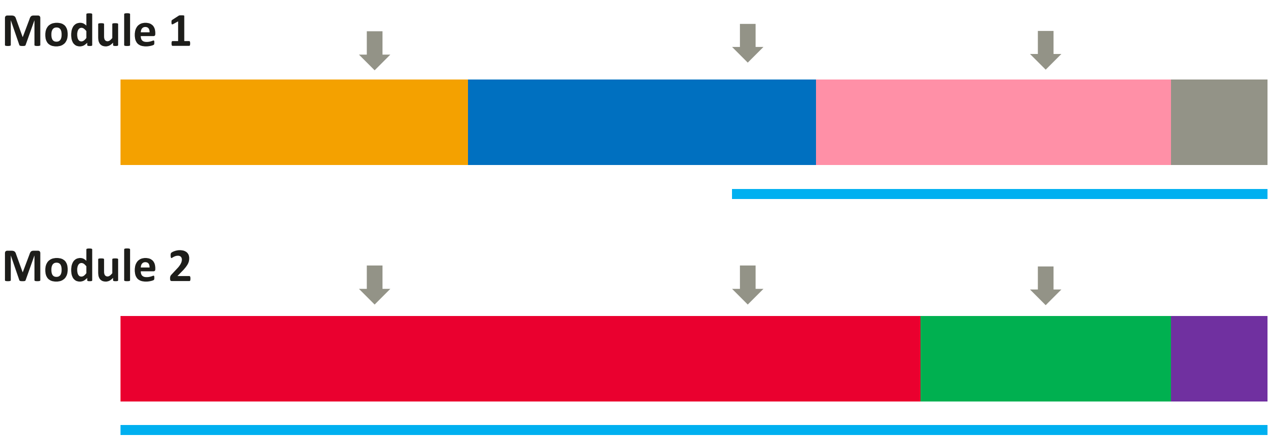
Discrete Optimization and Integer Programming.

Course overview

Course program

Topics

- Introduction in optimization
- Linear programming
- Introduction in Integer programming
- Theory and Practice of Integer programming
- Multi-criteria optimization problems



Practice

- MiniZinc – open-source mathematical optimization platform
  https://www.minizinc.org/

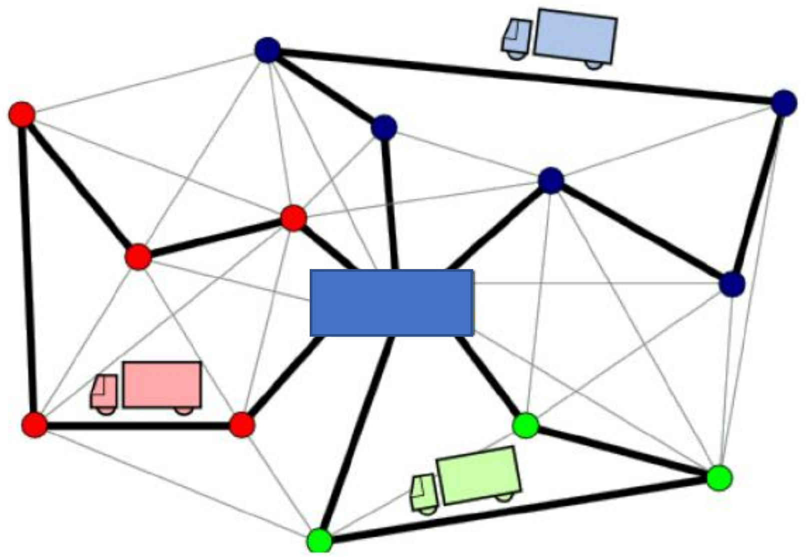
Home works & exam

- Home work
- Exam

Mark

0.6 ■ + 0.4 ■

Discrete optimization



There is a warehouse with a fleet of vehicles and many customers to whom it is necessary to deliver goods. For each client, a set of goods and a delivery time interval are defined. It is necessary to draw up a plan for optimal delivery, taking into account restrictions:

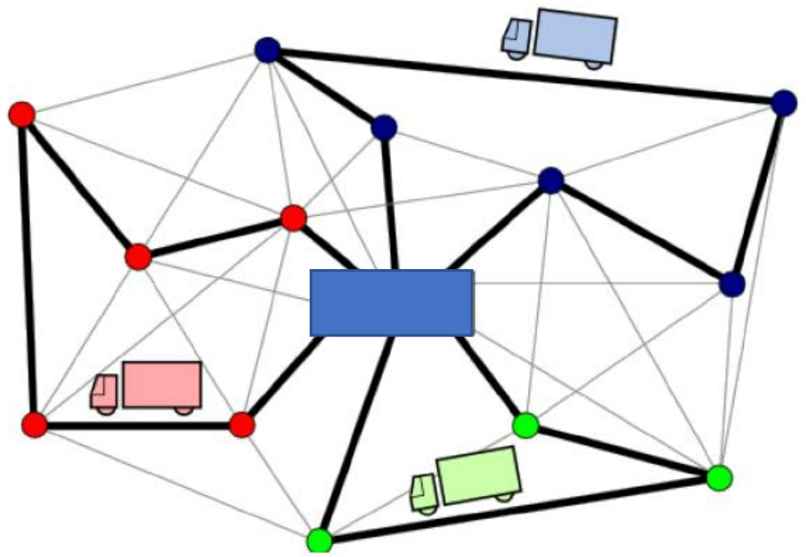
- The carrying capacity of the delivery vehicles must not be violated
- All goods must be delivered within the specified time intervals
- Delivery drivers must follow the rules of the road
- …

Objective: minimize cost = driver’s salary + fuel cost

Discrete optimization

Let's simplify it

- 1 car with infinite capacity
- No time constraints for product delivery
- No driving rules
- Driver works for free (for food)



Travelling Salesman Problem (TSP)

There is a full graph K_n $(n \geq 3)$ with defined set of edges E and travelling cost $c: E(K_n) \to \mathbb{R}_+$.

We need to find Hamiltonian cycle T, with minimal cost $\sum_{e \in E(T)} c(e)$.

Travelling Salesman Problem

Travelling Salesman Problem (TSP)
There is a full graph K_n $(n \geq 3)$ with defined set of edges E and travelling cost $c: E(K_n) \to \mathbb{R}_+$.
We need to find Hamiltonian cycle T, with minimal cost $\sum_{e \in E(T)} c(e)$.

Enumeration algorithm
Number of n −cycles in full graph K_n equals to
$(n-1) \cdot (n-2) \cdot \ldots = n - 1!$

Modern computer can evaluate $\sim 10^6$ cycles in 1 seconds.

Number of vertices	Computational time
12	40 seconds
15	1 day
19	203 years
21	77 000 years

Maybe technical progress can invent better computers?

Travelling Salesman Problem

Travelling Salesman Problem (TSP)
There is a full graph K_n $(n \geq 3)$ with defined set of edges E and travelling cost $c: E(K_n) \rightarrow \mathbb{R}_+$.
We need to find Hamiltonian cycle T, with minimal cost $\sum_{e \in E(T)} c(e)$.

Planck computer
Planck time (time quant) $t_P = 5{,}39116 \cdot 10^{-44}$ s the time required for light to travel a distance of 1 Planck length in a vacuum. No current physical theory can describe timescales shorter than the Planck time.

Imagine «Planck TSP - computer», which allows to evaluate 1 cycle in t_P or $1{,}8 \cdot 10^{43}$ cycles in one second.

Number of vertices	Computational time
38	0,7 seconds
42	20 days
49	1,5 ages of Universe

Travelling Salesman Problem

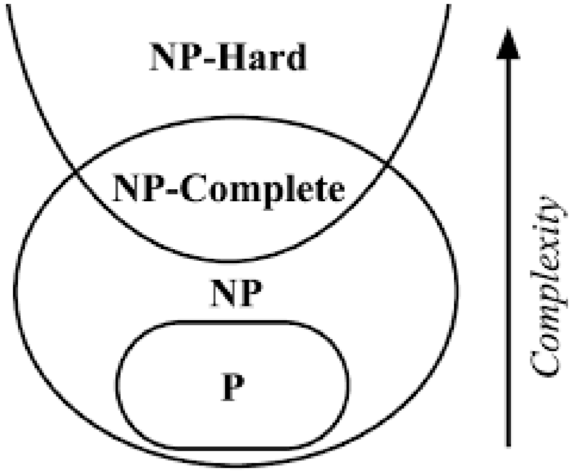
Bellman-Held-Karp algorithm: problem solution complexity for graph K_n – $O(n^2\,2^n)$ operations.

Modern computer:

Number of vertices	Computational time
28	1 day
36	1 year
68	1 universe age

Still not good enough? Problem is NP-complete.

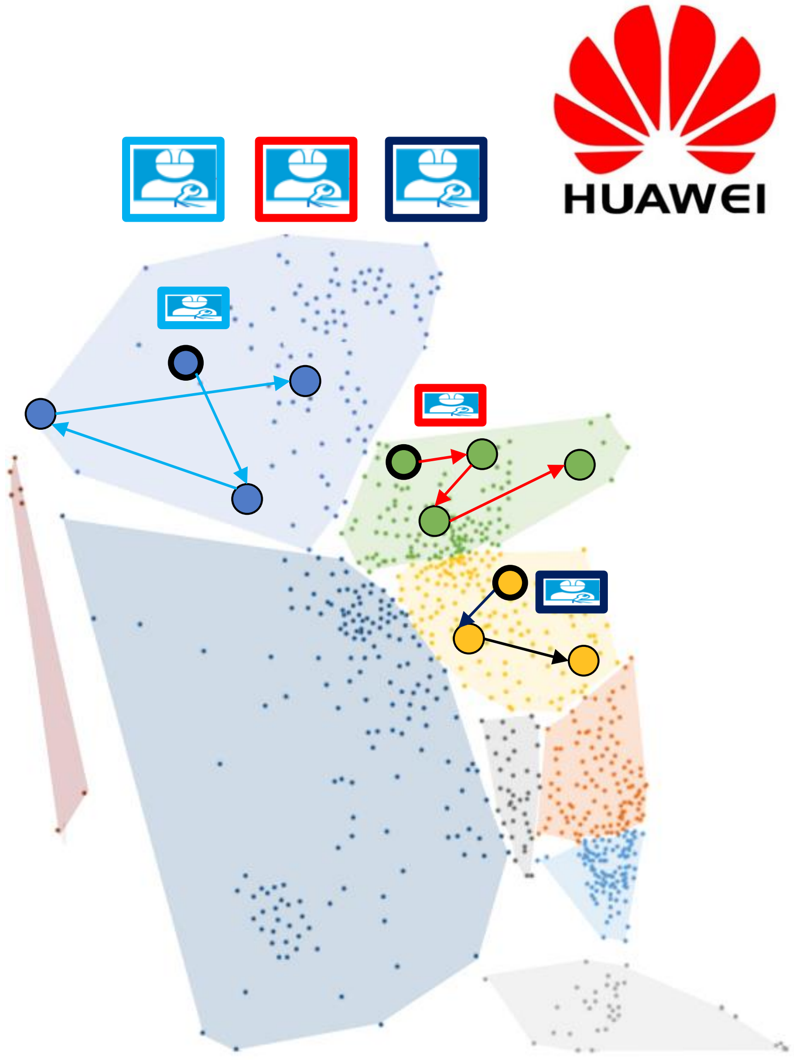
Travelling Salesman Problem (TSP)
There is a full graph K_n $(n \geq 3)$ with defined set of edges E and travelling cost $c: E(K_n) \to \mathbb{R}_+$.
We need to find Hamiltonian cycle T, with minimal cost $\sum_{e \in E(T)} c(e)$.



Travelling Salesman Problem

Bellman-Held-Karp algorithm: theoretical evaluation for worst case

Number of vertices	Computational time
28	1 day
36	1 year
68	1 universe age



Real application: scheduling problem for Huawei technical staff to service base stations

- 40 000 base stations
- 200 specialists with different skills
- Large number of additional business constraints
- Complex cost-based objective function

And what we can do?

How to solve NP-problems?

1. **Problem-specific.**

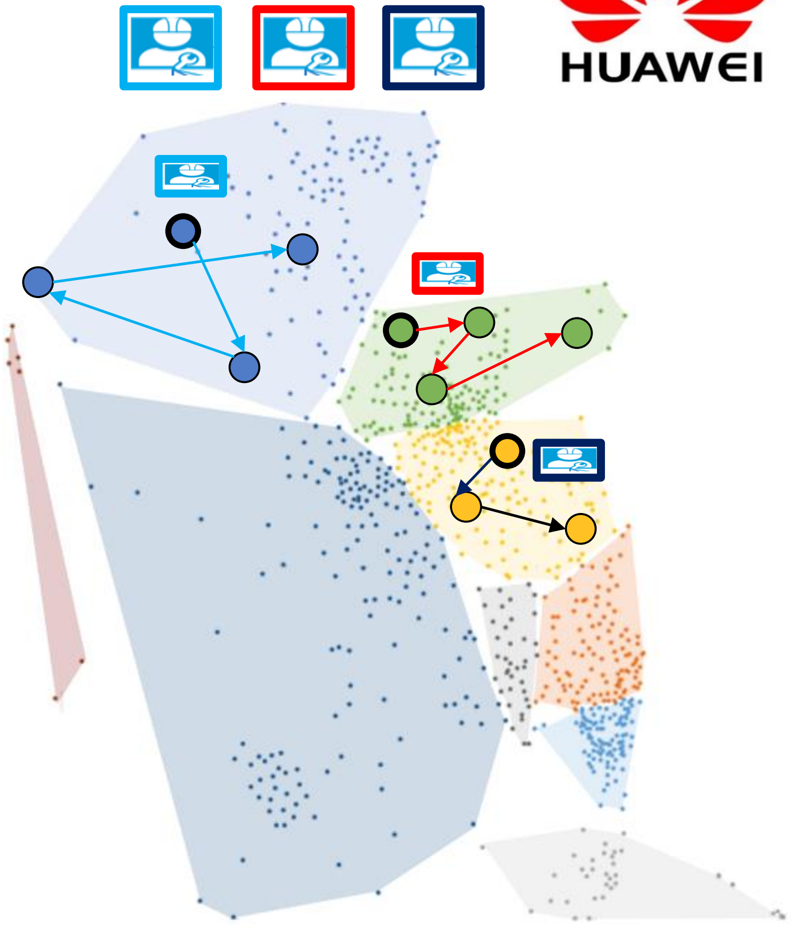
Overfeeds by problem specialties and data properties.
Example: TSP with Euclidean distances is simpler than general ones.

2. Domain-specific algorithms.

Example: Lin-Kernighan-Helsgaun heuristic (http://webhotel4.ruc.dk/~keld/research/LKH/) doesn't guarantee optimal solution, but can successfully solves TSP for instances with ~100 000 of vertices.

3. **General.**

Example: Solvers.



HUAWEI

Optimization problem solvers

Solvers – software packages which allow to solve a large variety of problems using the following methods:

- Linear Programming (LP)
- Mixed-integer Linear Programing (MILP)
- Quadratic Programming (QP)
- Constraint Programming (CP)
- Boolean satisfiability solvers (SAT)
- Exact algorithms (i.e. B&B)
- Heuristics: local search, greedy algorithms, …

Solvers give an access to ready-to-run algorithms. One just need to formulate the problem in terms of solvers API, tune parameters and push the button.

Linear programming (LP)

Constraints

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i, (i = 1, 2, \dots, m)$$

$$x_j \geq 0. (j = 1, 2, \dots, n)$$

Objective

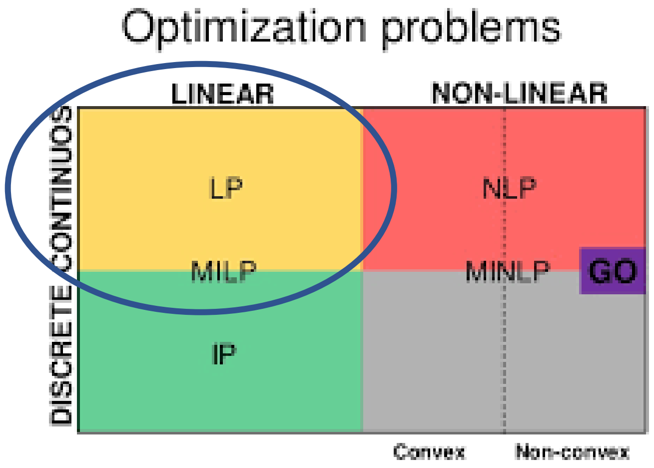
$$f(x) = \sum_{j=1}^{n} c_j x_j$$

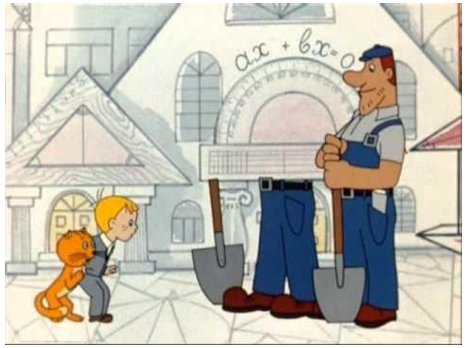
Solving methods

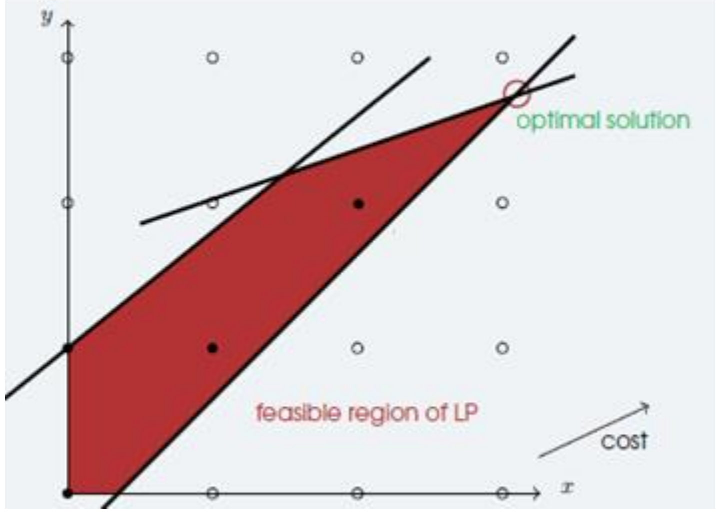
- Simplex methods
- Barrier methods
- Interior point method

Specialties

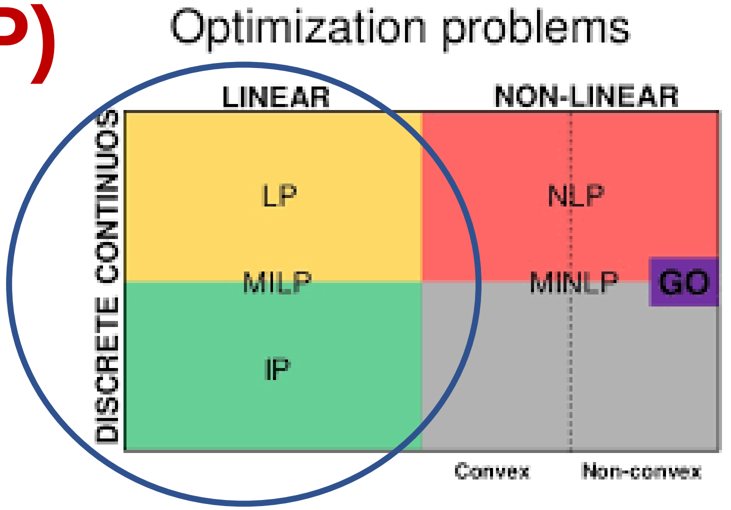
- Can be solved fast
- Can guarantee integer variable values only for totally unimodular matrix of problem input
- Limited number of problems can be formulated as pure LP







Mixed - integer linear programming (MILP)



Constraints

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i, (i = 1, 2, \dots, m)$$
$$x_j \geq 0. (j = 1, 2, \dots, k)$$
$$x_j \in \mathbb{Z}_+ (j = k + 1, \dots, n)$$

Objective

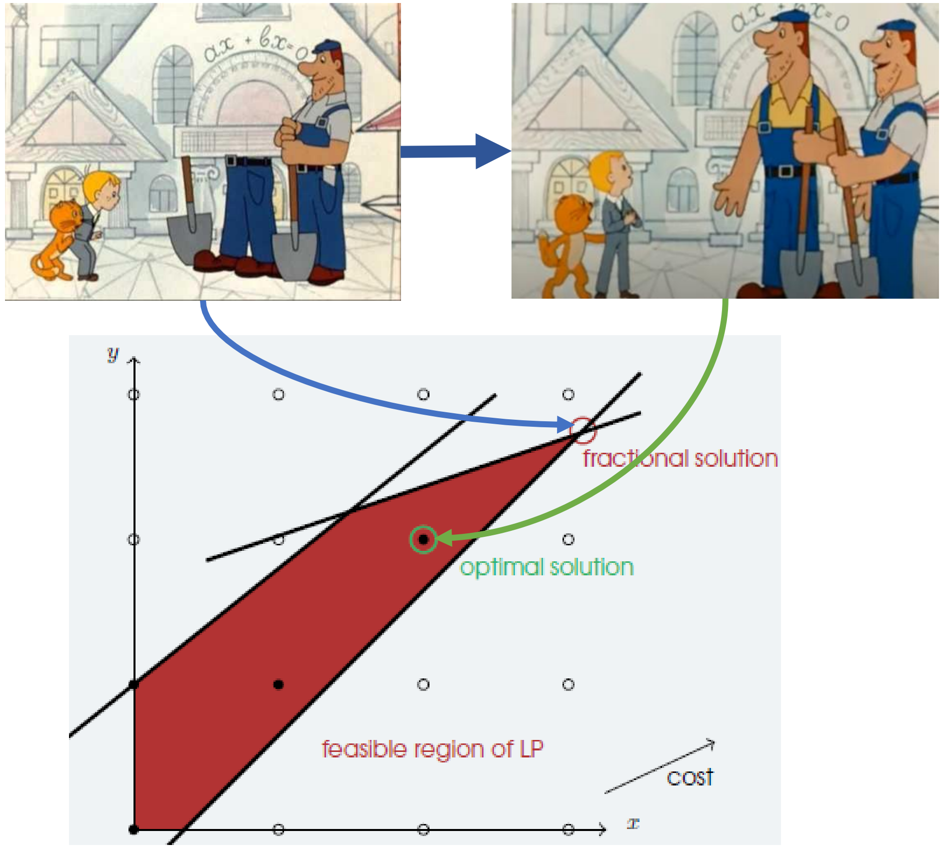
$$f(x) = \sum_{j=1}^{n} c_j x_j$$

Solving methods

- Linear programming
- Heuristics
- Cutting planes
- Branch & Bound

Specialties

- Non-polynomial
- Can guarantee that some variables are integer
- Rich variety of problems can be stated as MILP
- Plenty of solvers can be used to solve the stated problem



Constraint programming

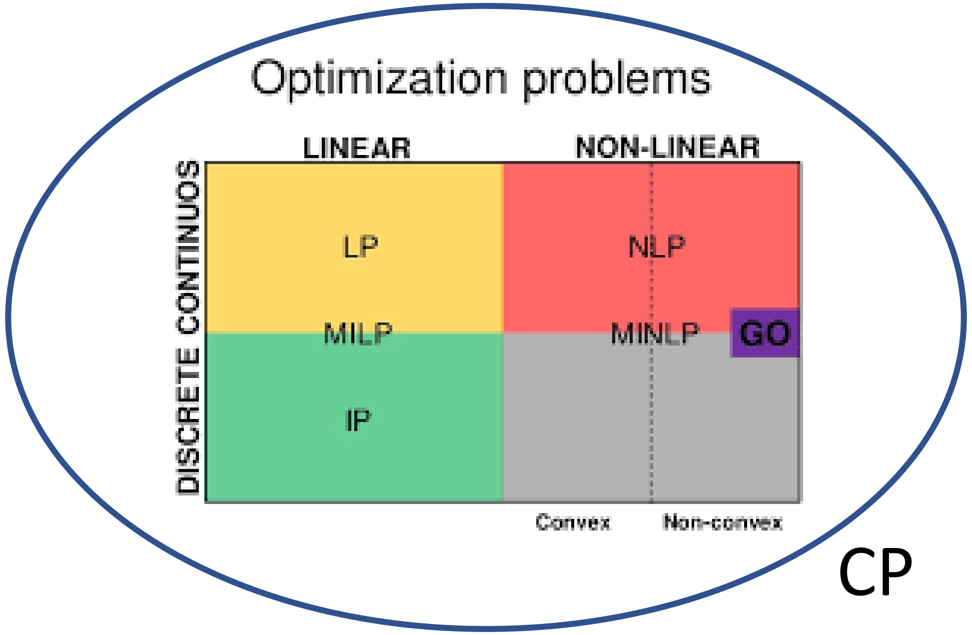
May have any kind of API.
The main idea is a detailed description of the space for checking the feasibility constraints.

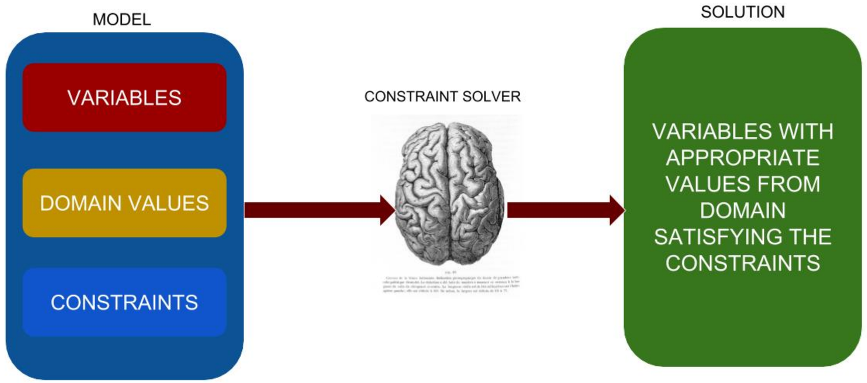
Solving methods

- Branch and bound
- Constraint propagation algorithms
- Heuristics

Specialties

- Exponential complexity for most problems
- Larger variety of modeled problems than MILP
- Non-linear constraints and objectives
- Global constraint propagation algorithms should be developed to create efficient solver. There are > 400 constraints presented in Global Constraint Catalogue https://sofdem.github.io/gccat/gccat/sec5.html





Constraint programming

Example: Battleship

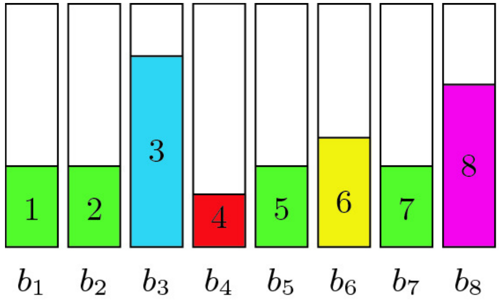
Example: Sudoku

	6		1		4		5	
		8	3		5	6		
2								1
8			4		7			6
		6				3		
7			9		1			4
5								2
		7	2		6	9		
	4		5		8		7	

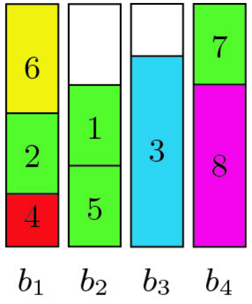
Constraint programming

Example: scheduling problems

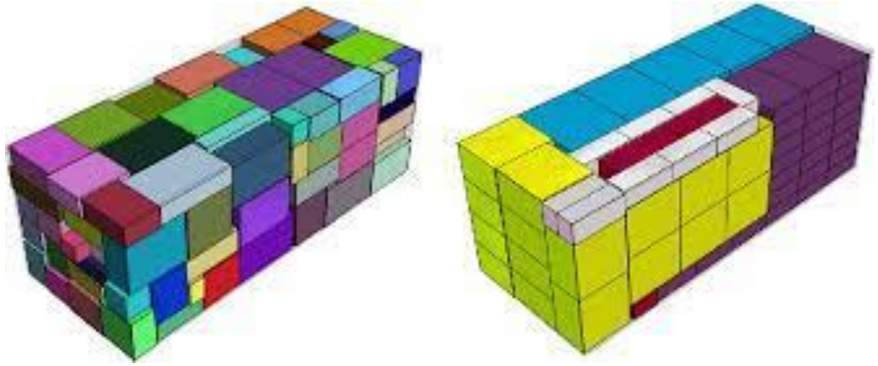
Example: bin packing



A feasible solution, with 8 bins



An optimal solution, with 4 bins



Solvers

Quick solver run guide

1. Create mathematical model.
   Different solvers have different interfaces, this is due both to the set of methods used and the fact that these are products of different companies. Most solvers have APIs for Java, C++ and Python. Some solvers have their own modeling languages (i.e. CPLEX OPL).
2. Push “Run” button. ☺

Find problem solution – more harder! The result depends on:

- Stated model. There are several ways to state correct model, not all of them are fast.
- Choose solver run settings. SCIP solver has ~2700 setting parameters. 🙃
- Choose the right solver for considered problem.
- Solver power and (If everything was simple, maybe we already proved P=NP?)

Solvers – not a panacea, but very powerful tool

Solver applications

Example: Gurobi

https://www.gurobi.com/

Business Problems

Production

- Inventory optimization
- Production mix
- Machine allocation

Finance

- Capital Budgeting
- Cash Management
- Revenue Optimization

Distribution

- Fuel use minimization
- Maintenance planning
- Less-than-truckload (LTL) loading

Investments

- Portfolio Optimization
- Fund Cloning
- Bond Management

Purchasing

- Inventory Stocking & Reordering
- Vendor Selection
- Shipment Planning

Human Resources

- Workforce Scheduling
- Office Assignment

Solver using aspects

- Unpredictable behavior. For most problems one should try solver to see its performance.
- Best solvers are commercial products. Detailed behavior of these solvers is kind of “black box” for users.
- There are no Russian solvers yet ☹

To use solver company need specialists which can,
Level 1: Build correct models
Level 2: Build models which finds solution fast
Level 3: Understand solvers back-end and how it can be combined with other algorithms

Course program

Topics

- Introduction in optimization
- Linear programming
- Introduction in Integer programming
- Theory and Practice of Integer programming
- Multi-criteria optimization problems

Practice

- MiniZinc – open-source mathematical optimization platform https://www.minizinc.org/

Home works & exam

- Home work
- Exam

+

- State MILP and LP problems with MiniZinc modeling interface
- Solve problems using COIN-OR Branch-and-Cut solver

Contacts

Arkhipov Dmitry
miptrafter@gmail.com

Lavrov Alexey
xnikiv@gmail.com

Huawei Russian Research Institute
Mathematical Modeling and Optimization
Algorithm Competence Center

7-9 Smolenskaya square.