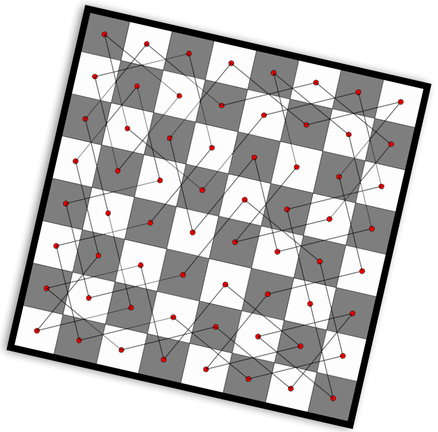
Advanced Algorithms for Optimization

LINFO2266

Introduction

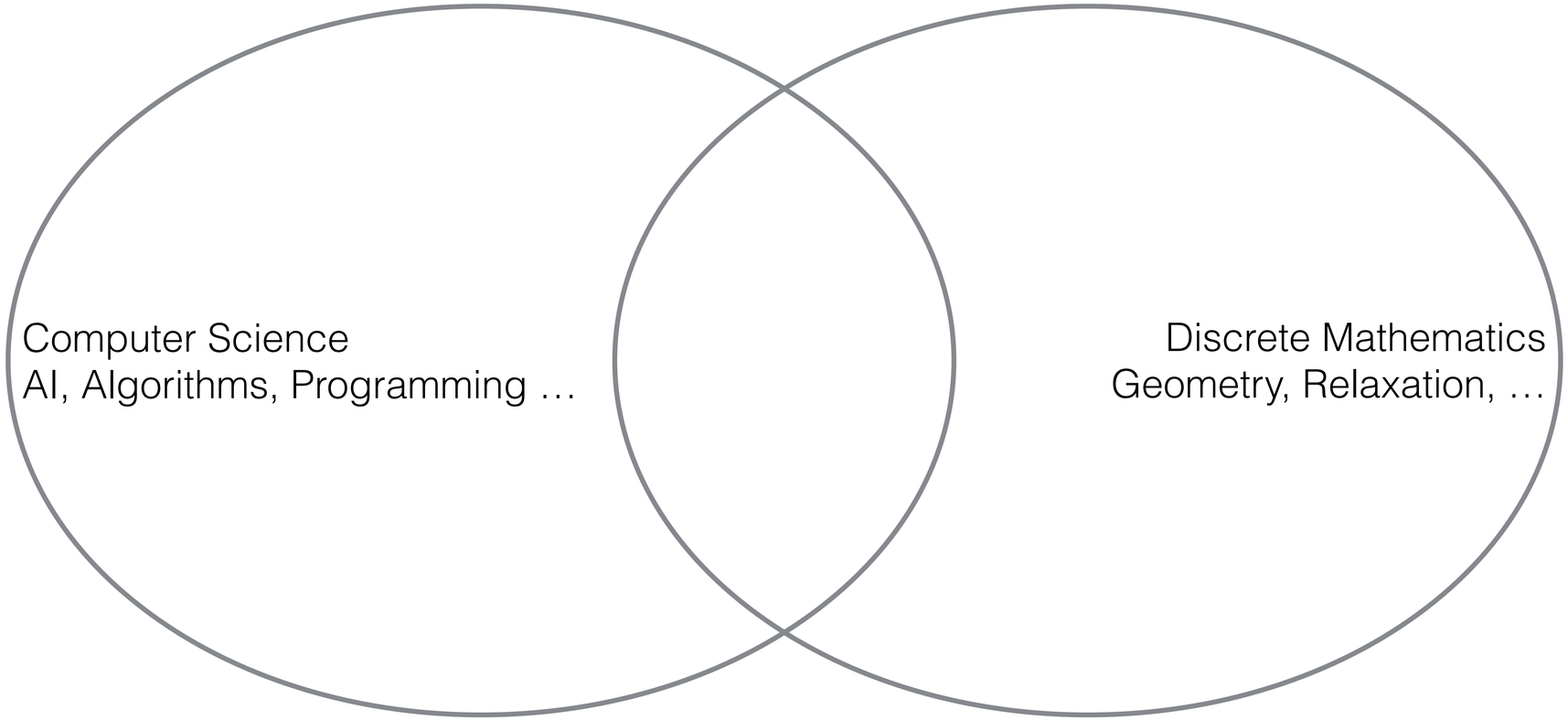
Prof: Pierre Schaus
TAs: Alice Burlats, Amaury Guichard



Optimization is everywhere!

- Logistics (rail, containers, vehicle routing, warehouse)
- Human Resource (nurse rostering, course, etc)
- Scheduling (industry production, tournaments)
- Networking (TE, routing, network design)
- Data-science and Machine Learning (Neural Networks, etc)
- …

A multi-disciplinary field



Since this course is more CS oriented, we will focus more on algorithms and computational aspects

Why study Algorithms for Optimization

- Many large companies (IBM, Google, N-SIDE, Artelis, Informs, Quintiq, Dassault, PSI, OM-Partners, Infrabel, Amazon, etc)
  - The return on investment (ROI) when deploying optimization can be huge, especially when large volumes.
- Because implementing advanced algorithms, able to solve problems that you cannot even solve manually is fun!
- Because whatever you do later, being « algorithm and optimization » minded can be useful and a real asset.

Objective of the course

- Be able to understand and formalize a constrained optimization problem.
- Be able to solve and implement well known discrete optimization algorithms using a well designed and generic approach when possible.
- Be able to understand the advantage and weaknesses of each approaches.

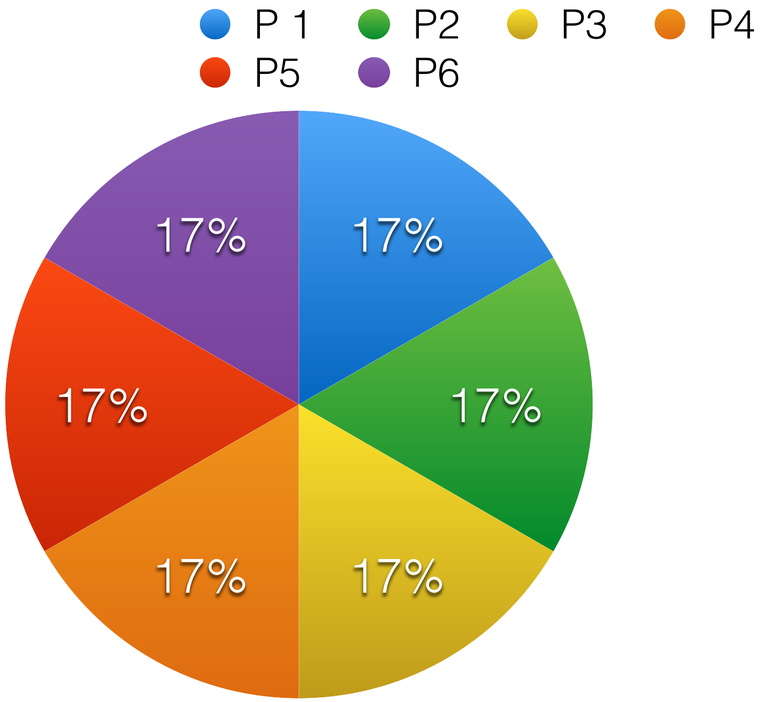
Pedagogy = Project based Learning

- *“I hear and I forget. I see and I remember. I do and I understand”*

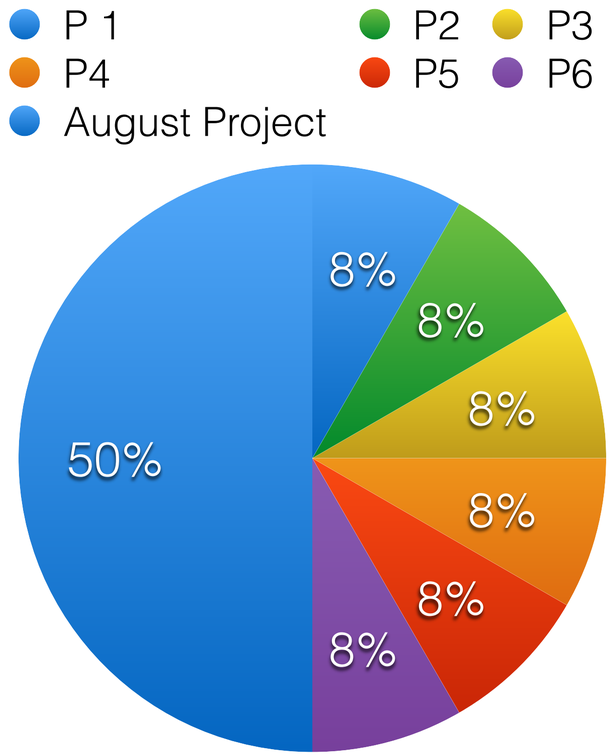
- 6 Projects = 1/6 of the final grades
  - Individual Projects
  - A lot to implement (complex code), tested on inginious
  - A report to write submitted on gradescope and evaluated manually (pdf)

Grading

January



August



Small projects cannot be done again in august , only one chance

Optimization Contest

- Introduced next week

Depending on your ranking, they can earn 0, 1, 2, 3, 4, or 5 points.

If you earn X points, and your grade on the projects is Y (on 20),

Your final grade is $\max\left(Y, \frac{Y}{20} \times (20 - X) + X\right)$

Individual Projects ⚠️

It means that any source code of a project estimated to be
- copied or inspired by the one of another student, or
- copied or inspired by a source code found on the internet or another source,

Will result in a **zero grade for the student at all the projects**.

The same consequences will hold for a student that voluntarily shares his code or make available to other students.

What is allowed

- Students are encouraged to exchange insightful ideas and observations.
- But they should never share code for solving the exercises.
- Your GitHub repository must ramain closed.

Deadlines for the projects

Week	Wed. 4h15PM	Lecture	Start Project (16h)	Deadline Project (14h)
1	18/09/2024	Intro + Dynamic Programming	P1: Dynamic Programming	
2	25/09/2024	Optimization Competition	Competition	
3	02/10/2024	Branch and Bound	P2: Branch and Bound	P1: Dynamic Programming
4	09/10/2024	Lagrangian Relaxation		
5	16/10/2024	Linear Programming	P3: LP and Flows	P2: Branch and Bound
6	23/10/2024	Network Flows		
7	30/10/2024	Smart-week		P3: LP and Flows
8	06/11/2024	Local Search	P4: Local Search	
9	13/11/2024	Optimization Contest Consultancy		
10	20/11/2024	Constraint Programming	P5: Constraint Programming	P4: Local Search
11	27/11/2024	Optimization Contest Consultancy		
12	04/12/2024	MDD based discrete optimization	P6: MDD Optimization	P5: Constraint Programming
13	11/12/2024	Optimization Contest Consultancy		
14	18/12/2024			P6: MDD Optimization
15	20/12/2024			Competition

Generative AI: What is allowed

Can be used but **academic honesty** and originality remain paramount.

- Original Work: Your submission should be primarily your own work. Directly copying and pasting solutions from AI outputs without understanding or modification is discouraged.

- Source Indication: Whenever you use generative AI to assist in your assignment, you are required to indicate so by providing a brief comment in your code on how the AI was used. For example:

```
# Used AI to suggest optimization for this loop.
for i in range(10): ...
```

Failure to adhere to these guidelines => academic penalties.

We may always request an interview about the projects!

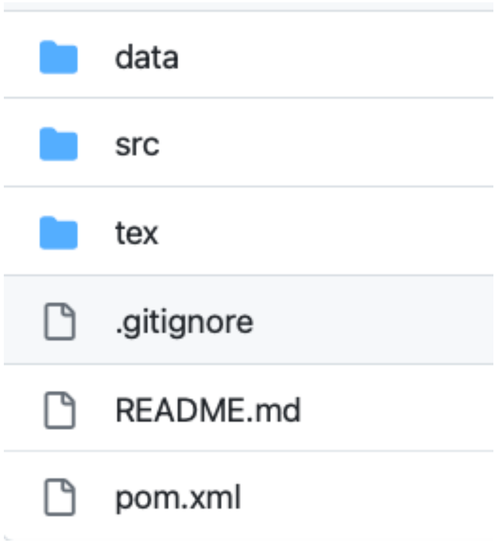
The repository of the course

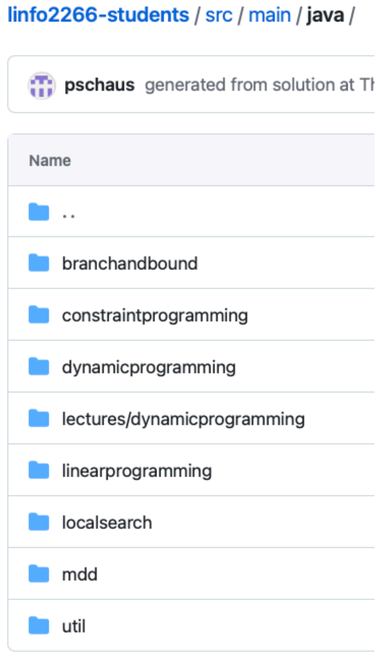
- Website with slides and project statements
  - https://linfo2266.readthedocs.io
- Inginious:
  - https://inginious.info.ucl.ac.be/course/INGI2266
- Github with the projects and latex templates
  - https://github.com/pschaus/linfo2266-students
- Gradescope for report submissions
  - https://www.gradescope.com/courses/869042

Submission and projects, how it works ? 1/4

- Step 1: read and approve the “Honor Code Agreement Contract”.

- Step 2: Create your repository, you simply give your GitHub id and you are then invited to be the collaborator of a project with some source-code.

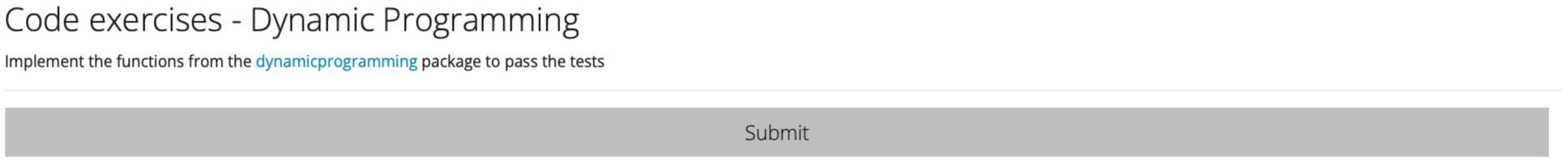




Submission and projects, how it works ? 2/4

- Step 3: You code your solution to the current project (commit frequently to your repository)

- Step 4: Press “submit” button on the task of the current project



Submission and projects, how it works ? 3/4

- Inginious will then:
  - Pull your code (with your solution)
  - Run tests (some of them are hidden)
  - Grade your solution
- If happy with the grading, stop there, otherwise work harder.
- Not yet finished: The pdf report to submit on https://www.gradescope.com/courses/869042 Entry code: 7E5W86

Submission and projects, how it works ? 4/4

- Small questions (experiment, theory, etc). Answer (Ipad+Scan manually written are ok) or edit the PDF or do it from the latex source. Then upload the PDF.

First & last name	
NOMA UCLouvain	

LINFO2266: Advanced Algorithms for Optimization

Project 1: Dynamic Programming

Exercise 1 Modeling the Traveling Salesman Problem

The Traveling Salesman Problem is a famous combinatorial optimization problem where a salesman has to find the shortest tour visiting all cities from a given set $N = \{1, \dots, n\}$. The salesman can travel between every pair of cities $i, j \in N$, which are separated by a distance d_{ij}. Find a dynamic programming model for the Traveling Salesman Problem.

1. What information is contained in the states of the model?

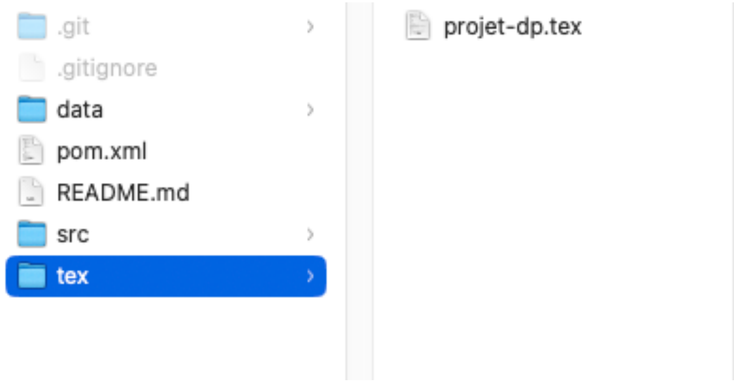
2. Give the Bellman recurrence equation that computes the optimal value for each state of the model. You can assume that the salesman starts at city 1.

3. Give the base case(s) of the recurrence, and the case that gives the optimal solution of the problem.

4. What is the space complexity of a program computing the solution of your dynamic programming model, with respect to the number of cities n. Why?

5. What about the time complexity? Explain why.

1

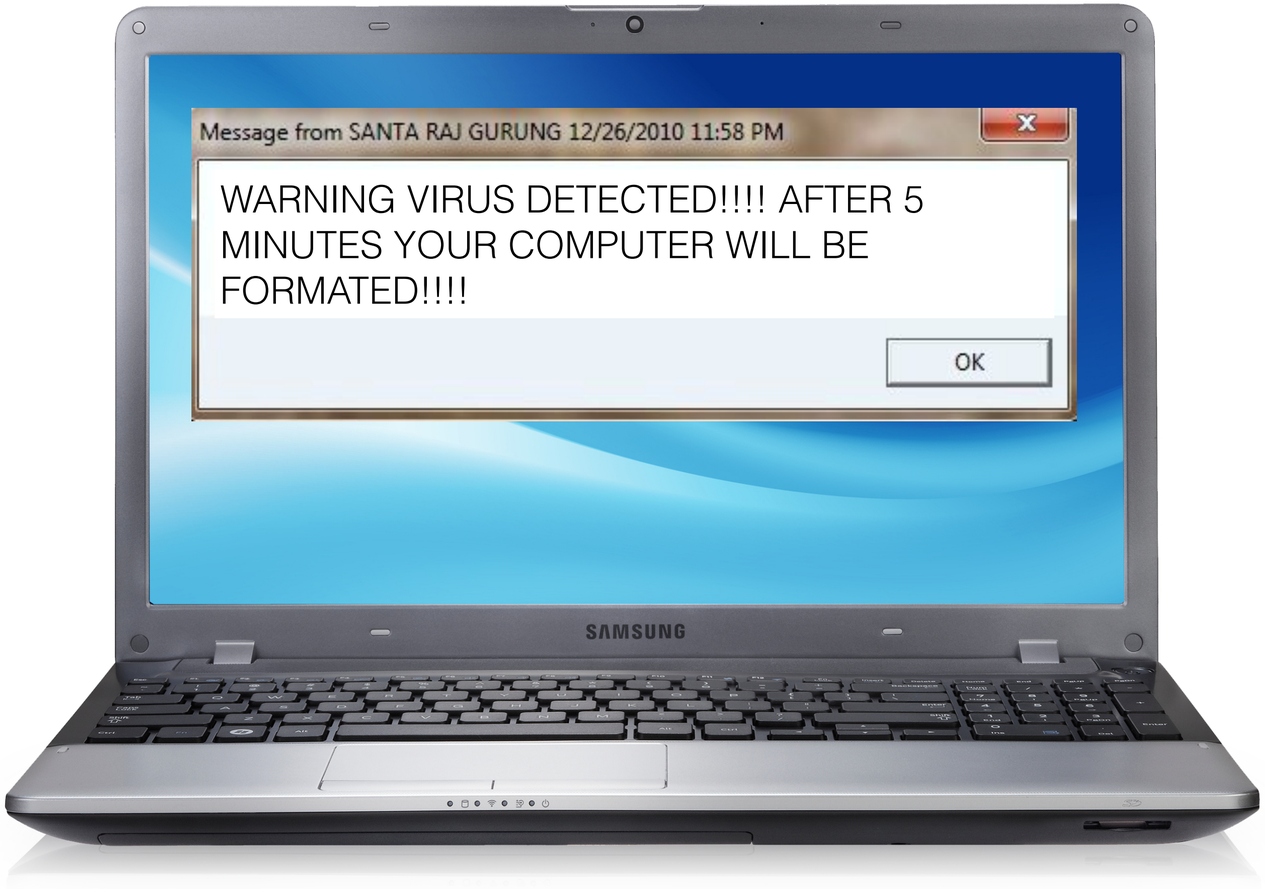


Overview of some optimization techniques

- Motivating example

- In order to show you how I generally decide what approach I should use to solve an optimization problem

Motivation



10 GB

The problem


1 ❤
2GB


1 ❤
2GB


1 ❤
2GB

10 ❤
2GB

10 ❤
2GB


13 ❤
8GB


1 ❤
3GB

10 GB

?

Which files to save
to maximize my value?

Knapsack Problem

- the set of items: I
- is item i selected: $x_i \in \{0, 1\}$
- Objective: Maximize $\sum_{i \in I} v_i x_i$
- Constraints: $\sum_{i \in I} w_i x_i \leq C$

Maximize value of selected items

Under capacity constraint

Reminder Calculability: P, NP and NP-Complete

- **P** is the class of problems with efficient algorithms for solving them (polinomial time in the size of the input)

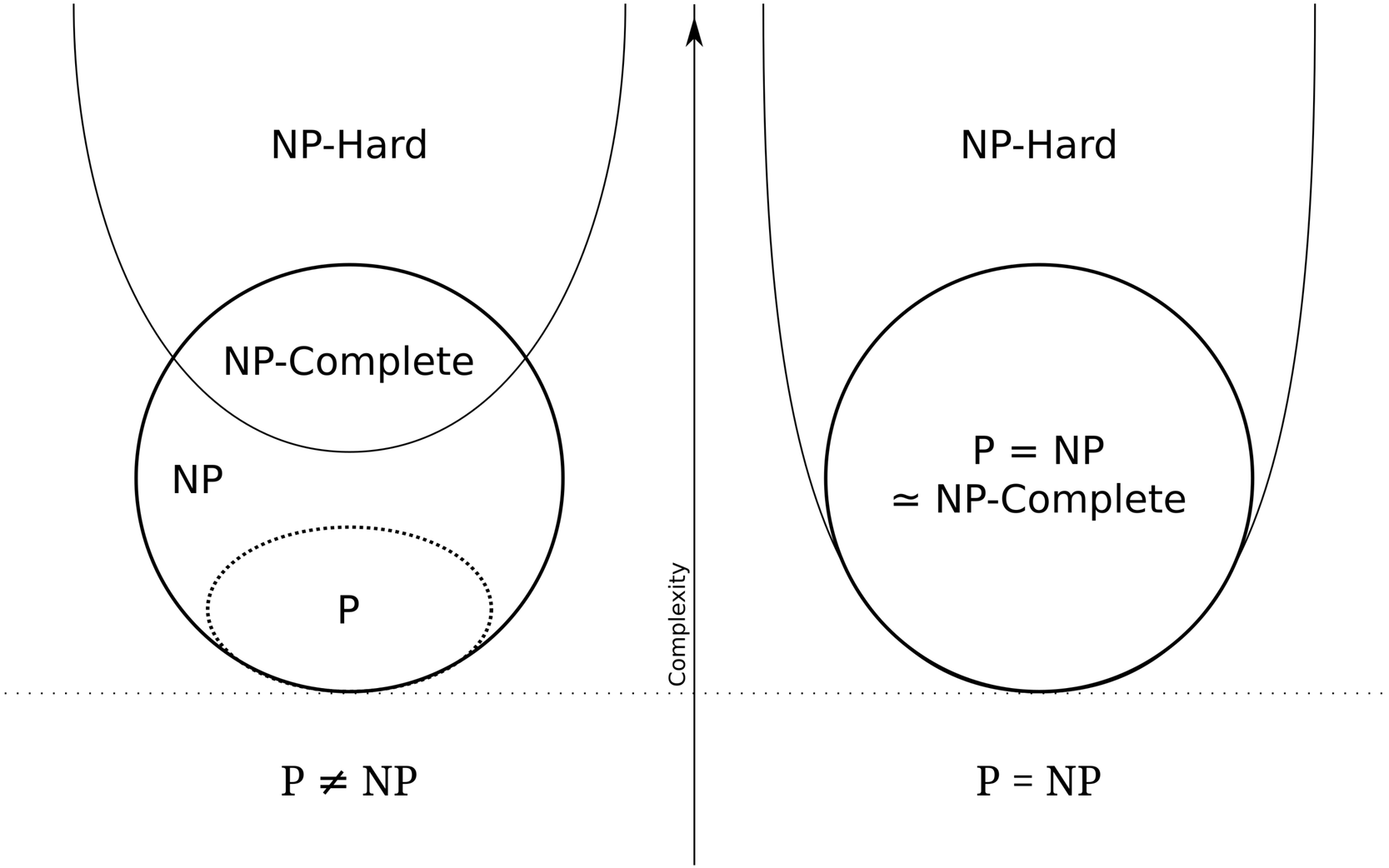
- **NP** is the class of problems with efficient algorithm for (in)validating a candidate solution

- In NP, there are problems for which there no known algorithm for solving them efficiently (TSP for instance). We call those problems **NP-complete**. This is clique of problems because there is a know polynomial time reduction between any pairs of these.

- This is an open-question whether P = NP ?

- Nice read: Fortnow, L. (2009). The status of the P versus NP problem. *Communications of the ACM*, *52*(9), 78-86.

Summary


NP-Hard
NP-Complete
NP
P
Complexity
NP-Hard
P = NP
≃ NP-Complete
P ≠ NP
P = NP

Reminder Reduction

Suppose there is a polynomial-time reduction from problem A to problem B.

It this statement true ?

- If A has no polynomial-time algorithm then neither does B

Reminder Reduction

Suppose there is a polynomial-time reduction from problem A to problem B.

It this statement true ?

- If A has no polynomial-time algorithm then neither does B ✅
- *Suppose that B would have an polynomial time algorithm, then it means that you could also solve A efficiently (using the reduced problem), which contradicts the hypothesis.*
-
  - A is the problem of finding an Hamiltonian Cycle in a graph (a path in a directed graph that visits each vertex exactly once)
  - B is problem of sorting numbers.
  -

Reminder Reduction

Suppose there is a polynomial-time reduction from problem A to problem B.

It this statement true ?

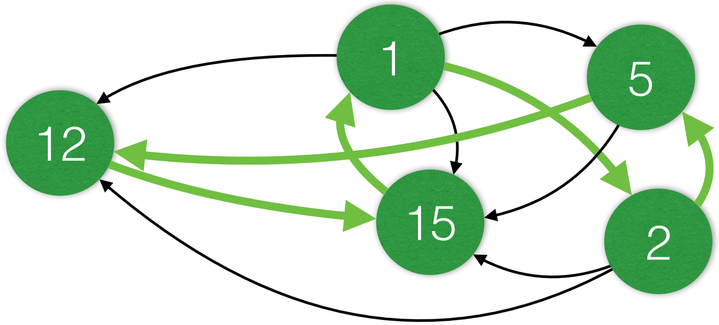
- If B has no polynomial-time algorithm then neither does A

Reminder Reduction

Suppose there is a polynomial-time reduction from problem A to problem B.

It this statement true ?

- If B has no polynomial-time algorithm then neither does A ❌
  - Counter example:
    - A is problem of sorting numbers (efficient algorithms exists for sorting)
    - B is the problem of finding an Hamiltonian Cycle in a graph (a path in a directed graph that visits each vertex exactly once), known to be NP-Complete
    - Reduction: add an arc from i to j if i < j and from the largest to the smallest, a Hamiltonian cycle in this graph corresponds to visiting number in increasing order



Reminder Reduction

Suppose there is a polynomial-time reduction from problem A to problem B.

It this statement true ?

- If A is NP-hard and B has a polynomial-time algorithm then P=NP

Reminder Calculability

Suppose there is a polynomial-time reduction from problem A to problem B.

It this statement true ?

- If A is NP-hard and B has a polynomial-time algorithm then P=NP ✅

First Question (important): Is this problem NP-Hard?

- Yes if the related decision problem is NP-Complete

$$\sum_{i \in I} v_i x_i \geq V$$

- $$\sum_{i \in I} w_i x_i \leq C$$

$$x_i \in \{0, 1\}$$

$$\texttt{Natural numbers } c_1, \ldots, c_n, K.$$

$$\texttt{Find } S \subseteq \{1, \ldots, n\} \texttt{ s.t. } \sum_{j \in S} c_j = K$$

First Question (important): Is this problem NP-Hard?

- Yes if the related decision problem is NP-Complete

$$\sum_{i \in I} v_i x_i \geq V$$

$$\sum_{i \in I} w_i x_i \leq C$$

$$x_i \in \{0, 1\}$$

- We know *subset sum* is NP-Complete (a quite similar problem)

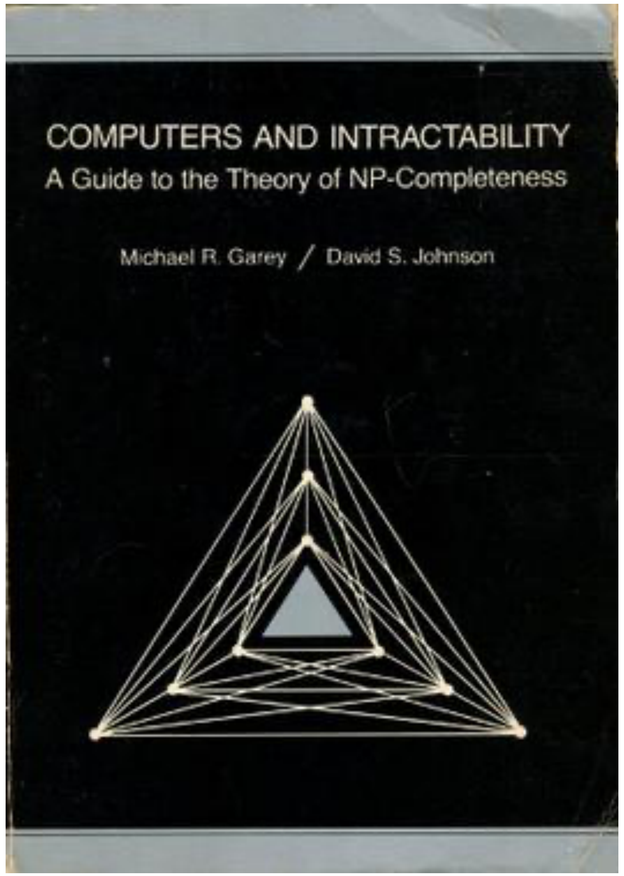
$$\texttt{Natural numbers } c_1, \ldots, c_n, K.$$

$$\texttt{Find } S \subseteq \{1, \ldots, n\} \texttt{ s.t. } \sum_{j \in S} c_j = K$$

- Exercise: Find a (polynomial) **reduction** from subset sum to knapsack (if you can solve knapsack efficiently, then you can solve subset sum efficiently).

A data-base of NP-Complete problems

This book if is used a lot to make reductions for new problems:

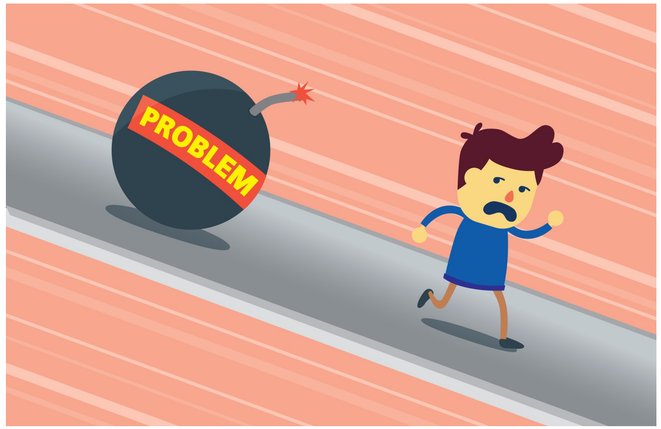


This website is also useful https://www.csc.kth.se/tcs/compendium/

A first bad news

- Most of the real-life constrained optimization problems are NP-hard.

- Does-it mean that we should run away and not solve them ?



- Of course not (at least I hope you will be convinced about that after having followed this course)

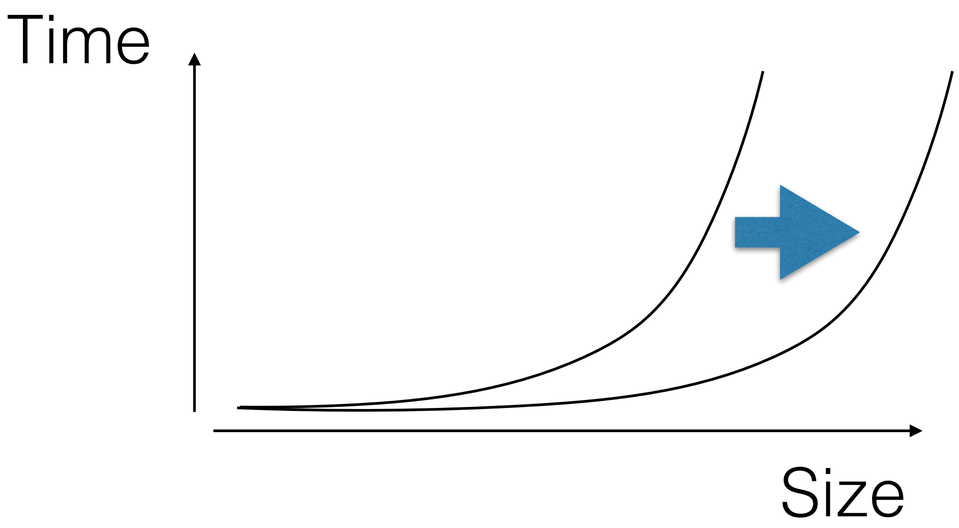
Second question: what size are the instances ?

- Very small : A naive approach (like brute force) can easily find the best possible solution

- Try every possible solutions
  - n items, 2^n solutions
  - n = 50, 1 ms to test one solution, > 30.000 years

Second question: what size are the instances ?

- Medium: I want to find the best solution

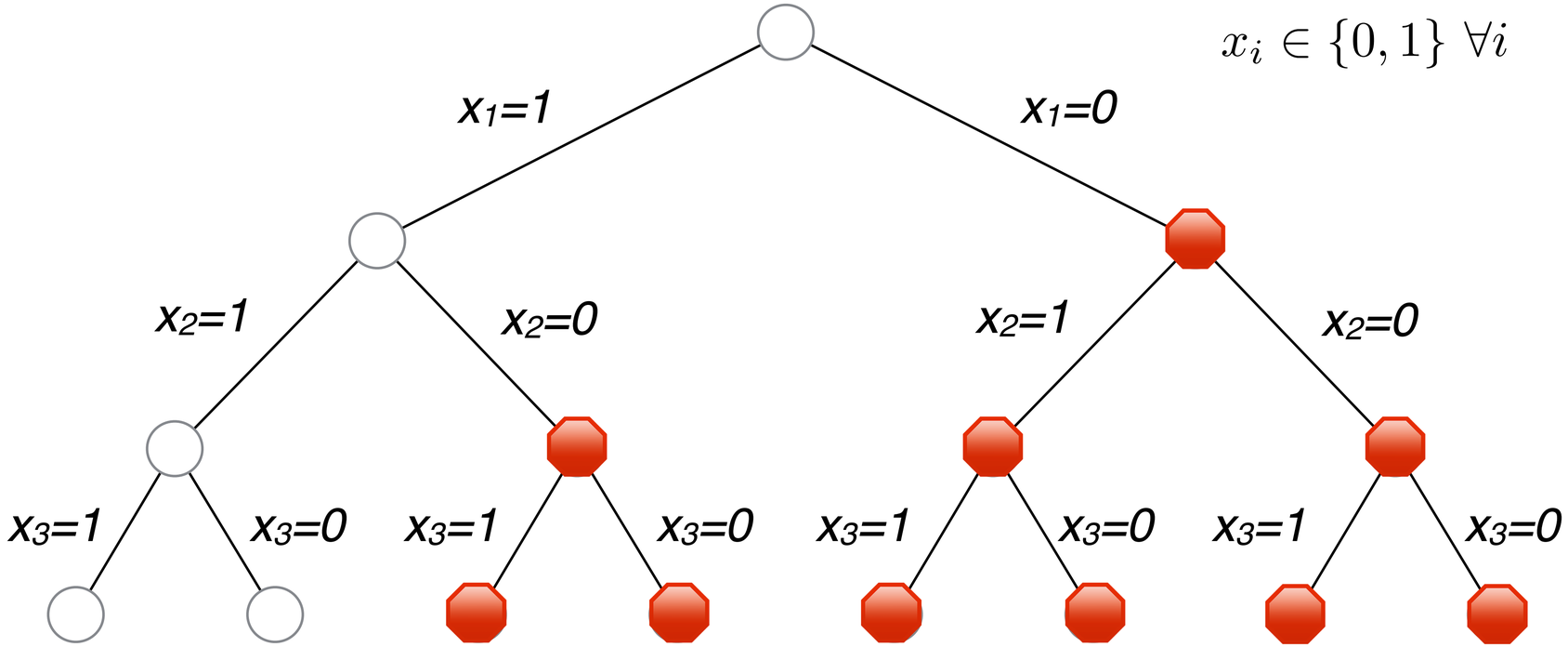


- Objective: shift the exponential behavior as much as possible to the right

How to shift

- Almost all the exact approaches explore a search tree

$$\texttt{maximize } 28x_1 + 30x_2 + 20x_3$$
$$\texttt{subject to } 4x_1 + 6x_2 + 4x_3 \leq 9$$
$$x_i \in \{0, 1\}\ \forall i$$



- The goal is then to **prune this search tree** as much as possible to spare time (lower-bounding, dominance, etc)

The role of relaxation

- I have a solution to my knapsack (maximisation) problem with objective O
- How far can it be from the optimal solution O^* ?
- Impossible to know, unless we have an upper bound UB.
- Then I can guarantee that I am at most $\frac{(O - UB)}{UB}$ suboptimal
- **Finding quickly good lower/upper-bounds** is **THE KEY** in optimization
- Many techniques can help us: Linear Programming, Lagrangian relaxation, Dedicated Bounding algorithms, etc

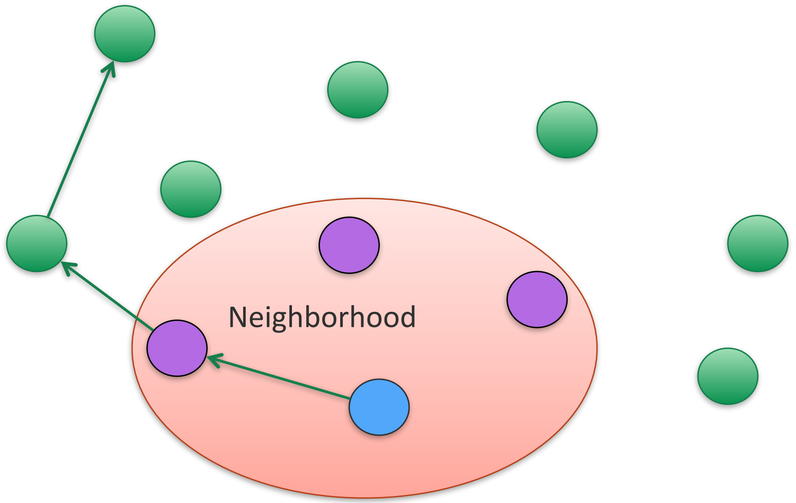
How to find relaxations ?

- By relaxing some constraints, making the problem easier to solve

Second question: what size are the instances ?

- Medium: but you prefer to find a good enough solution quickly (within some time-limit)

- Local Search methods are generally the one we prefer in this case. Gradually apply small (improving) modifications to your solution. You can stop at any time. You can put a lot of intelligence in the exploration and the moves (as we will see in a few lectures)



Second question: what size are the instances ?

- Large: Forget about proving optimality. You can generally only apply a greedy heuristic and a few local search moves depending on the the time you have.

- Example: Knapsack, what would you do as a greedy heuristic ?

$$maximize \sum_{i \in I} v_i x_i$$

$$subject\ to \sum_{i \in I} w_i x_i \leq C$$

$$x_i \in \{0, 1\}$$

Other consideration 💰

- What is the money budget for solving the problem
  - Do you have money to buy 10K $ licence of a mixed integer programming solver ?
  - How much time do you have to develop a solution for your problem (usually also a cost budget that depends on the return).

- Funny (but important) fact: Your (future) salary may cost (much) more than a licence price. Developing a solution using an existing optimization solver might be the best solution for your employer. Commercial solvers are very good today (Gurobi, Cplex, CP-Optimizer, Hexaly, etc). Some company have an open-source model like Cosling

Some Optimization Projects

- We are working on at the INGI department with my research team

Computer Networks

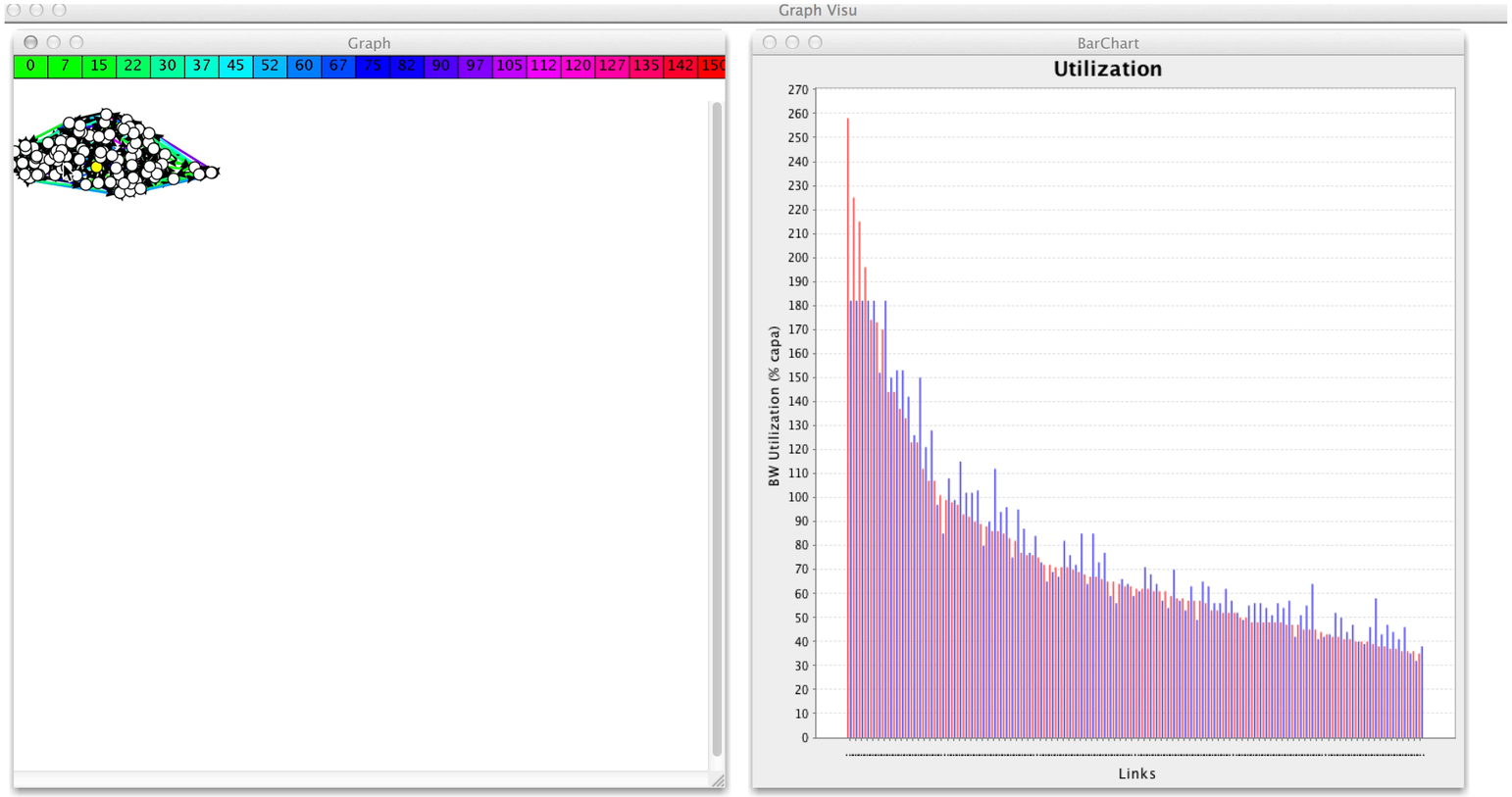
- Traffic demand between every pair of nodes in a network (steady state, in MB per second).

- Traffic follow the shortest-path, you remember 

?

- Two Traffic Engineering problems:
  - Find weights to put on each link to avoid congestion (accumulated traffic > capacity of the link).
  - Modify the paths (just a little, by introducing waypoints deviation) to avoid congestion

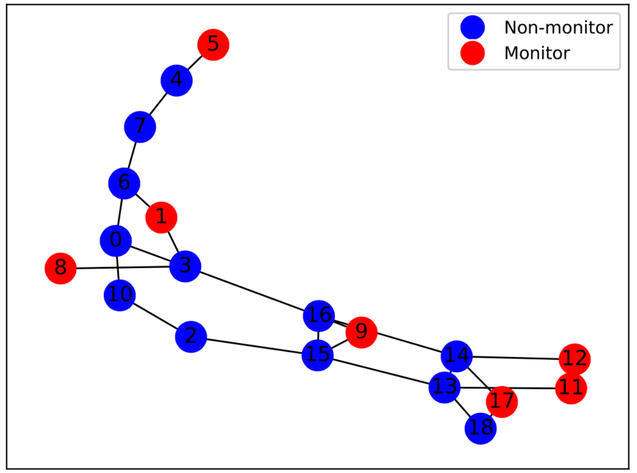
Hartert, R., Vissicchio, S., Schaus, P., Bonaventure, O., Filsfils, C., Telkamp, T., & Francois, P. (2015). A declarative and expressive approach to control forwarding paths in carrier-grade networks. *ACM SIGCOMM computer communication review*, *45*(4), 15-28.


Graph Visu
Graph
0 7 15 22 30 37 45 52 60 67 75 82 90 97 105 112 120 127 135 142
BarChart
Utilization
BW Utilization (% capa)
Links

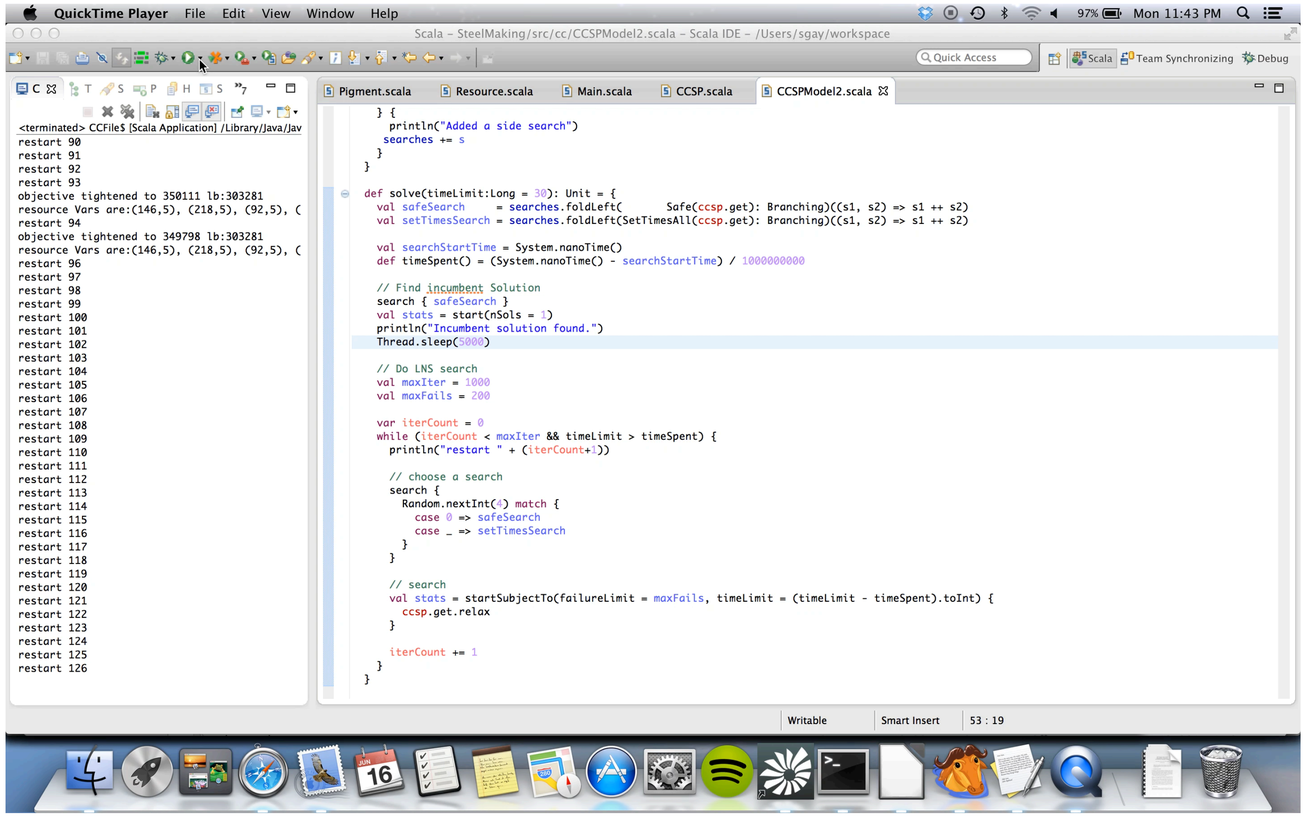
Computer Network Monitoring

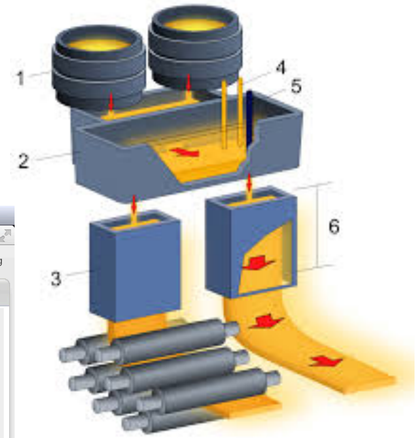
- Bad things can happen open a computer network.
- We want to detect it as soon as possible.
- Idea: introduce monitors exchanging between them. Whenever a path is broken, we know something is broken.
- Problem: Where do we place them so that there is a minimal number of them

A. Burlat, C. Pelsser, P. Schaus (2024). Boolean Network Tomography: CP and ILP Models for Effective Failure Detection and Diagnosis}. CPAIOR 2024



Steel Production Planning





Gay, S., Schaus, P., & De Smedt, V. (2014). Continuous casting scheduling with constraint programming. In *Principles and Practice of Constraint Programming: 20th International Conference, CP 2014, L*
Proceedings 20 (pp. 831-845). Springer International Publishing

Rostering of a hospitality management school

•Create the roster for groups of students over the semester. For each group, each week (AM/PM), we must decide the course (atelier) to schedule

Horizontal constraints:

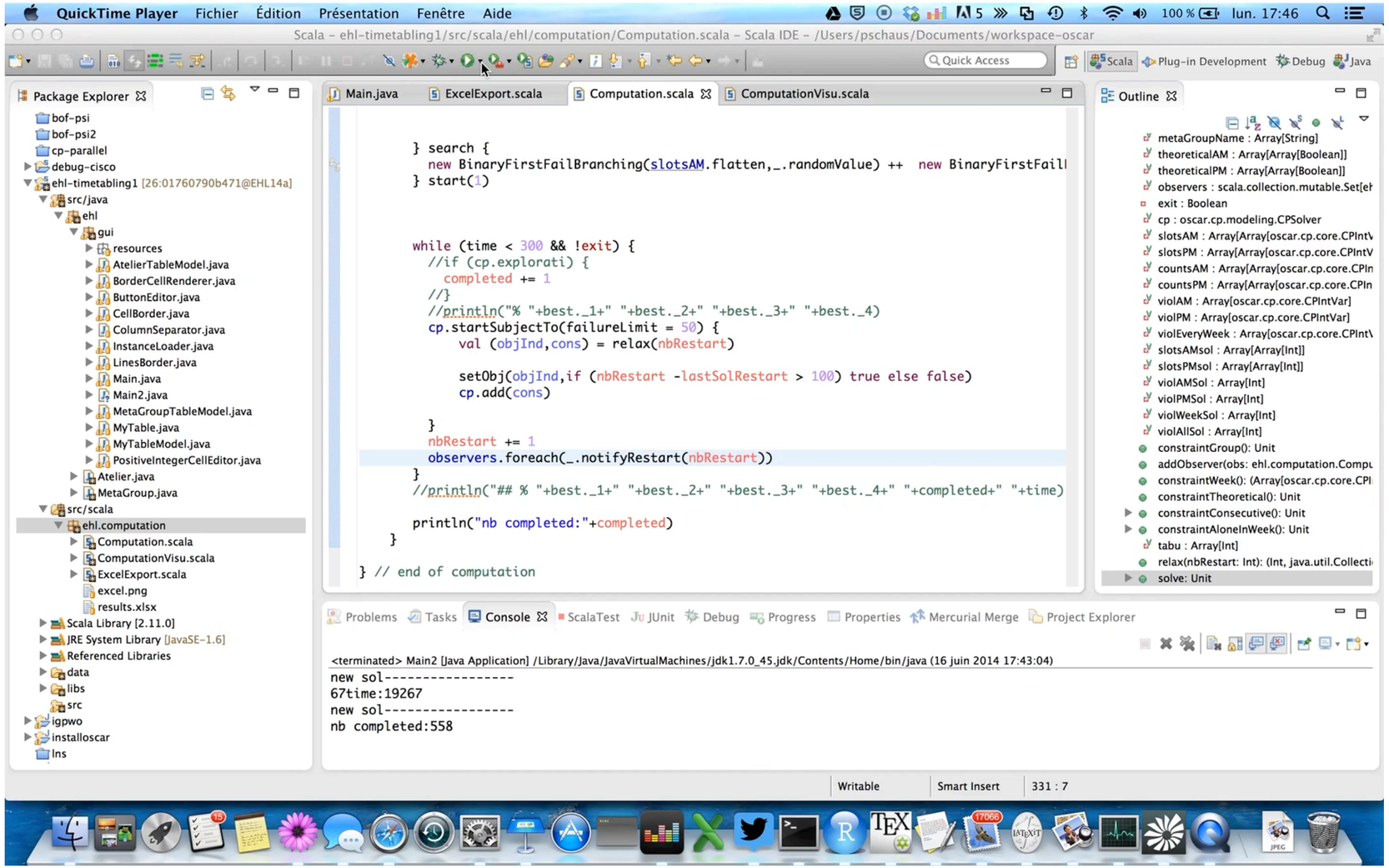
- •A theoretical course imposed in some slots
- •For some courses: if scheduled must be scheduled two consecutive full weeks
- •Some courses must scheduled the whole week (morning and afternoon)
- •Some courses can be scheduled a maximum number of times
- •Some courses must be scheduled a minimum number of times (some equivalence)

Vertical Constraints:

- •Each course must be scheduled a min/max number of times each AM/PM
- •Some courses must be scheduled every week
- •Synchronize groups with a same mother language to gave them theoretical courses at the same time

Schaus, P. (2013, November). Variable objective large neighborhood search: A practical approach to solve over-constrained problems. In *2013 IEEE 25th International Conference on Tools with Artificial Int*

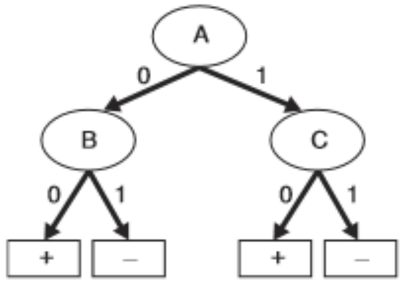
Rostering of a hospitality management school



Schaus, P. (2013, November). Variable objective large neighborhood search: A practical approach to solve over-constrained problems. In *2013 IEEE 25th International Conference on Tools with Artificial Int*

Machine Learning: Decision Trees

- Find the decision tree (with limited depth) that minimizes the error rate on a training set



Training:

Instance	A	B	C	Class
1	0	0	0	+
2	0	0	1	+
3	0	1	0	+
4	0	1	1	–
5	1	0	0	+
6	1	0	0	+
7	1	1	0	–
8	1	0	1	+
9	1	1	0	–
10	1	1	0	–

Validation:

Instance	A	B	C	Class
11	0	0	0	+
12	0	1	1	+
13	1	1	0	+
14	1	0	1	–
15	1	0	0	+

https://github.com/aia-uclouvain/pydl8.5

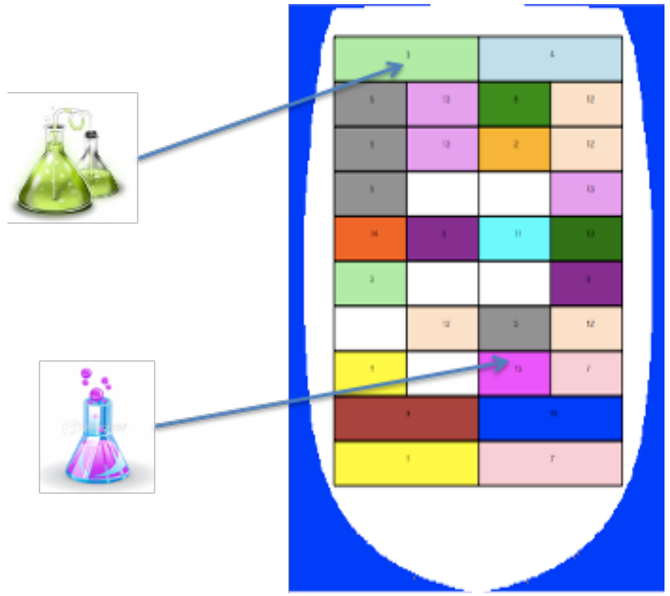
Aglin, G., Nijssen, S., & Schaus, P. (2020, April). Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 04, pp. 3146-3153).

Chemical Tanker

- For each product an amount (liters) to place is given
- Each tank has a maximum capacity (in liters)
- A set of chemical products must be placed into tanks on a tanker

Objective:

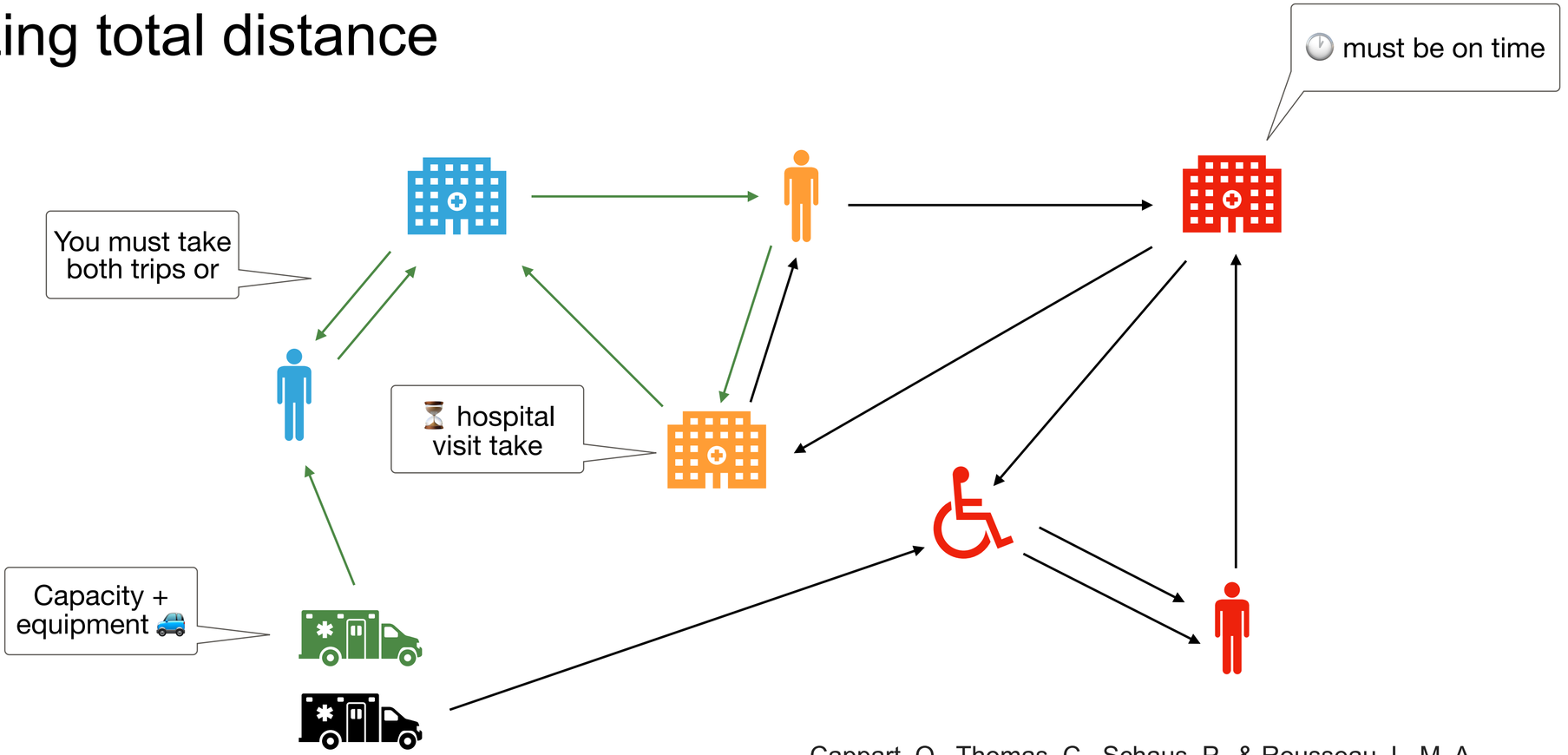
- Maximize the empty/unused space while placing all the products



Schaus, P., Régin, J. C., Van Schaeren, R., Dullaert, W., & Raa, B. (2012). Cardinality reasoning for bin-packing constraint: Application to a tank allocation problem. In *Principles and Practice of Constraint P Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings* (pp. 815-822). Springer Berlin Heidelberg.

Patient Transportation Problem (PTP)

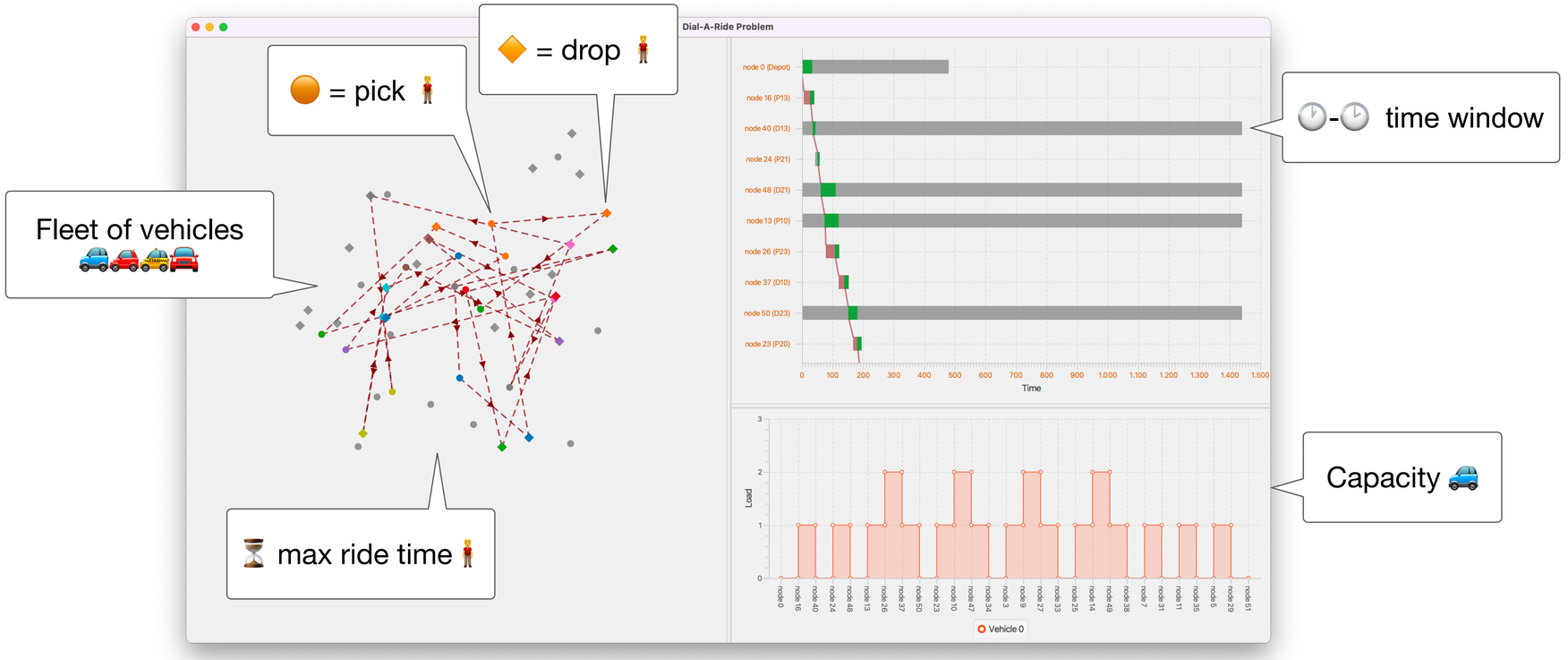
- 1: maximize the number of served requests
- 2: minimizing total distance



Cappart, Q., Thomas, C., Schaus, P., & Rousseau, L. M. A constraint programming approach for solving patient transportation problems. CP2018

Dial A Ride Problem

- Serve all the requests while minimizing total distance



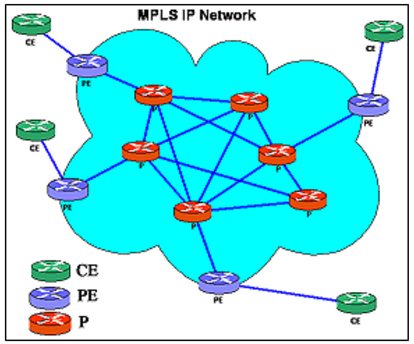
Delecluse, A., Schaus, P., & Van Hentenryck, P. (2022). Sequence Variables for Routing Problems. In *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

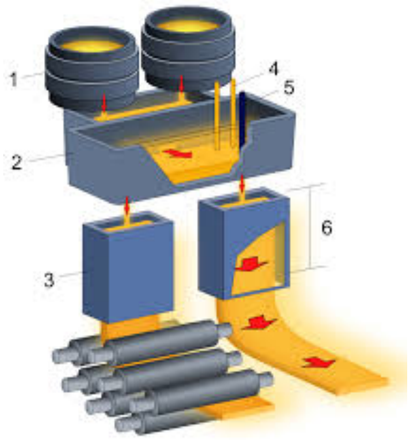
Air Plane Dismantling

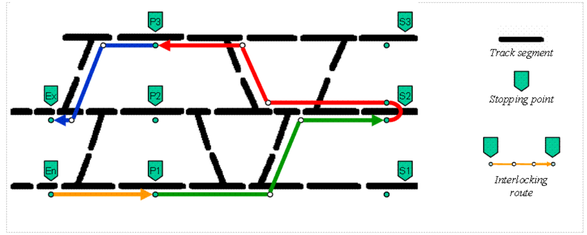
- Schedule the dismantling of a plane (with Charles Thomas)
  - Worker skills constraints
  - Precedence constraints
  - Tools and equipment constraints
  - Value objective, minimize makespan

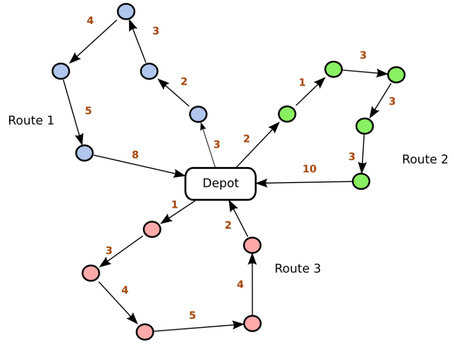
Thomas, C., & Schaus, P. (2024, May). A Constraint Programming Approach for Aircraft Disassembly Scheduling. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research* (pp. 211-220). Cham: Springer Nature Switzerland.

All the techniques we will see are used!









Advanced Algorithms for Optimization

LINFO2266

Introduction

Prof: Pierre Schaus
TAs: Alice Burlats, Amaury Guichard