
Operations Research
Database management systems
Distributed computing
Shortest path problem
Business intelligence tools
Data mining
Machine learning
Heuristics
Globalization
mathematical programing
Critical path analysis
Network optimization
Weight elicitation
Inventory Management
Automation
Meta-heuristics
Facility location
Supply Chain Management
Systems engineering
Allocation problem
Pricing strategy
Bayesian search theory
Routing
MCDM
Geographic information science
Policy analysis
Financial Engineering
Multi-actor Decision problems
Monte-Carlo simulation
Forecasting
Transportation
Scheduling
Risk management
Network data traffic
Project planning
Floor planning
Control theory

- OR incorporates analytical tools from many different disciplines, so that they can be applied in a rational way to help decision-makers solve problems and control operations in practice
- OR has been taking shape since the industrial revolution, and most notably after WWII

The term Operations Research was first coined in 1940 by McClosky and Trefthen in a small town, Bowdsey, of the United Kingdom, in a military context (the Battle of England).

# System modelling principles

- A model is a simplified, idealized representation of a real object, a real process, or a real system
- In mathematical models, the building blocks are mathematical structures (equations, inequalities, matrices, functions and operators)
- Building a model:



# The art and science of modeling

- The best model of a system strikes a practical compromise between being realistic vs understandable and computationally tractable
- Detail $\neq$ Accuracy (not all details are correct nor necessary)
- A too detailed model brings complexity to analyze it and exploit it
- Models need to be realistic and simple

*"Everything should be made as simple as possible, but not simpler"*

*Albert Einstein*

# Practical issues in modelling

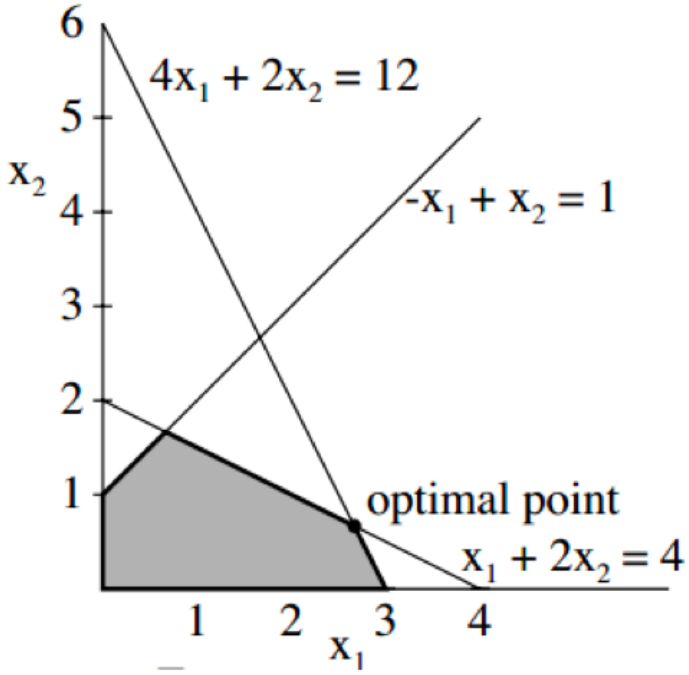
- Does the problem need to be solved?
- What is the *real* problem?
- Would the eventual solution to the problem useful for somebody?
- Would anybody try to implement the solution?
- How much of the analyst's time and cost is worth?
- Are there time and resources available to solve the problem?
- Will the solution create other serious problems for which there is no apparent remedy?

# Problem formulation in Operational Research

**Decision variables** In what are we going to base our decision?

**Objective function** How is our function build with respect to the variables?

**Constraints** What are the limits in our model variables and function?



# Algorithm efficiency and problem complexity

## Algorithm

Sequence of operations that can be carried out in a finite amount of time

- It can be repeated or called recursively but it will eventually terminate.
- Factors influencing the execution time (unrelated to the algorthm itself):
  - the programming language,
  - the programmer's skills,
  - the hardware being used,
  - the task load on the computer system during execution.
- The performance of an algorithm is typically linked to the size of the problem.

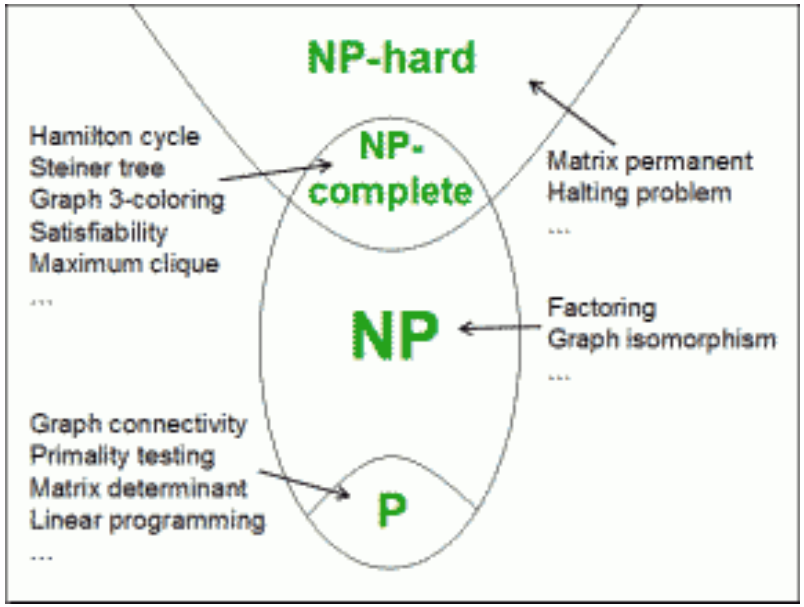
# Two types of problems

## Class P

Problems that can be solved by an algorithm within an amount of computation time proportional to some polynomial function of the problem size.

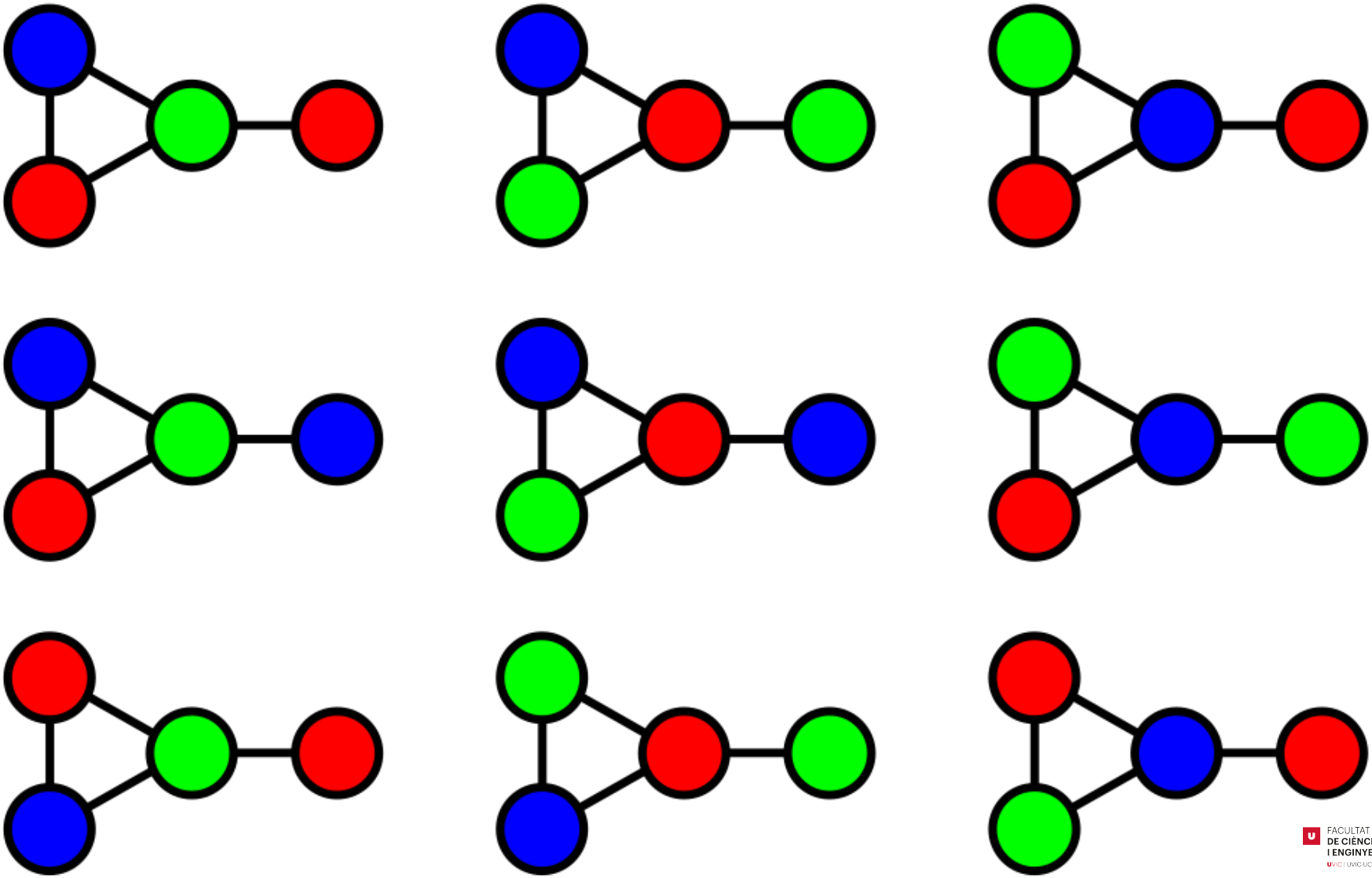
## Class NP

Problems that require computation time proportional to some exponential (or larger) function of the problem size. (The subset sum problem, for example, is $O(2^n \cdot n)$)



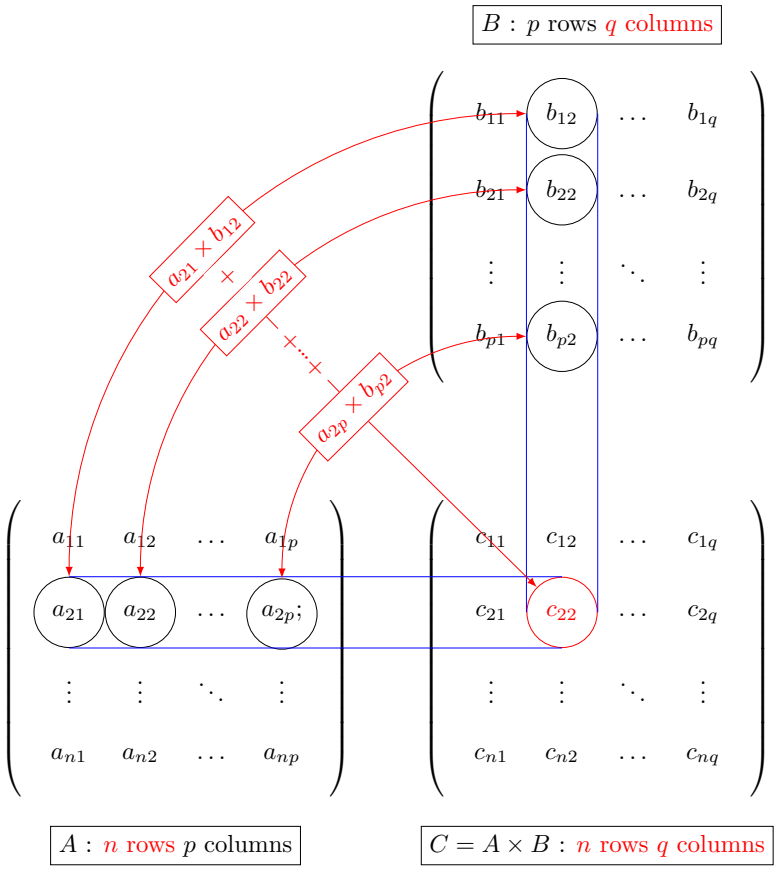

http://www.solipsys.co.uk/new/PVsNP.html

# 3-color problem is NP hard



# Algorithm efficiency

- Performance of algorithm independent of software/hardware/developer skill....? # of computational steps.
- Consider worst case scenario: the largest number of steps that may be necessary.




Multiplying two $n \times n$ matrices, in worst case, involves time proportional to $2n^3$.

# Matrix multiplication is $O(n^3)$

```
# this code is contributed by shivanisinghss2110
# https://www.geeksforgeeks.org/strassens-matrix-multiplication/

def multiply(A, B, C):

        for i in range(N):

                for j in range( N):

                        C[i][j] = 0
                        for k in range(N):

                                C[i][j] += A[i][k]*B[k][j]
```

# $O(n)$ notation

- We say $f(n) = O(g(n))$, and we read as "$f(n)$ is big-O of $g(n)$", if there is some $C$ and $N$ such that

$$|f(n)| \leq Cg(n), \ \forall n \geq N.$$

  In the previous example, multiplying two $n \times n$ matrices costs $2n^3 = O(n^3)$ flops[1]. We say the algorithm is of order 3.
- $n$ denotes the problem size and $g(n)$ is some function of problem size.
- $g(n)$ is the algorithm's worst case step count, as a function of $n$.
- $c$ is the constant of proportionality, and accounbts for extraneous factors affecting execution time (hardware speed, programming style, computer system load during execution)

[1] FLOating Point operationS per second

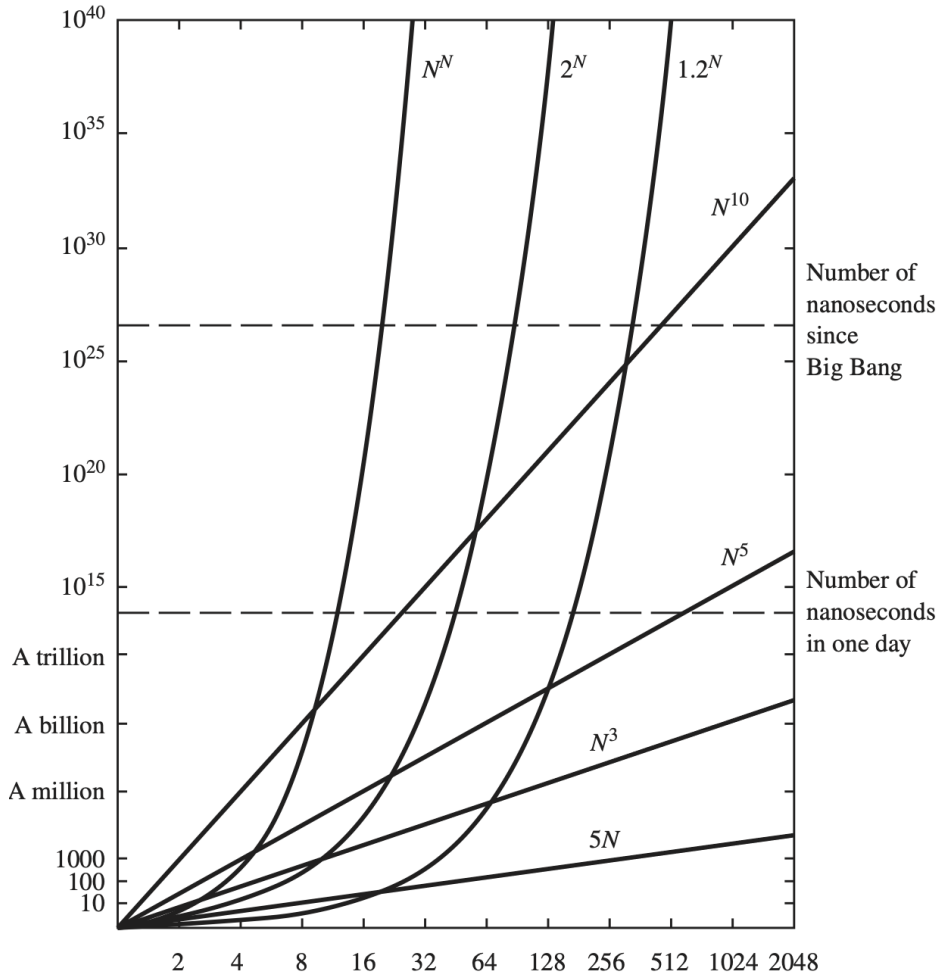
# Different orders, different complexity

| | Function \ N | 20 | 60 | 100 | 300 | 1000 |
|---|---|---|---|---|---|---|
| Polynomial | $5N$ | 100 | 300 | 500 | 1500 | 5000 |
| | $N \times \log_2 N$ | 86 | 354 | 665 | 2469 | 9966 |
| | $N^2$ | 400 | 3600 | 10,000 | 90,000 | 1 million (7 digits) |
| | $N^3$ | 8000 | 216,000 | 1 million (7 digits) | 27 million (8 digits) | 1 billion (10 digits) |
| Exponential | $2^N$ | 1,048,576 | a 19-digit number | a 31-digit number | a 91-digit number | a 302-digit number |
| | $N!$ | a 19-digit number | an 82-digit number | a 161-digit number | a 623-digit number | unimaginably large |
| | $N^N$ | a 27-digit number | a 107-digit number | a 201-digit number | a 744-digit number | unimaginably large |

Extracted from [2]

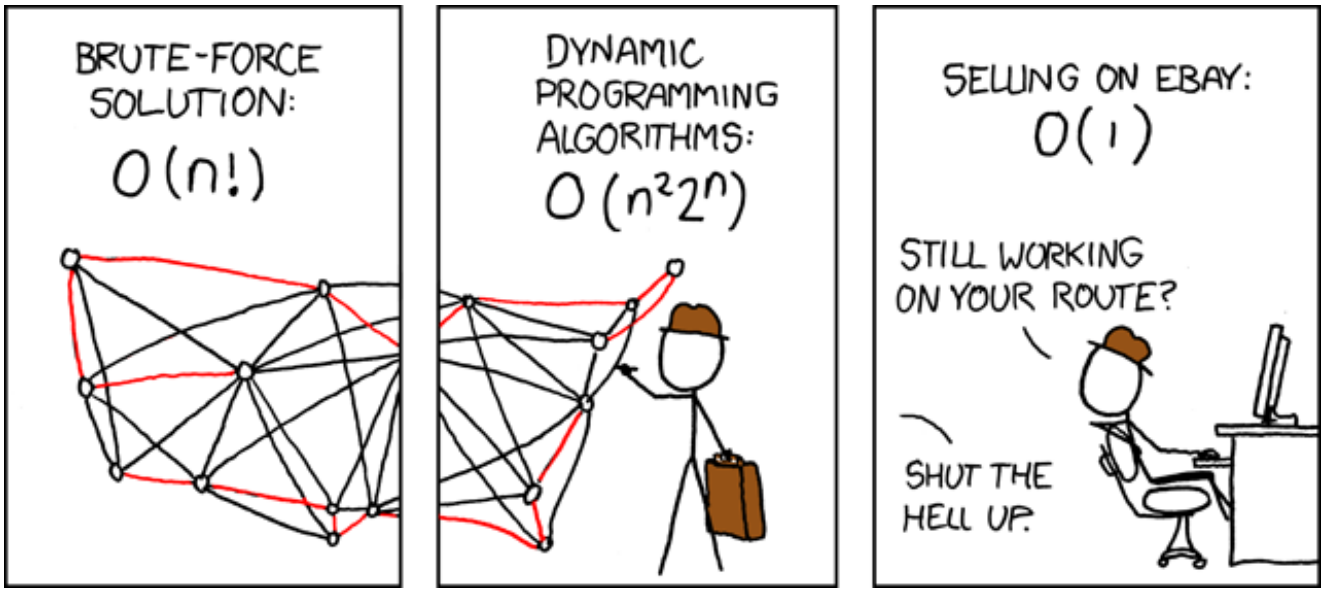
# Growth rate of some functions

For comparison: the number of protons in the known universe has 79 digits; the number of nanoseconds since the Big Bang has 27 digits.[2]



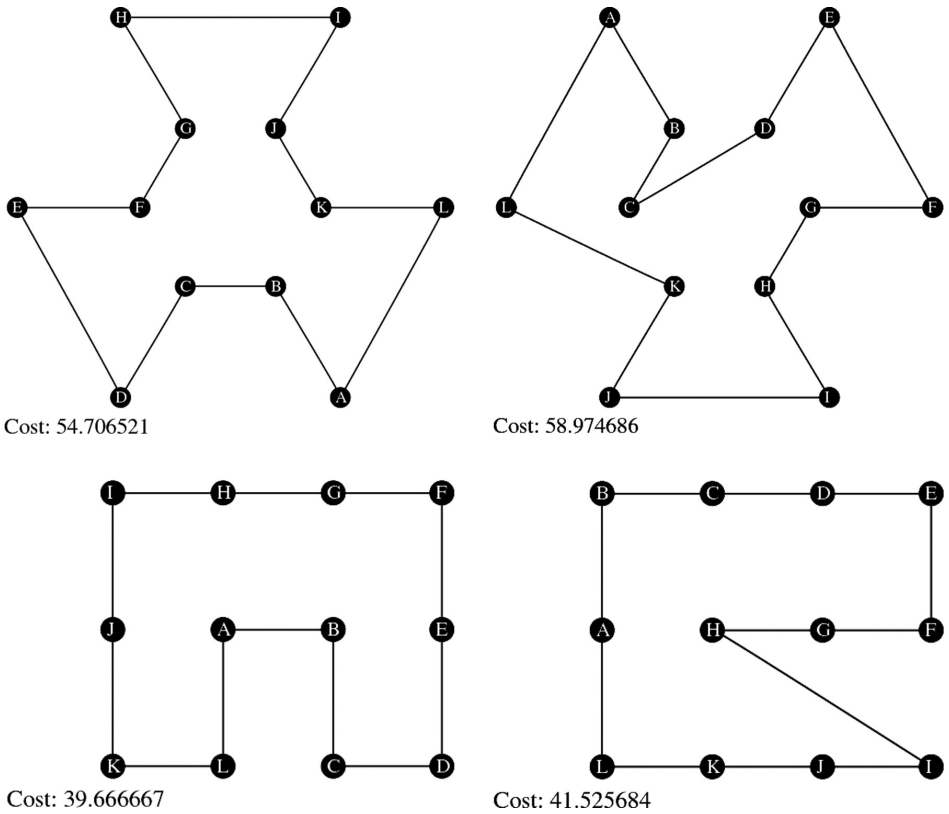
# Search space complexity: the travelling salesman problem (TSP)

- A salesman must visit $n$ cities
- Each city must be visited just once
- Which path should he take to minimize the total distance travelled?




https://xkcd.com/399

# Search space complexity: the travelling salesman problem (TSP)




https://doi.org/10.1073/pnas.0609910104

FACULTAT DE CIÈNCIES, TECNOLOGIA I ENGINYERIES
UVIC | UVIC-UCC

# TSP practical example

Let us assume there are 4 cities and these are the "distances"[2] between them:

| $D_{ij}$ | city A | city B | city C | city D |
|---|---|---|---|---|
| city A | 0 | 5 | 10 | 20 |
| city B | 10 | 0 | 35 | 20 |
| city C | 20 | 10 | 0 | 5 |
| city D | 10 | 5 | 10 | 0 |

Check code at GitHub

[2] Note that the distance is not necessarily identical in both directions

# TSP practical example

- The number of combinations is $(n-1)!$. This is the search space
- Asssuming a computer takes a milisecond to evaluate a solution, How long would it take for the computer to find the optimal solution in our 4 nodes problem? and how long for problems with 10, 20, 50 cities?

# Optimality and Practicality

- We have been trained in mathematics to find exact/perfect solutions. This is not always possible:
  - Models are approximate representations of the real systems
  - Accumulated round-off errors in computers
  - Input data is usually approximated
  - Exponential time algorithms require suboptimal solutions
- "good enough" is not always lowering expectations, as the real world can be extremely complex.

# Software for Operations Research

**Modeling environments** Spreadsheets (Excel, Numbers, Google), AMPL, MPL, LINGO, OPL, AIMMS, SAS/OR, OPTMODEL,GAMS, NEOS[3], ...

**Solvers** Gurobi, Frontline Solver, CPLEX, ...

**Software libraries** Google OR-Tools, COIN-OR, IMSL, ...

We will be using Google colab through the course when possible.

[3] check the TSP demo in NEOS/GAMS

# Summary

# References


[1] Michael W. Carter, Camille C. Price, and Ghaith Rabadi. Operations Research, 2nd Edition. CRC Press.

[2] David Harel, with Yishai Feldman. Algorithmics: the spirit of computing, 3rd Edition. Addison-Wesley.

[3] Ronald L. Rardin. Optimization in Operations Research, 2nd Edition. Pearson.

[4] J. Hefferon. Linear algebra (4th Ed).

[5] K.F. Riley, M.P. Hobson, S.J. Bence. Mathematical Methods for Physics and Engineering (2nd Ed). McGraw Hill.

[6] J. Nocedal, S. J. Wright. Numerical Optimization (2nd Ed). Springer.

[7] Kenneth J. Beers. Numerical methods for chemical engineering: applications in Matlab. Cambridge University Press.

[8] D. Barber. Bayesian reasoning and machine learning. Cambridge University Press.