Constraint Systems

Optimization and
Meta-constraints in or-tools

Optimization in or-tools

Branch and Bound in or-tools works via a search monitor

In detail:

```
m = slv.Minimize(<variable>, <step>) # Minimization
m = slv.Maximize(<variable>, <step>) # Maximization
...
slv.NewSearch(<dec builder>, [m, <other mon.>])
```

Instead of a cost function, we have a cost variable

- Cost function implemented by posting a constraint

The **<step>** parameter represents the required improvement

- Bounding constraint (minimization): $z \leq zbest - step$

Meta-constraints in or-tools

Reified constraints in or-tools

- Not all constraints can be reified!
- But many of them can

We just need to use the constraint as a term in an expression

- Usually, this requires adding brackets

```
expr = 2 * (x <= y)
```

- `(x <= y)` represents the feasibility state of $x \leq y$
- It can be used like any other expression

Max/Min operators in or-tools

Just a word of warning:

The **`sum`** function in python repeatedl applies **+**

- Since **+** is redefined in or-tools…
- …we can use **`sum`** to build expressions

In python **`min`** and **`max`** are functions, not operators

- Hence, they are not redefined in or-tools
- Istead, we have ad-hoc functions in the solver API

```
slv.Min(<expr/var>, <expr/var>) # Binary min
slv.Min(<list of vars>) # Min with many terms
slv.Max(<expr/var>, <expr/var>) # Binary max
slv.Max(<list of vars>) # Max with many terms
```

Constraint Systems

Tanks Problem (from Lab 2)

Problem Description

A number of chemicals must be stored in an array of tanks.

Problem Description

A number of chemicals must be stored in an array of tanks.

- There is a known amount of each chemical
- Each tank can store a single chemical
- Each chemical must be stored in a single tank
- Each tank has limited capacity, which cannot be exceeded
- Some chemicals are dangerous
  - Dangerous chemicals must be stored in special "safe" tanks
- Each chemical should be kept within a certain temperature range
  - If $tmax_i < tmin_j$ or $tmax_j < tmin_i$
  - Then chemical i and j cannot stay in adjacent tanks

Problem Description

Build a (CP based) solution approach for the problem

- The file **`ch4-tanks.py`** contains a template script
- The script accepts as a command line argument an instance file

```
python ch4-tanks.py <instance file>
```

- Instance files can be found in the folder **`tank-data`**
- During development, use **`data-tanks-debug.json`**
- Then attack more complex stuff

Problem Description

Use the following constraint library as reference:

- Arithmetic expressions: `+, -, *, /, abs...`
- Equality/disequality/inequality `==, !=, <, <=, >=, >`

Try to build the model incrementally

- And remember you don't have to start implementing immediately
- Plain old paper is good enough to design a model!