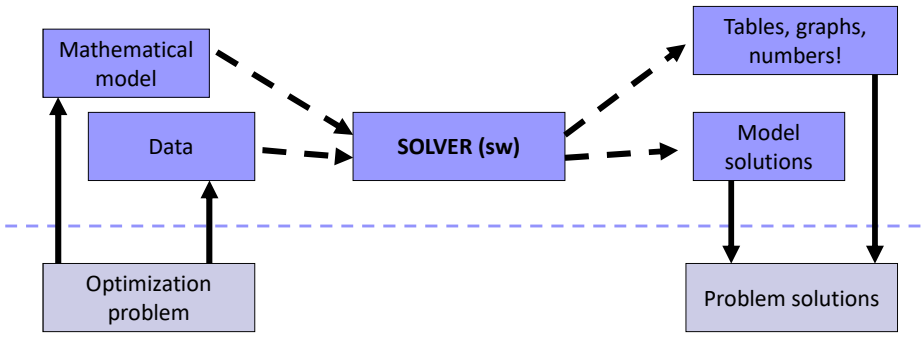
# Solvers for Mathematical Programming

## Solvers (optimizing engines)

A **solver** is a software application that takes the description of an optimization problem as input and provides the solution of the model (and related information) as output.




2.2

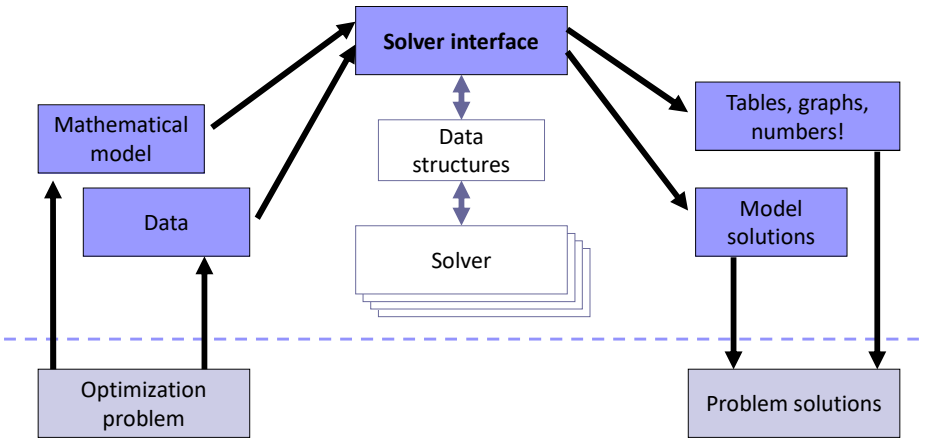
## MILP solvers

- Most used in practice:
  - very efficient
  - numerical stability
  - easy to use or embed
- 1 000 000 000 speed-up in the last 15 years
  - hardware speed-up: x 1000
  - simplex improvements: x 1000
  - branch-and-cut improvement: x 1000
- e.g. Cplex, Gurobi, Xpress, Scip, Lindo, GLPK etc.

2.3

## Solver interfaces

A solver can be accessed via **modelling languages** or **general-purpose-language libraries**




2.4

## IBM Ilog Cplex

- One of the first MILP solvers
- Includes **state-of-the-art** technology
- (One of) the best solvers available (Gurobi, Xpress)
- Possible interfaces
  - Interactive optimizer
  - **OPL** / AMPL / ZIMPL … algebraic modelling language
  - **C – API libraries (Callable libraries)**
  - C++ libraries (Concert technologies)
  - Python / Java / .Net wrapper libraries
  - Matlab / Excel plugins

2.5

## Accessing / Getting IBM Ilog Cplex

- Installed at LabTA
- From home
  - Getting your own free academic license
  - Accessing OPL via ssh / X-windows (or similar)
  - Accessing Cplex via ssh
- See `01.gettingStarted/IBMacademic.txt` for details!

2.6

## Optimization Programming Language - OPL

- Close to algebraic modelling language
  - direct mapping of sets, parameters, decision variables, constraints
  - use algebraic primitives (`forall`, `sum` etc.)
- Integrated Development Environment (IDE) available
- Included in the Cplex Studio package
- Learning OPL by examples

2.7

## Basic commands

- To enable Cplex Studio

  `. cplex_env` (notice “dot blank”)

- To run the OPL IDE

  `/opt/ibm/ILOG/CPLEX_Studio1261/opl/oplide`

2.8

## IDE commands

- Basic OPL projects
  - **model files** (.mod): models in OPL language
  - **data files** (.dat): parameters data
  - **Run Configurations**: collect models and data to configure a specific problem instanceopl/oplide
- Basic IDE commands
  - `File->New->OPL Project`
    (create a new project in a specific directory)
  - `File->Import->Existing OPL Project`
    (open an existing project)
  - `Help->Help Contents->IDE and OPL->Optimization Programming Language (OPL)`

2.9

## A first simple model `[1.mix_perfumes]` 1/2

- **d**ecision **var**iables:

```
dvar <dvar_type> decision_variable_name;
```

| | | |
|---|---|---|
| `<dvar_type>` = | `float` | (real variables) |
| | `float+` | (real variables ≥ 0) |
| | `int` | (integer variables) |
| | `int+` | (integer variables ≥ 0) |
| | `boolean` | (binary variables) |

- Objective function:

```
maximise (or minimise) <expression>;
```

2.10

## A first simple model [1.mix_perfumes] 2/2

- Constraints:

```
subject to {
    constraint1_name: <expression>;
    constraint2_name: <expression>;
    ...
}
```

<expression> = e.g.

```
sum( i in setI, j in setJ )
    <expression using indexes i and j>
```

*try with diet_food...*

2.11

## Generalizing the model [3.mix_general_model] 1/2

- Sets

```
setof(<data_type>) set_name = { <element_list> };
<data_type> = string, int, float, etc. etc.
```

- Parameters

```
<data_type> parameter_name = parameter_value;
<data_type> 1dim_vector_name[set_name] =
                        [element1,element2,...];
<data_type> 2dim_vector_name[set1][set2] = [
    [element_1_1,element_1_2, element_1_3, ...],
    [element_2_1,element_2_2, element_2_3, ...],
    ...
];
<data_type> Ndim_vec[set1][set2]…[setN] = …
```

2.12

## Generalizing the model [3.mix_general_model] 2/2

- Constraints

```
forall ( k in set ) {
     constraint_name: <expression using index k>
}
```



## Separating model and data

[4.mix_general_separated] 1/3

- `.mod` file (cont.)

```
//sets
setof(<data_type>) set_name = ...;


//parameters
<data_type> parameter_name = ...;
<data_type> 1dim_vector_name[set_name] = ...;
<data_type> 2dim_vector_name[set1][set2] = ...;
<data_type> Ndim_vec [set1][set2] ]…[setN] = ...;
```

## Separating model and data

[4.mix_general_separated] 2/3

- (cont.) **.mod** file

```
//decision variables
dvar <dvar_type> decision_variable_name;
dvar <dvar_type> 1dim_dec_var_vector[set_name];
dvar <dvar_type> 2dim_dec_var_vector[set1][set2];
dvar <dvar_type> Ndim_dec_var[set1][set2]…[setN];
```



## Separating model and data

[4.mix_general_separated] 3/3

- **.dat** file

```
set_name = { element1, element2, ...}

parameter_name = <value>;
1dim_vector_name = [element1,element2,...];
2dim_vector_name = [
     [element_1_1,element_1_2, element_1_3, ...],
     [element_2_1,element_2_2, element_2_3, ...],
     ...
];
```

*try with cover models*

## Exercises

- Basic transportation model `[transport OPL project]`
- Facility location with fixed costs
  `[LocationWithFixedCosts OPL project]`
- Build the OPL project, model and data for the "Moving scaffolds (iron rods) between construction yards" problem *(do it yourself!)*

2.17