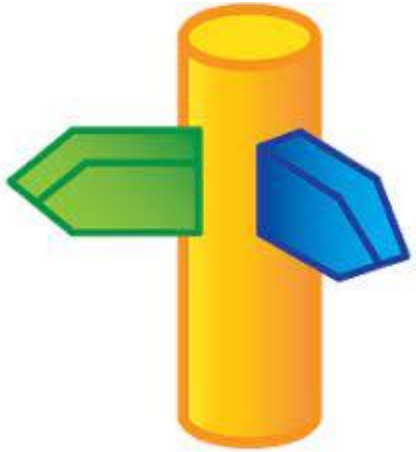


CPLEX IBM solver

Brahimi Mohammed

CPLEX Toolkit

- CPLEX allows one to work in several ways.
- An IDE that uses the OPL modeling language
- An interactive command-line optimizer
- A callable library in several languages
  - Java
  - Python
  - C
  - C++ (Concert Technology)
  - ...

CPLEX & Optimization Programming Language (OPL)

- Math-like Modelling Language
- Clean separation between model and data
- CPLEX solver for mathematical models
- Solve linear, quadratic, and quadratically constrained programs

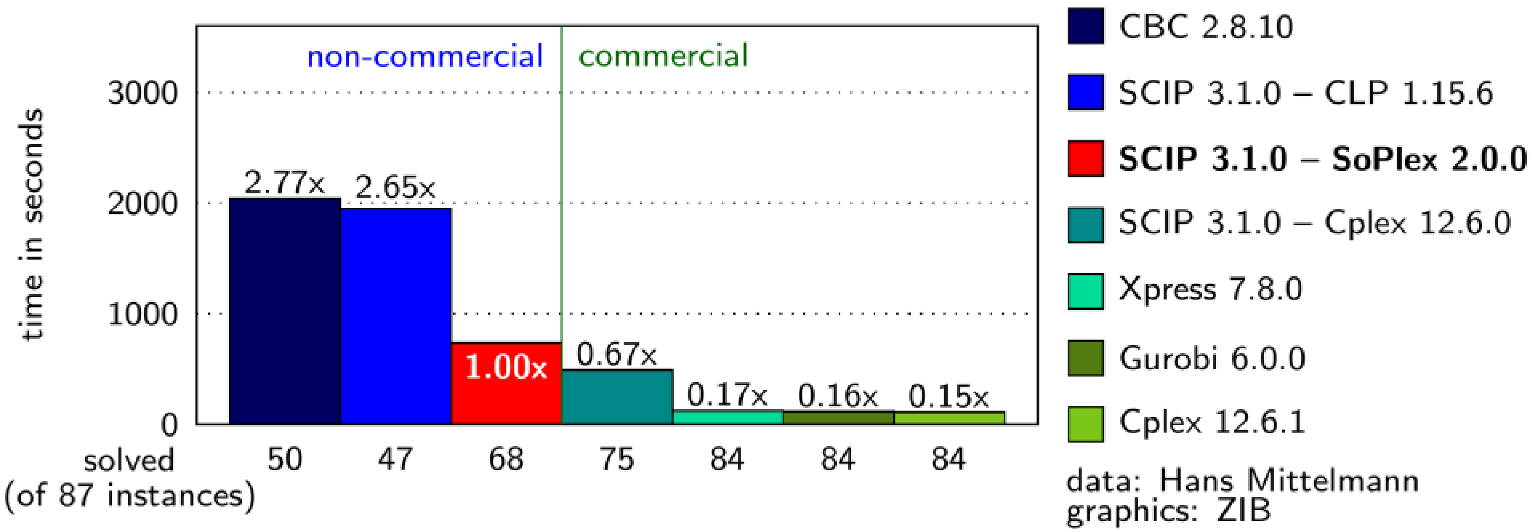
```
dvar float+ Gas;
dvar float+ Chloride;

maximize
    40 * Gas + 50 * Chloride;

subject to {
    ctMaxTotal:
        Gas + Chloride <= 50;
    ctMaxTotal2:
        3 * Gas + 4 * Chloride <= 180;
    ctMaxChloride:
        Chloride <= 40;
}
```

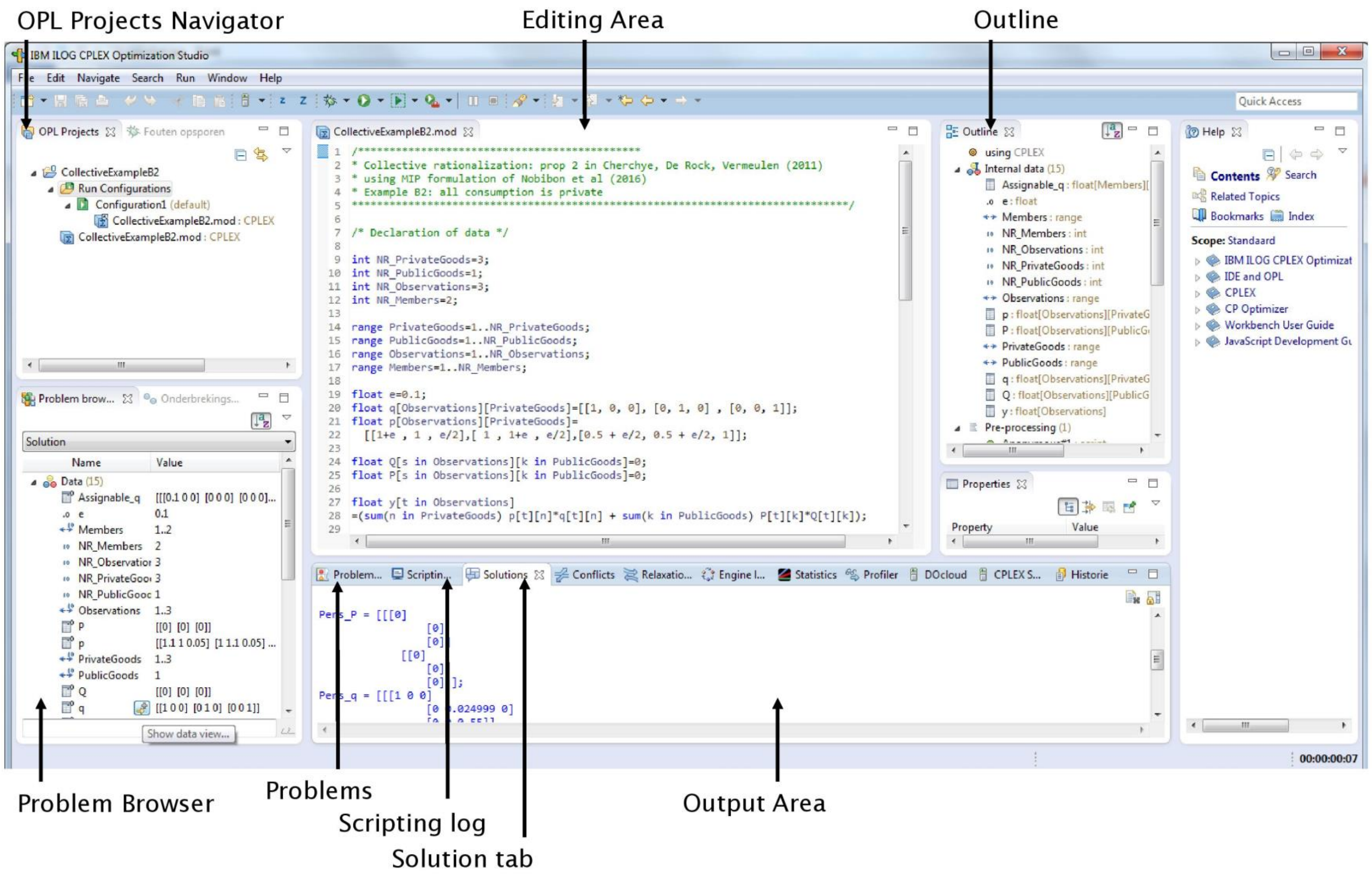
Why do we use CPLEX?

- CPLEX is a state-of-the-art commercial solver
- It is in active development and is continuously improved
- It is widely used in both academia and industry



Source: SCIP website

CPLEX Optimization studio



CPLEX interactive command-line

CPLEX Interactive

```
Welcome to IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 22.1.0.0
  with Simplex, Mixed Integer & Barrier Optimizers
5725-A06 5725-A29 5724-Y48 5724-Y49 5724-Y54 5724-Y55 5655-Y21
Copyright IBM Corp. 1988, 2022.  All Rights Reserved.

Type 'help' for a list of available commands.
Type 'help' followed by a command name for more
information on commands.

CPLEX> 
```

Describing an optimization problem

- OPL is a domain-specific language, created for describing optimization problems.

- What must we define?
  - **Constants/parameters** used in the problem.
  - **Decision variables** used in the problem.
  - **Objective function:** linear objective function.
  - **Constraints:** linear inequalities defining the feasible region.

Simple type declaration

- **Constants/ parameters**
  - float
  - float+
  - int
  - int+
  - string
- **Decision variables**
  - dvar float
  - dvar float+
  - dvar int
  - dvar int+

```
int+ n;
```

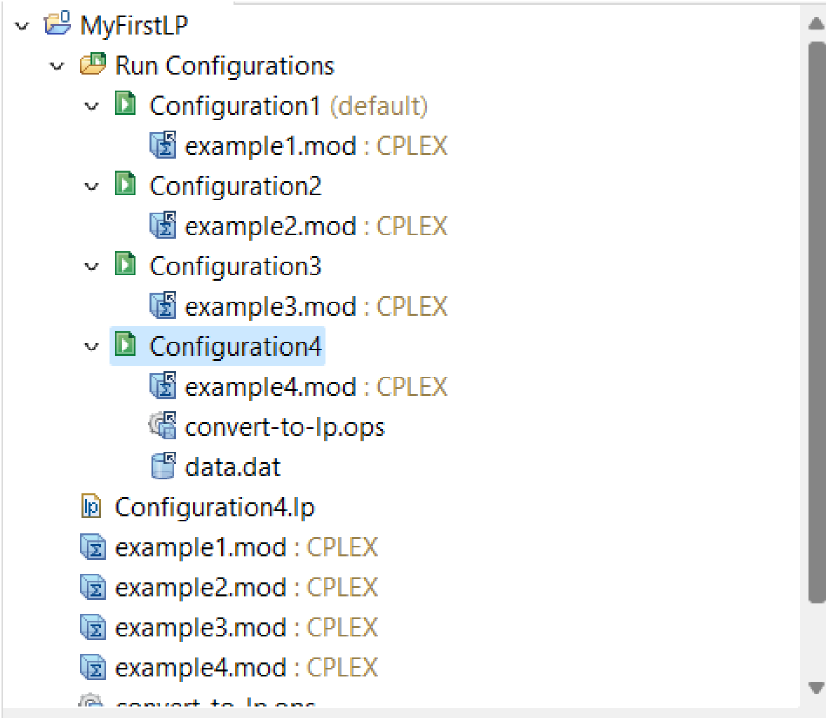
```
dvar float+ Gas;
```

Data structures in CPLEX

- Sets:
  - `{string}  subs = {"vitamin A", "calories"};`
  - `{string} foods = {"corn", "milk", "bread"};`
- We can define parameter arrays (single and multi-dimensional):
  - `cost[foods] = [1.80, 2.30, 0.50];`
  - `contents[foods][subs] = [[107,72],[500,121],[0,65]];`
  - `maxcontent[subs] = [50000, 2250];`
- Decision Variable:
  - `dvar float+ amount[foods];`

OPL Project Structure

1. Configuration file:
   - Specifies the solver and other settings for the OPL project.
2. Model file:
   - Contains the mathematical optimization model written in OPL.
3. Data file(s):
   - Data input for the OPL model.
   - It includes parameters, decision variables, and constraints.
4. Settings file:
   - Additional settings and options for the OPL project.



Representing a problem

- OPL separates the model and its instance.
  - Model: **.mod** extension, describes the structure of a problem.
  - Instance: **.dat** extension (or can be baked into .mod), describes the data in a problem.

- Any linear program has the same structure. Only the data changes!

- In the OPL IDE, a model and data file are associated in a run configuration.

Model structure in OPL

```
Variables/constants;

Maximize expression;

Subject to  {
     Constraint 1;
     Constraint 2;
     ..
     Constraint n;
}
```

A simple production problem

- Products: Ammoniac gas (NH_3) and ammonium chloride (NH_4Cl)
- Limited Stocks:
  - 50 units of nitrogen (N), 180 units of hydrogen (H), 40 units of chlorine (Cl)
- Profit:
  - 40 Euros profit for each sale of an ammoniac gas unit
  - 50 Euros profit for each sale of an ammonium chloride unit
- Objective: Maximize Profits
- Constraints: Utilize available stocks.

LP formulation

$\mathbf{Maximize}\ \mathrm{z} = 40\ Gas + 50\ Chloride$

$\boldsymbol{Subject\ to}$

$$Gas + Chloride \leq 50 \quad \text{(Nitrogen)}$$

$$3\ Gas + 4\ Chloride \leq 180 \quad \text{(Hydrogen)}$$

$$Chloride \leq 40 \quad \text{(chlorine)}$$

Example (**example.mod**)

```
dvar float+ Gas;
dvar float+ Chloride;

maximize
        40 * Gas + 50 * Chloride;

subject to {
        ctMaxTotal:
                Gas + Chloride <= 50;
        ctMaxTotal2:
                3 * Gas + 4 * Chloride <= 180;
        ctMaxChloride:
                Chloride <= 40;
}
```

Example (sets and arrays)

```
{string} Products = {"gas","chloride"};
dvar float production[Products];

maximize
       40 * production["gas"] + 50 * production["chloride"];

subject to {
           production["gas"] +     production["chloride"] <= 50;
       3 * production["gas"] + 4 * production["chloride"] <= 180;
           production["chloride"] <= 40;
}
```

Compact form (scripting language)

```
{string} Products = {"gas" , "chloride"};
{string} Components =  { "nitrogen" , "hydrogen" , "chlorine" };

float Demand[Products][Components] = [ [1 , 3 , 0], [1 , 4 , 1] ];
float Profit[Products] = [80 , 40];
float Stock[Components] = [50 , 180 , 40];

dvar float+ Production[Products];

maximize
        sum( p in Products ) Profit[p] * Production[p];

subject to {
        forall( c in Components )
                ct:
                sum( p in Products )
                        Demand[p][c] * Production[p] <= Stock[c];
}
```

Compact forms

- `sum( p in Products ) Profit[p] * Production[p];`

- ```
  forall(c in Components )
        sum( p in Products )
              Demand[p][c] * Production[p] <= Stock[c];
  ```
  ```

Data file (.dat)

- Data file "**.dat**"

```
Products = {"gas" , "chloride"};
Components = { "nitrogen" , "hydrogen" , "chlorine" };
Demand = [ [1 , 3 , 0], [1 , 4 , 1] ];
Profit = [80 , 40];
Stock = [50 , 180 , 40];
```

Use of data file

```
{string} Products = ...;
{string} Components = ...;

float Demand[Products][Components] = ...;
float Profit[Products] = ...;
float Stock[Components] = ...;
dvar float+ Production[Products];

maximize
        sum( p in Products ) Profit[p] * Production[p];

subject to {
        forall( c in Components )
                ct:
                sum( p in Products )
                        Demand[p][c] * Production[p] <= Stock[c];
}
```

Interactive CPLEX

- Enter problem / Read LP file
- Optimize
- Display Details
  - Problem details (variables, objective, constraints...)
  - Solutions
- Post-optimality details
  - Sensitivity analysis
  - Reduced cost
  - Dual price

Example

```
Maximize
        obj1: 80 Production("gas") + 40 Production("chloride")

Subject To
        ct("nitrogen"):   Production("gas") +   Production("chloride") <= 50
        ct("hydrogen"): 3 Production("gas") + 4 Production("chloride") <= 180
        ct("chlorine"):                         Production("chloride") <= 40
End
```

Enter the problem

```
CPLEX> enter
Enter name for problem: example
Enter new problem ['end' on a separate line terminates]:
Maximize
        obj1: 80 Production("gas") + 40 Production("chloride")

Subject To
        ct("nitrogen"):   Production("gas") +   Production("chloride") <= 50
        ct("hydrogen"): 3 Production("gas") + 4 Production("chloride") <= 180
        ct("chlorine"):                         Production("chloride") <= 40
End
```

Optimize

```
CPLEX> optimize
Version identifier: 22.1.0.0 | 2022-03-09 | 1a383f8ce
Tried aggregator 1 time.
LP Presolve eliminated 1 rows and 0 columns.
Reduced LP has 2 rows, 2 columns, and 4 nonzeros.
Presolve time = 0.00 sec. (0.00 ticks)

Iteration log . . .
Iteration:     1   Dual infeasibility =            50.000000
Iteration:     3   Dual objective     =          4000.000000

Dual simplex - Optimal:  Objective =  4.0000000000e+03
Solution time =    0.00 sec.  Iterations = 3 (2)
Deterministic time = 0.00 ticks  (4.38 ticks/sec)
```

Display Details

```
CPLEX> display

Display Options:

auxiliary          display auxiliary information used during optimization
conflict           display the conflict that demonstrates problem infeasibility
problem            display problem characteristics
sensitivity        display sensitivity analysis
settings           display parameter settings
solution           display existing solution

Display what:
```

Conclusion

- CPLEX/OPL is a powerful framework for optimization problems.
- It promotes code organization and data separation for efficient problem management.
- CPLEX/OPL can be applied to a diverse range of applications.
- It is particularly useful for solving large-scale optimization problems.
- With its robust capabilities, CPLEX/OPL is a valuable tool for tackling complex optimization tasks.