Discrete Optimization

Quentin Louveaux

ULg - Institut Montefiore

2016

Discrete optimization problems

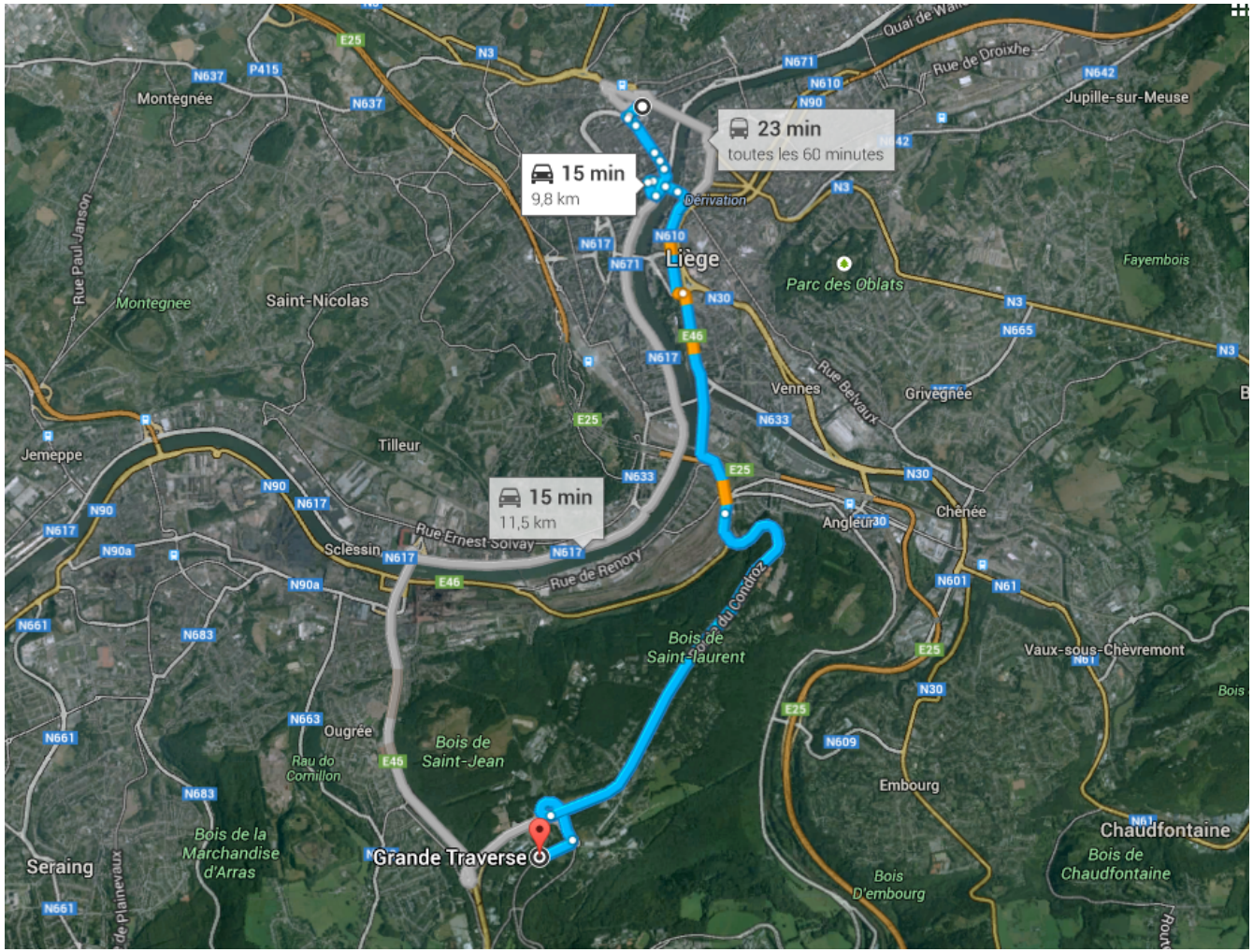
What is a discrete optimization problem ?

It is a problem with

- Optimization : there is a set of solutions. We look for the solution that maximize or minimize one criterion.
- Discreteness : (as opposed to continuous) some decisions must be taken as a whole : e.g. left or right, yes or no, $1, 2, 3, 4, \ldots$
  Very common in practice !
- (in pure cases) : a finite (but often large) set of solutions

Some well-known discrete problems

The shortest path

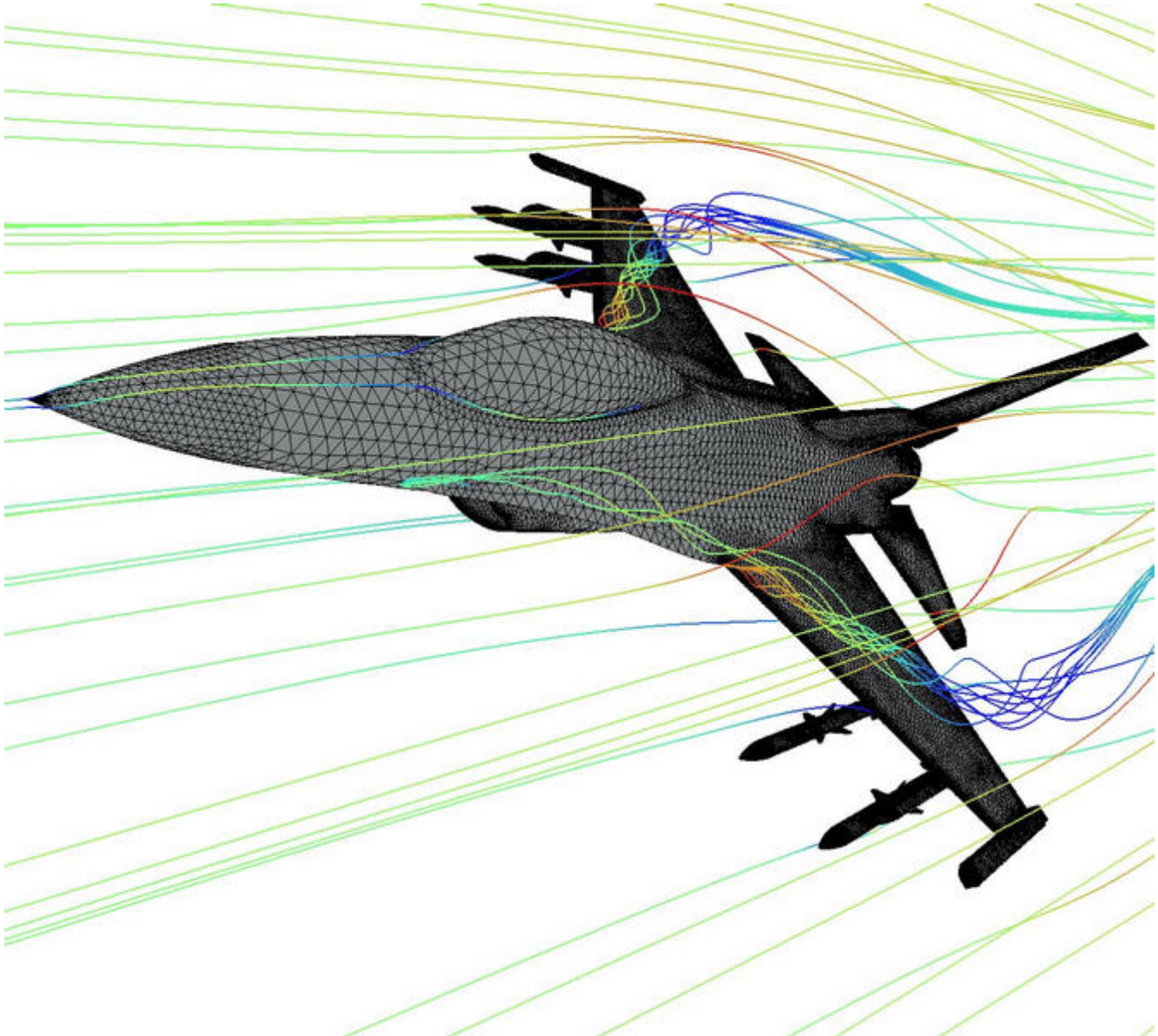


Some well-known discrete problems

The sorting problem

A lesser known but tractable problem

The matching problem



A popular discrete problem : the traveling salesman problem

Probably the most well-known discrete problem. . .
With many applications : Vehicle routing, VLSI design,. . .

Given n cities, in which order should one visit them in order to minimize the total distance ?

Example : Visit 23 EU cities with minimal traveling distance
Optimal solution

Why do discrete optimization problems have a bad reputation ?

Because of complexity theory and in particular

NP-hardness

Complexity theory in a nutshell

(to be covered in more detail in INFO0016 - Introduction to theory of computation (P. Wolper))
Following the work of Cook (1970's), most problems can be divided in two main categories

Polynomially solvable problems

Problems that can be solved with a complexity that is a polynomial in the size of the problem and the size of the input.

- Sorting problem $\mathcal{O}(n \log n)$
- Shortest path $\mathcal{O}(n^2)$
- Matching

NP-hard problems

A family of problems (including the TSP) for which no exact algorithm running in polynomial time has ever been found.

- If one of them can be solved in poly time, then all of them are !
- Still an open theoretical question whether there exists a poly time algorithm
- The common belief is that there does not exist a poly time algorithm

Really too hard ?

Common wrong belief

If your problem is a NP-hard problem, there is no way you can solve it efficiently !

This is a spread out thought even in the optimization community (sometimes even in the discrete optimization community) !

Goal of this lecture

Show that today's technology can actually solve these impossible problems !

Of course : efficiency of the methods will be very much problem structure dependent.
This is the major difference with polynomially solvable problems.

Main hurdle : being able to model correctly the problems.

Why is that so theoretically complicated ?

After all, there is a finite number of solutions
In particular $n!$ possible permutations

Imagine we can check 10^{12} possibilities per second
That is already a pretty amazing machine...

$10!$	0 sec
$20!$	28 days
$30!$	8400 billion years
$40!$	5 quadrillions times the age of the Earth...

Pure enumeration is hopeless

An algorithm based on exponential running time has an inherent limit.
Example : Enumerate all 2^n potential solutions of a problem with n binary choices.

n	Computer 1	Computer 2
Checks	10^9 per second	10^{15} per second
10	0 sec	0 sec
20	0 sec	0 sec
30	1 sec	0 sec
40	18 min	0 sec
50	13 days	1 sec
60	36 years	19 min
70	37000 y.	13 days
80	38 million years	38 years

To solve $n \approx 500$, one would need a computer 10^{140} times faster.
It is however possible to deal with relatively small instances.
Today we can deal routinely with problems with 1000 discrete variables

A first success-story : the traveling salesman problem

Applications of the TSP

- Truck routing
- Arranging school buses routes (the very first application)
- Scheduling of a machine to drill holes in a circuit board
- Delivery of meals to homebound people
- Genome sequencing
  Cities are local strings, and the cost is the measure of likelihood that a sequence follows another
- Link points through fiber optic connections in order to minimize the total distance and ensure that any failure leaves the whole system operational
- A robot that needs to explore its environment

Milestones in the solution of TSP instances

Year	Research team	Number of cities
1954	Dantzig, Fulkerson, Johnson	49 cities
1971	Held, Karp	64 cities
1977	Grötschel	120 cities
1980	Crowder, Padeberg	318 cities
1987	Padberg, Rinaldi	2392 cities
1994	Applegate, Bixby, Chvatal, Cook	7397 cities
2006	Applegate, Bixby, Chvatal, Cook, Helsgaun	80 000 cities

There is a $ 1000 prize for a challenge of 100 000 cities

Techniques : formulating as an integer programming problem, branch-and-bound, cutting planes.
$\rightarrow$ topics covered in the class

Its bad reputation...

- A simple word in a respectable optimization class of UC Berkeley...
  `https://inst.eecs.berkeley.edu/ ee127a/book/login/l-intro-complex.html`
- Data scientist Randy Olson who can solve approximately a 50-city TSP in 20 minutes using genetic algorithms made news headline.
  `http://www.randalolson.com/2015/03/08/computing-the-optimal-road-trip-across-the-u-s/`
- Reality of the technology
  TSP concorde app on iOS can solve it optimally on an ipod in less than 1 second ! Also available as a solver on windows and linux, the source code is open for academic use.
- TSP of 1000 cities can probably be solved as fast as an SDP with 1000 rows and columns in the matrix.

Another example of application

Exam scheduling

- Since 2015-2016, the exam schedule of the faculty is computed using discrete optimization algorithms
- 300 versions of courses, 1000 students, 600 dfferent cursus
- Make sure that no student has ever two exams on the same day !
- Try to maximize the rest days between two exams for a student
- Take care of professor constraints
- Compute the schedule in 10 hours of computing time - probably impossible to do with a customized algorithm

Other examples of applications

- Unit commitment problems
  Solved every day by electricity producers with thousands of variables
- Electricity market clearing
  Solved every day in 10 minutes (see Bertrand Cornélusse)
- Sports scheduling
  In baseball, football for example.
- Airlines
  network planning, fleet assignment, crew planning and rostering, gate assignment
- Health care
  treatment of prostate cancer with brachytherapy (placement of radioactive seeds inside a tumor)

Amazing speedups

Algorithmic speedups

- Cplex 1 (1991) $\rightarrow$ Cplex 11 (2007) : 30 000 speedup
- gurobi (2009) $\rightarrow$ gurobi (2016) : 11 speedup

Machine speedup

- 1990 $\rightarrow$ 2016 : 1000 speedup

Total speedup

- 6 months (1990) $\rightarrow$ 1 second (2007)
- 10 years (1990) $\rightarrow$ 1 second (2016)

Learning outcomes

- Being able to model discrete problems
- Being able to recognize a good from a bad formulation
- Being able to recognize well-solved discrete problems from NP-hard ones
- Know the main algorithmic techniques to solve the easy and hard discrete problems

Organization

Schedule

- Lecture at 2 :00 pm every Friday
- Exercises follow the lecture
- 2 Modeling and Implementation projects (one individual and by groups of 2)
  Will start very soon !

Grading

- The two projects count for 1/2 of the mark.
- An exercise exam (**written**) counts for 1/2 of the mark.
- I use a geometric mean, i.e.

$$Grade = \sqrt{Exam \times (\frac{Project\ 1 + Project\ 2}{2})}$$

Modeling

The main focus of the lecture is to formulate problems using

mathematical optimization formulations.

That means that we want to define variables, mathematical constraints (inequalities and equalities essentially) and an objective function to maximize or minimize.

$$\begin{aligned} \min\ & c(x) \\ \text{s.t. } & f(x) \leq b \\ & g(x) = 0 \\ & x \in X. \end{aligned}$$

Why mathematical programming ?

Typical programming approach

- Analyze the problem
- Write an algorithm to solve the problem using `while`, `if`, `then`, `else`,...
- Proof the corectness of the algorithm
- Analyze the complexity of the algorithm

Pros and cons

- Works well for well-posed problems.
- Works well for tractable problems.
- Needs to be done from scratch if there is a slight change in the problem or additional constraints

Why mathematical programming ?

The mathematical programming approach

- Analyze the problem
- Write a static mathematical model
- Rely on meta-algorithms that work on all models that are correctly written

Pros and cons

- Sometimes difficult to formulate the problems in the right way (no `if`, `then`, `else`)
- Works also for hard problems
- Two different models of the same problem may not be as good as the other
- Very flexible to add complicating constraints.

Some random applications

Electricity production

An electricity producer wants to plan the level of production of his main power plants in order to fulfill the demand and minimize his costs. The demand for 6 time periods in the day are listed in the following table.

Time	0-4h	4-8h	8-12h	12-16h	16-20h	20-0h
Demand (GWh)	2	3	9	9	17	8

The producer has 6 coal power plants. 3 of them have a power of 1GW while the remaining 3 have a power of 2GW.
The cost of operating a 1GW plant is 100 €/GWh while the cost of operating a 2GW plant is 200 €/GWh.
There is a fixed cost for using a plant of 100 €per hour of use.
Finally, starting up a plant costs 400 €.

Type	Number	Power (GW)	Varying Cost €/GWh	Fixed Cost €/h	Startup Cost €
1	3	1	100	100	400
2	3	2	200	100	400

What is the optimal production planning to minimize the costs while satisfying the demand ?

Location of GSM transmitters

A mobile phone operator decides to equip a currently uncovered geographical zone.

The management allocates a budget of 10 million € to equip this region.

7 locations are possible for the construction of the transmitters.

Every transmitter only covers a certain number of communities.

Where should the transmitters be built to cover the largest population with the given budget ?

Sequencing jobs on a bottleneck machine

In workshops, it may happen that a single machine determines the throughput of the production $\rightarrow$ critical machine.

A set of tasks is to be processed. The execution is non-preemptive. For every task i, a release date and duration are given.

Different possible objectives : total processing time (makespan), average processing time, total tardiness (if due dates are given)

Scheduling of telecommunications via satellite

A digital telecommunications system via satellite consists of a satellite and a set of stations on earth.

We consider for example 4 stations in the US that communicate with 4 stations in Europe through the satellite. The total traffic is given through a matrix $TRAF_{tr}$.

The satellite has a switch that allows any permutation between the 4 transmitters and the 4 receivers.

The cost of a switch is the duration of its longest packet.

The objective is to find a schedule with minimal total duration.

Scheduling nurses

A working day in a hospital is subdivided in 12 periods of two hours.
The personnel requirements change from period to period.

A nurse works 8 hours a day and is entitled to a break after having worked for 4 hours.

Determine the minimum number of nurses required to cover all requirements ?

If only 80 nurses are avalable, and assuming that it is not enough, we may allow for a certain number of nurses to work for a fifth period right after the last one. Determine a schedule the minimum number of nurses working overtime.