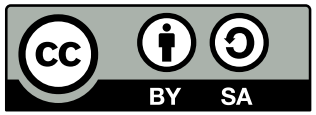
Modeling and Optimization with OPL
2 Introduction to OPL

Andreas Popp



Content

About CPLEX

- solver (software with implementent solution methods) for optimization problems distributed since 1988
- first sold by CPLEX Optimization Inc., then ILOG, then sold to IBM
- widely spread in science and industry
- interfaces for well known programming languages like C++, Java or C#
- own modeling language: OPL (Optimization Programming Language)
- cost free for academic purposes as part of the IBM Academic Iniative

2.1 Structure of an OPL project

Types of OPL files

model files description of a generic optimization model (extension: `.mod`)

data files data for instantiation of an OPL model (extension: `.dat`)

settings files settings for the solver (extension: `.ops`)

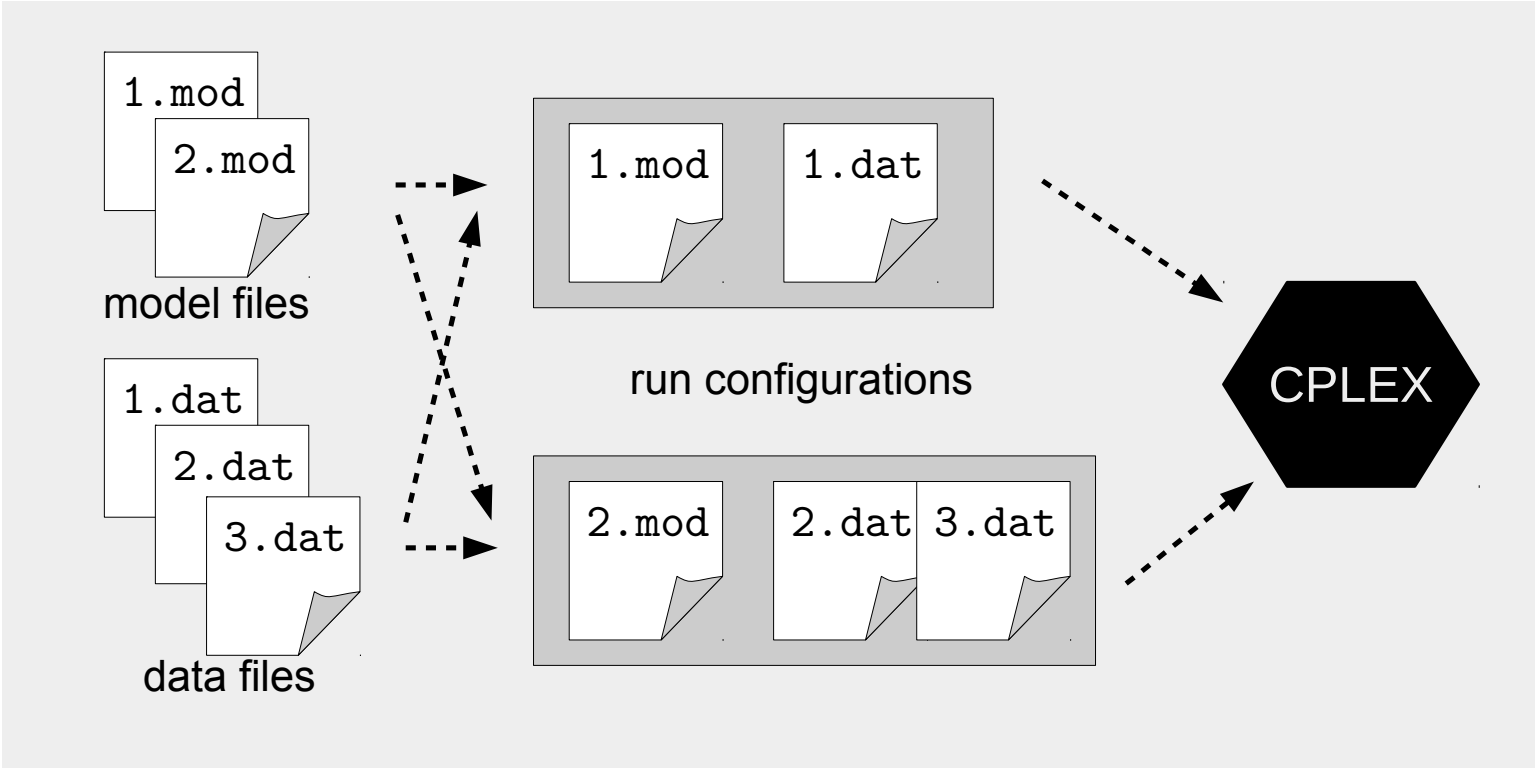
Types of OPL files

model files description of a generic optimization model (extension: `.mod`)

data files data for instantiation of an OPL model (extension: `.dat`)

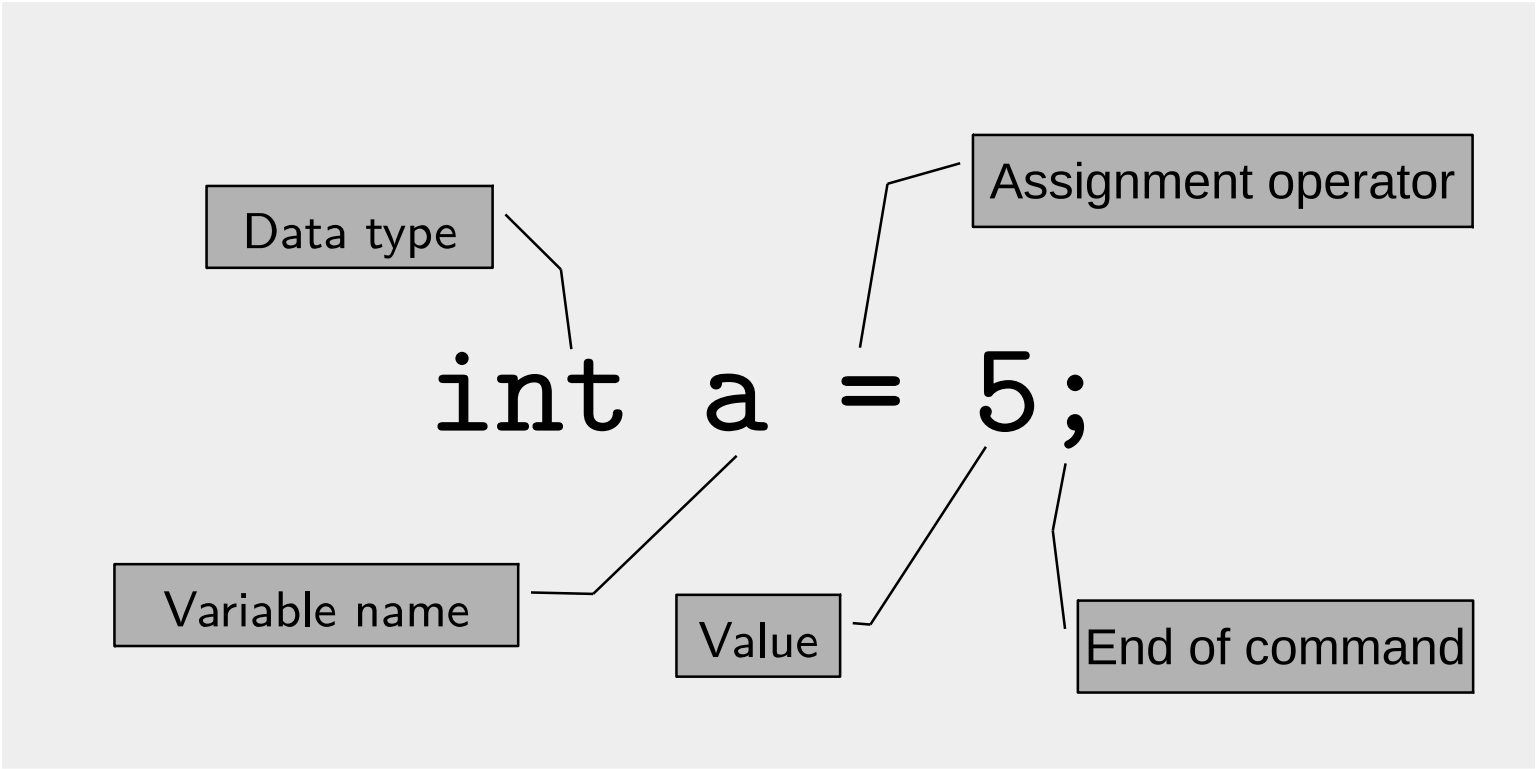
settings files settings for the solver (extension: `.ops`)

Structure of an OPL project



2.2 Basic data types and operators

Structure of a simple assignment command



Primitive data types

`int` short for: “integer”, an integer value with arbitrary sign. Example literals: `0, 1, -2, -786`

`float` floating point number with arbitrary sign. Example literals: `0.0, 1.0, 3.14, -7.86`

`boolean` technically a logical value; as decision variable a 0-1-variable.

`string` a charachter string. Example literals: `"1", "B", "Berlin"`

Derived data types

Set an ordered set of elements of (i.a.) primitve data types, e.g.

```
{string} Locations =
 {"Ansbach", "Berlin", "Cottbus"};
```

Array a tuple of (i.a.) primitive data type, sets and other arrays indexed over a set, e.g.

```
float Fixcosts[Locations] =
 [27.4, 58.3, 30.0];
```

access by index, z.B.: `Fixcosts["Cottbus"]` → `30.0`

Derived data types: mutiple arrays

- Arrays can be nested into one another to represent multiple indexes, e.g.

```
float Entf[Locations][Locations] =
  [[0.0, 5.05, 4.89],
  [5.05, 0.0, 1.22],
  [4.89, 1.22, 0.0]];
```

- Mapping rule: from left to right, from outer to inner

Simple Operators

- assignment operator =
- arithmetic operators
  - + addition
  - - subtraction
  - * multiplication
  - / division (rare in linear models)
- comparison operator (for linear models)
  - == equal
  - <= less or equal
  - >= greater or equal

Indexed operators

- sum operator

$$\sum_{i \in I} \ldots \rightarrow \texttt{sum(i in I)(...)}$$

- universal quantifier

$$\forall i \in I \rightarrow \texttt{forall(i in I)}$$

2.3 Mathematical models in OPL syntax

Example: Production problem – index sets

Mathematical model

Index sets:

I set of products

R set of ressources

Model file

```
//Index sets
{string} I = ...; //products
{string} R = ...; //ressources
```

Data file

```
//Index sets
I = {"product 1", "product 2", "product 3"};
R = {"machine A", "machine B"};
```

Example: Production problem – parameters

Mathematical model

Parameters:

p_i price of product $i \in I$

c_r capacity of ressource $r \in R$

v_{ri} capacity consumption of product $i \in I$ on ressource $r \in R$

Model file

```
//Parameter
float p[I] = ...; //price
float c[R] = ...; //capacity
float v[R][I] = ...; //capacity consumption
```

Example: Production problem – parameters

Mathematical model

Parameters:

p_i price of product $i \in I$

c_r capacity of ressource $r \in R$

v_{ri} capacity consumption of product $i \in I$ on ressource $r \in R$

Data file

```
//Parameters
p = [2.9, 3.3, 2.2];
c = [64.0, 48.0];
v = [
  [5.3, 2.9, 2.5],
  [3.9, 4.8, 3.1]
];
```

Example: Production problem – decision variables

Mathematical model

Decision variables:

x_i production quantity of product $i \in I$

$$[\ldots]$$

$x_i \geq 0 \quad \forall i \in I$

Model file

```
//Decision variables
dvar float+ x[I]; //production quantity
```

Example: Production problem – objective function

Mathematical model

$$\max \quad \sum_{i \in I} p_i \cdot x_i$$

Model file

```
//objective function
maximize sum(i in I)(p[i]*x[i]);
```

Example: Production problem – constraints

Mathematical model

$$\text{s.t.} \quad \sum_{i \in I} v_{ri} \cdot x_i \leq c_r \quad \forall r \in R$$

Model file

```
//constraints
subject to{

  //capacity constraints
  forall(r in R)
    sum(i in I)(v[r,i]*x[i]) <= c[r];

}
```

Example: Produktionsproblem.mod

```
1  //index sets
2  {string} I = ...; //products
3  {string} R = ...; //ressources
4
5  //parameters
6  float p[I] = ...; //price
7  float c[R] = ...; //capacity
8  float v[R][I] = ...; //capacity consumption
9
10 //decision variables
11 dvar float+ x[I]; //production quantity
12
13 //objective function
14 maximize sum(i in I)(p[i] * x[i]);
15
16 //constraints
17 subject to{
18
19   //capacity constraints
20   forall(r in R)
21     sum(i in I)(v[r][i]*x[i]) <= c[r];
22
23 }
```

Beispiel: LewigSanstetten.dat

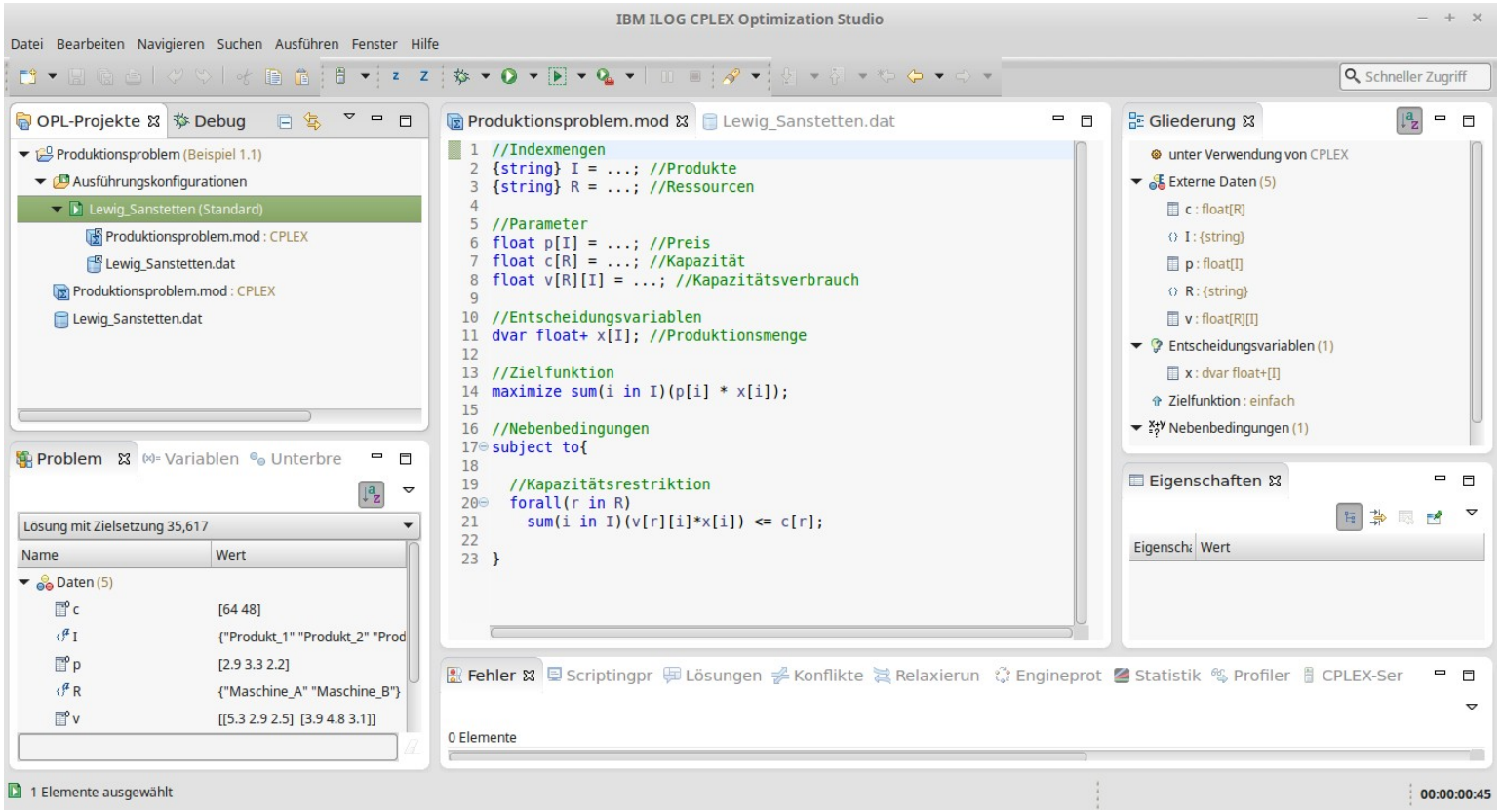
```
//index sets
I = {"product_1", "product_2", "product_3"};
R = {"machine_A", "machine_B"};

//parameters
p = [2.9, 3.3, 2.2];
c = [64.0, 48.0];
v = [
   [5.3, 2.9, 2.5],
   [3.9, 4.8, 3.1]
];
```

Solution of model instance

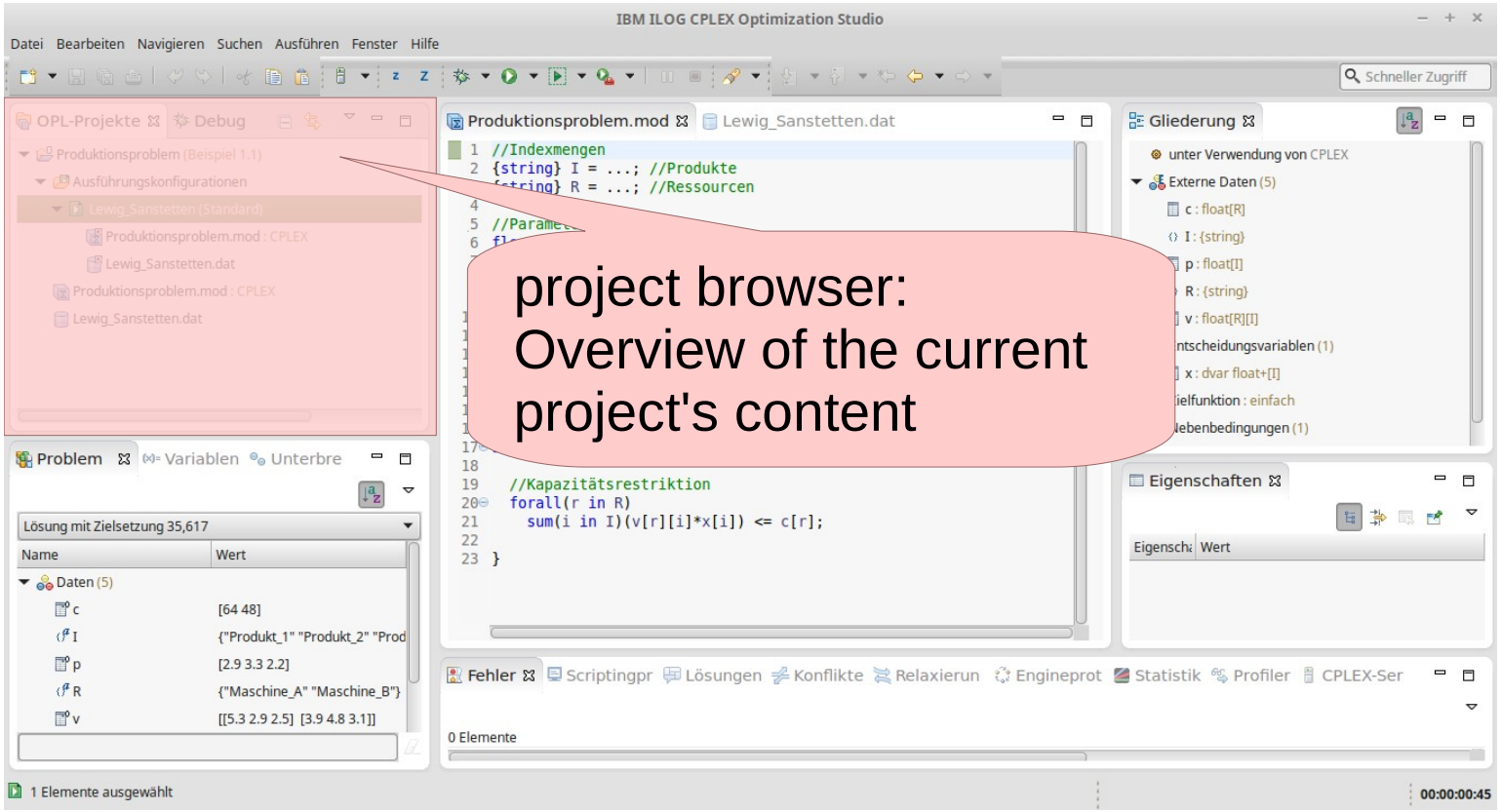
```
> oplrun -v Produktionsproblem.mod
LewigSanstetten.dat

                              ...

OBJECTIVE: 35.61677                    ← optimal value

                              ...

x = [11.737 0 0.71856];                ← optimal solution
```

2.4 The CPLEX Studio IDE

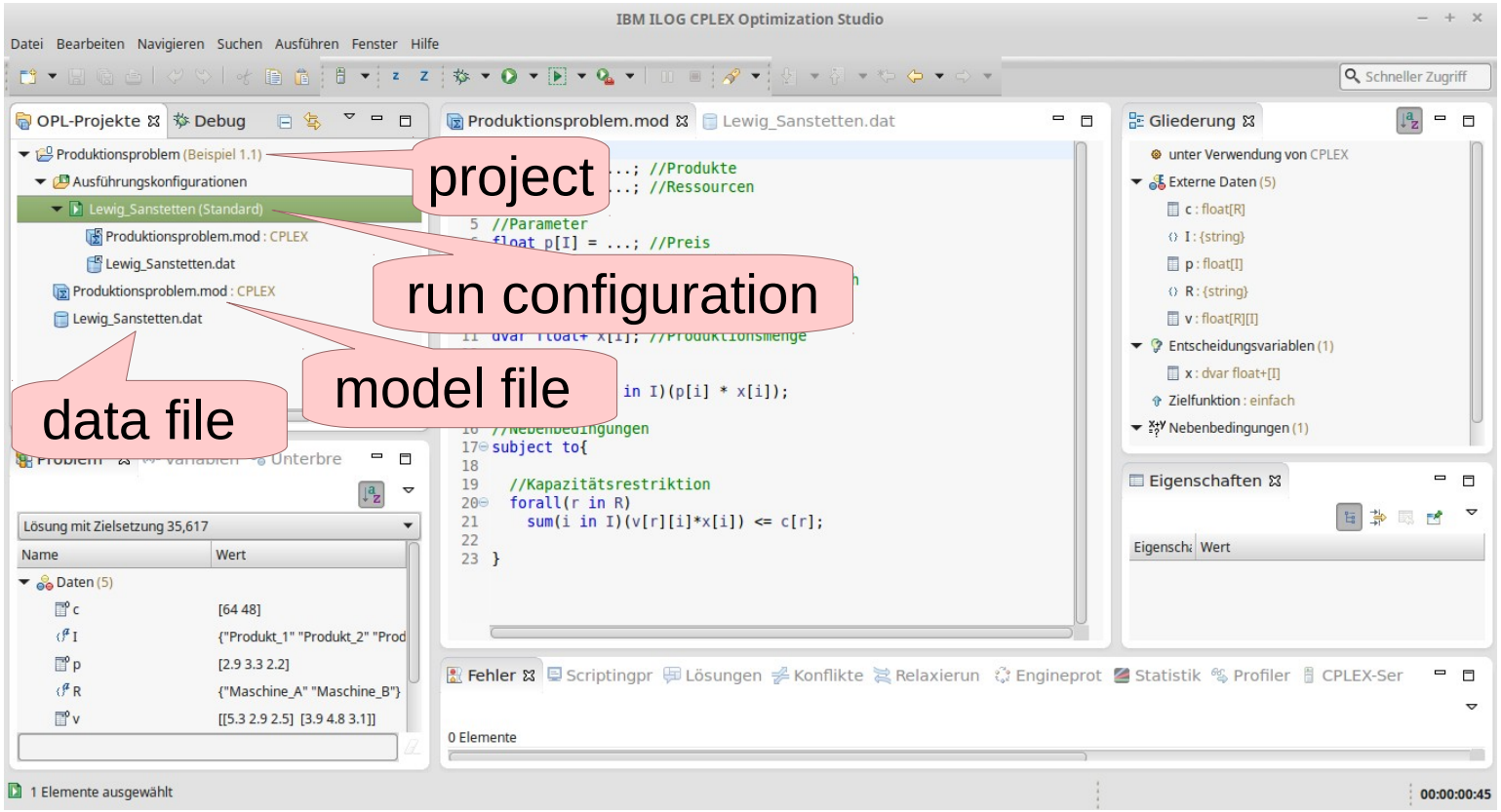
The CPLEX Studio IDE



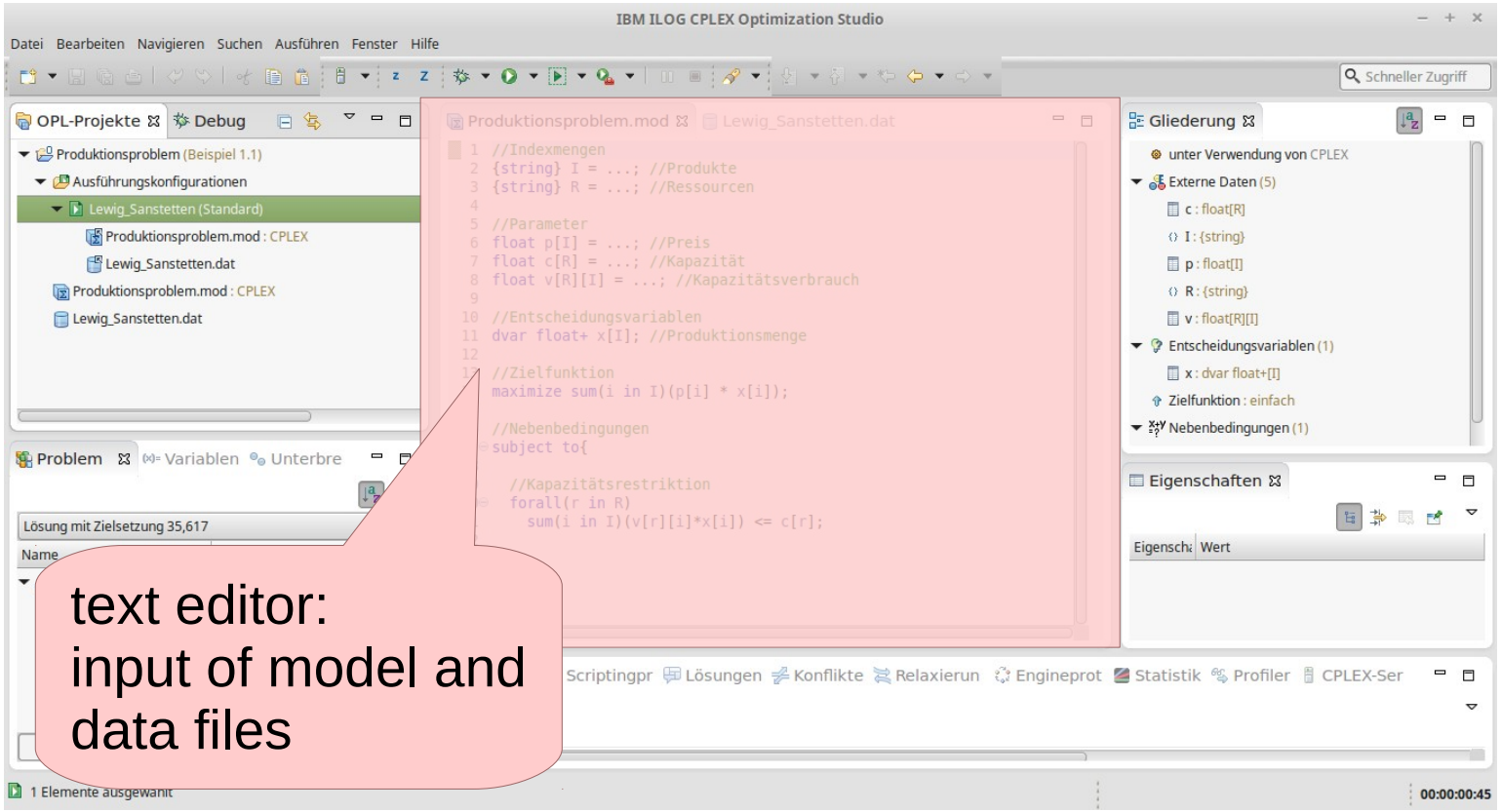
The CPLEX Studio IDE



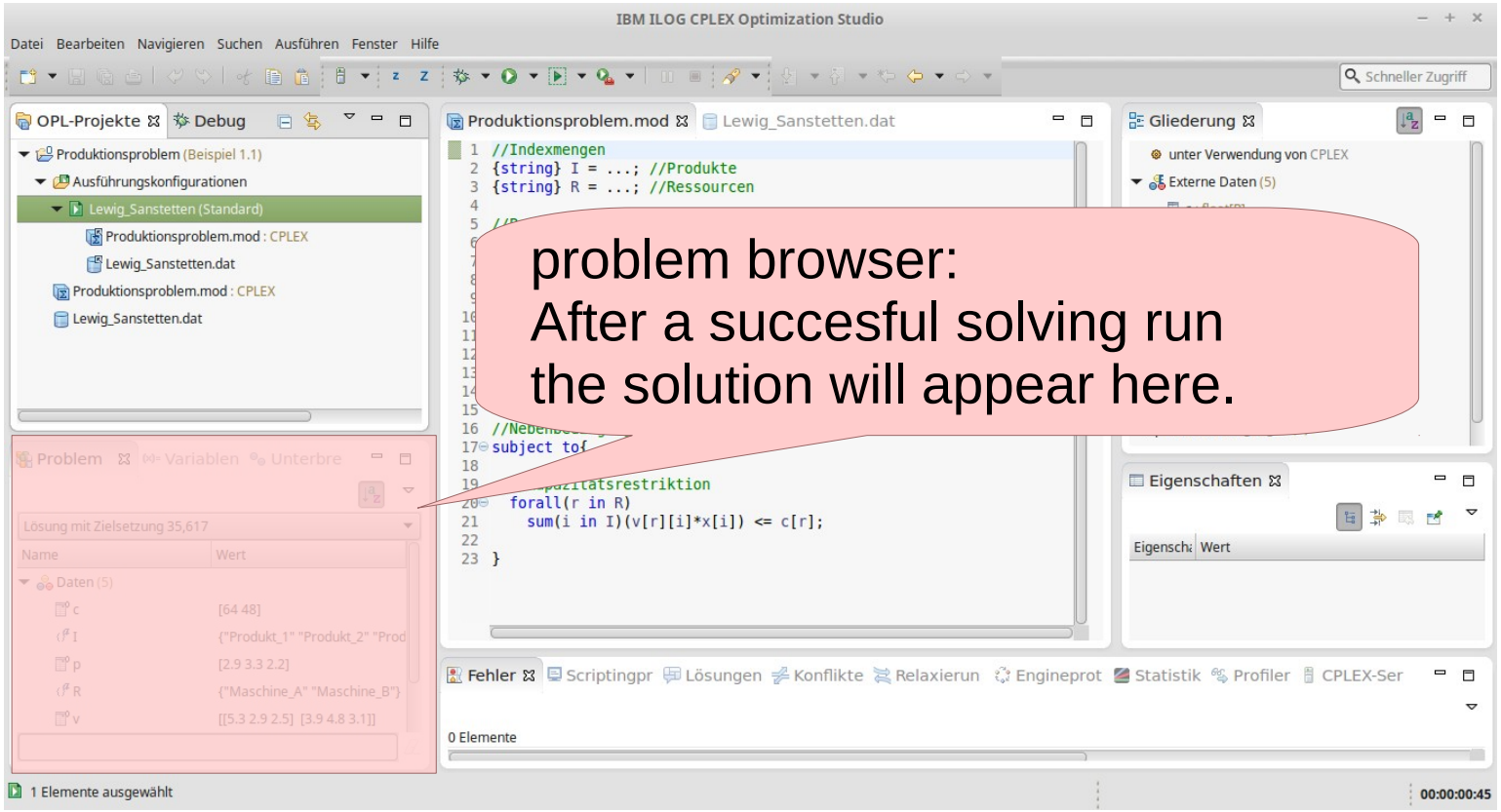
The CPLEX Studio IDE



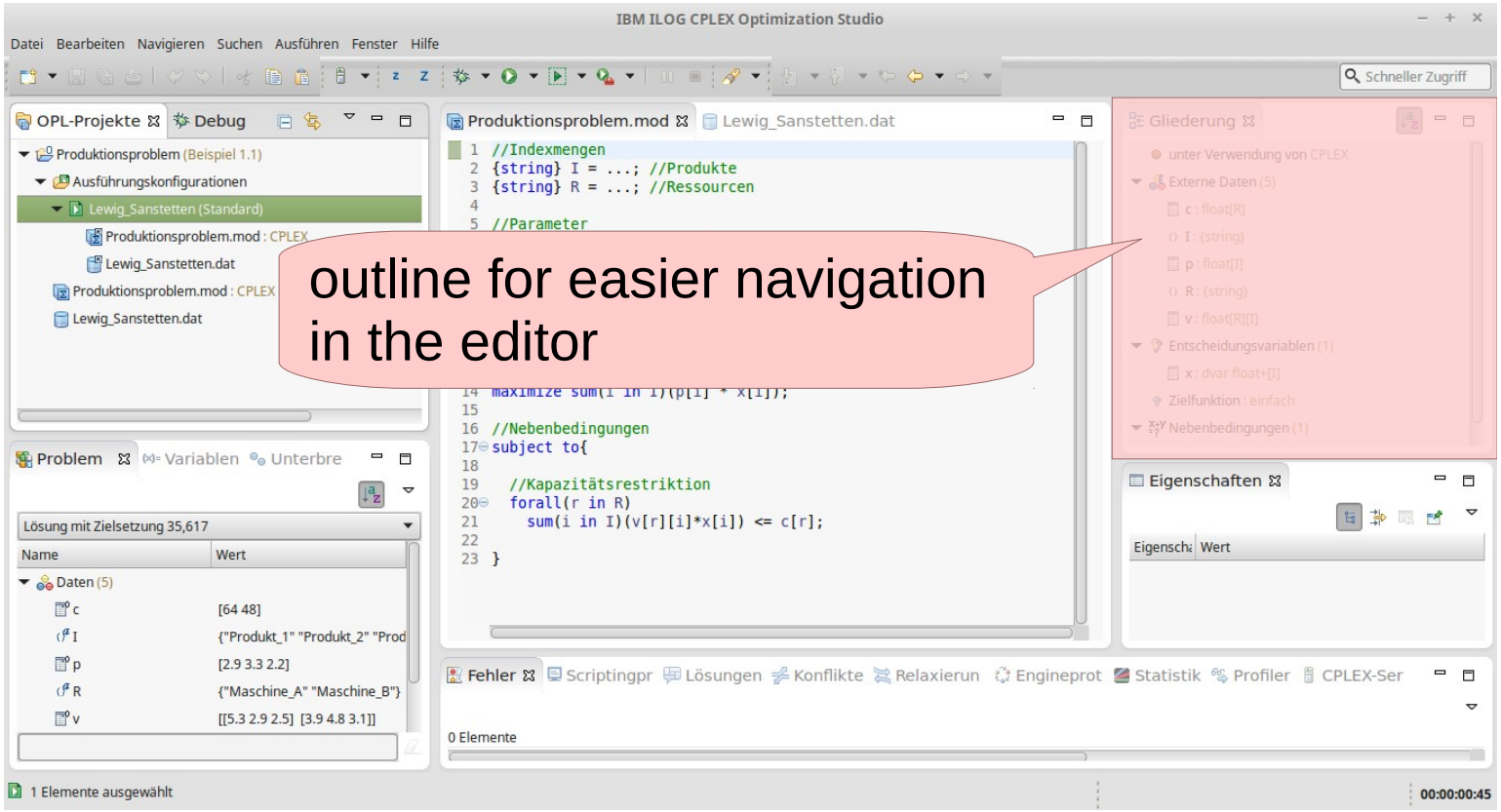
The CPLEX Studio IDE



The CPLEX Studio IDE



The CPLEX Studio IDE



2.5 Errors and warnings in OPL

Types of errors

Classification by severity

Error prevents the succesful completion of the solution run

Warning does not prevent the solution run, but can cause unexpected results. Sometimes clue to mistakes in the code.

Classification by time of occurence

compiler errors occur during the translation of the problem for the solver. Will be recognized by the IDE.

runtime errors occur during solver runtime. Will not be recognized by the IDE but displayed after a solution run.

Common error messages during first tries

- `syntax error, unexpected ...` (compiler error)
  - Compiler does not understand the command after „unexpected“ here
  - missing semi-colon?
- `syntax error, unexpected =` (compiler error)
  - special case of above
  - often mix-up of the assignment operator = and the comparison operator ==
- `The type ... cannot be used for ...` (compiler error)
  - data type mix-up
- `index out of bound for array ...` (runtime error)
  - tried to access an array with a wrong index value