7.1 Optimization routine in OPL

7.2 Data sources

Excel spreadsheets

Data bases

7.3 OPL in programming sequences

Command line with data sources

The Concert-API, example: Java

ILOG Script

Modeling and optimization with OPL

7 OPL interfaces to other applications

Andreas Popp

Inhalt

7.1 Optimization routine in OPL

7.2 Data sources

Excel spreadsheets

Data bases

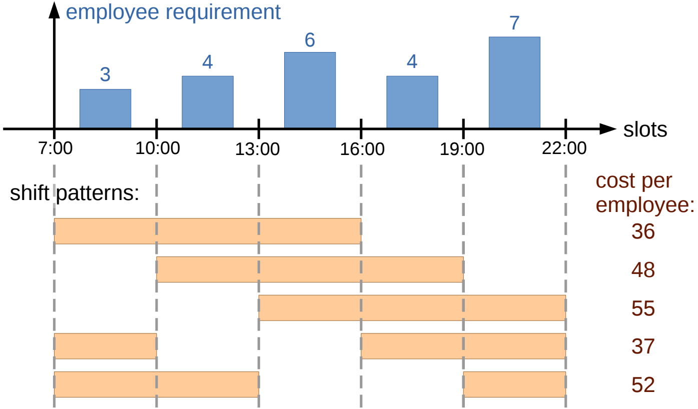
7.3 OPL in programming sequences

Command line with data sources

The Concert-API, example: Java

ILOG Script

Example: Vindoo Support



(cf. Pinedo: Planning and Scheduling in Manufacturing and Services)

7.1 Optimization routine in OPL
7.2 Data sources
Excel spreadsheets
Data bases
7.3 OPL in programming sequences
Command line with data sources
The Concert-API, example: Java
ILOG Script

Model: Cyclic staffing problem

index sets:

T set of time slots

S set of shift patterns

parameters:

c_s cost per employee in shift pattern $s \in S$

d_t requirement of employees in shift pattern $t \in T$

a_{ts} availability of employees in shift pattern $s \in S$ in time slot $t \in T$

decision variables:

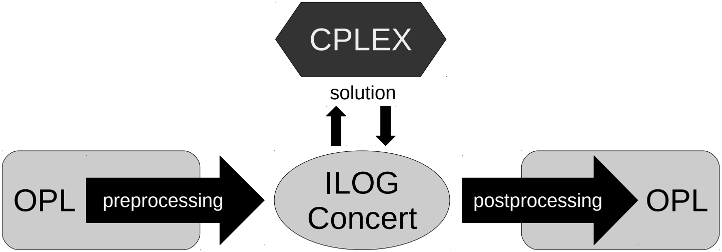
x_s deployed employees in shift pattern $s \in S$

model description:

$$\begin{aligned} \min \quad & \sum_{s\in S} c_s \cdot x_s \\ s.t. \quad & \sum_{s\in S} a_{ts} \cdot x_s \geq d_t && \forall t \in T \quad \text{(I)} \\ & x_s \in \mathbb{Z}_0^+ && \forall s \in S \end{aligned}$$

7.1 Optimization routine in OPL

Optimization routine in OPL



7.2 Data sources

Building a `SheetConnection`

The object data type `SheetConnection` connects to an Excel spreadsheet (MS Windows only)

Construction of a `SheetConnection`

`SheetConnection` *name*`("`*path to spreadsheet*`");`

Possible data types for reading and writing

- zero-dimensional data types, i.e. single variables. Reading data from one cell.
- one-dimensional data types, i.e. sets and single arrays. Reading data from a row or column.
- two-dimensional data types, i.e. double arrays. Reading data from a cell matrix.

Reading and writing with absolute cell adressing

Reading with absolute cell adressing

variable name `from SheetRead(`*SheetConnection name*`,`
`"`*table name*`!`*starting cells*`:`*ending cell*`")`

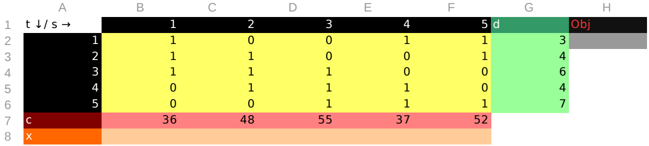
Writing with absolute cell adressing

variable name `to SheetWrite(`*SheetConnection name*`,`
`"`*table name*`!`*starting cell*`:`*ending cell*`")`

7.1 Optimization routine in OPL
7.2 Data sources
Excel spreadsheets
Data bases
7.3 OPL in programming sequences
Command line with data sources
The Concert-API, example: Java
ILOG Script

In example “Vindoo Support”

Excel spreadsheet for example “Vindo Support”



	A	B	C	D	E	F	G	H
1	t ↓/ s →	1	2	3	4	5	d	Obj
2	1	1	0	0	1	1	3	
3	2	1	1	0	0	1	4	
4	3	1	1	1	0	0	6	
5	4	0	1	1	1	0	4	
6	5	0	0	1	1	1	7	
7	c	36	48	55	37	52		
8	x							

Excerpt from the data file

```
// SheetConnection
SheetConnection sheet("CyclicStaffingProblem.xls");
```

```
// index sets
S from SheetRead(sheet, "Data!B1:F1");
```

```
//parameters
d from SheetRead(sheet, "Data!G2:G6");
```

```
//decision variables
x to SheetWrite(sheet, "Data!B8:F8");
```

Reading and writing with named ranges

In MS Excel it is possible to name cell ranges.

Reading with named ranges

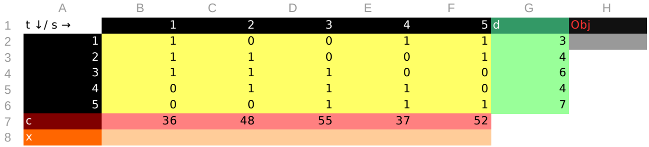
variable name `from SheetRead(`*SheetConnection name*`,`
`"`*range name*`")`

Writing with named ranges

variable name `to SheetWrite(`*SheetConnection name*`,`
`"`*range name*`")`

In example “Vindoo Support”

Excel spreadsheet for example “Vindo Support”



	A	B	C	D	E	F	G	H
1	t ↓/ s →	1	2	3	4	5	d	Obj
2	1	1	0	0	1	1	3	
3	2	1	1	0	0	1	4	
4	3	1	1	1	0	0	6	
5	4	0	1	1	1	0	4	
6	5	0	0	1	1	1	7	
7	c	36	48	55	37	52		
8	x							

The yellow range is named “`ParamA`”

Excerpt from the data file

```
// SheetConnection
SheetConnection sheet("CyclicStaffingProblem.xls");
```

```
//parameters
a from SheetRead(sheet, "ParamA");
```

Building a DBConnection

The object data type DBConnection connects to a data base.

Construction of a DBConnection

```
DBConnection name("interface", "connection string");
```

Supported data base interfaces

- DB2
- Oracle (versions 10 and 11)
- OLE DB (MS SQL Server)
- ODBC (i.a. for MS Access)

example data base: CyclicStaffingProblem

s 🔑	t 🔑	a
1	1	1
1	2	1
1	3	1
1	4	0
1	5	0
2	1	0
2	2	1
2	3	1
2	4	1
2	5	0
3	1	0
3	2	0
3	3	1
3	4	1
3	5	1
4	1	1
4	2	0
4	3	0
4	4	1
4	5	1
5	1	1
5	2	1
5	3	0
5	4	0
5	5	1

(a) table A

ind 🔑	d
1	2
2	4
3	6
4	4
5	7

(b) table T

ind 🔑	c	x
1	36	
2	48	
3	55	
4	37	
5	52	

(c) table S

Reading array data

The example data base CyclicStaffingProblem shall be connected via an ODBC interface. The user “user” with the password “password” has the necessary rights to access the data base.

Excerpt from the data file

```
// DBConnection
DBConnection
  db("odbc", "CyclicStaffingProblem/user/password");
```

```
// index sets
T from DBRead (db, "SELECT ind from T");
```

```
// parameters
c from DBRead(db, "SELECT ind,c from S");
a from DBRead(db, "SELECT t,s,a from A");
```

Reading and writing of tuple data

Excerpt from the model file

```
//Tuple
tuple shift{
  int ind;
  float c;
}
tuple result{<
  int x;
  int ind;
}
```

```
//Postprocessing
{result} r = {<x[s],s.ind>|s in S};
```

Excerpt from the data file

```
//index sets
S from DBRead(db, "SELECT ind,c from S");
```

```
//decision variables
r to DBUpdate(db, "UPDATE S SET x=? WHERE ind=?");
```

7.1 Optimization routine in OPL

7.2 Data sources

Excel spreadsheets

Data bases

7.3 OPL in programming sequences

Command line with data sources

The Concert-API, example: Java

ILOG Script

7.3 OPL in programming sequences

Example: cyclic staffing problem

Algorithmic solutions for the cyclic staffing problem

1. Let P_1 be the LP relaxation of the cylic staffing problem. Solve P_1, if solution is integer end, otherwise go to 2.
2. Add the constraint
$$\sum_{s \in S} x_s = \left\lfloor \sum_{s \in S} x_s^* \right\rfloor$$
to P_1 to get P_2. If P_2 has a solution end, otherwise go to 3.
3. Add the constraint
$$\sum_{s \in S} x_s = \left\lceil \sum_{s \in S} x_s^* \right\rceil$$
to P_1 to get P_3. An optimal solution of P_3 is always an optimal solution for the cyclic staffing problem.

Integrating OPL/CPLEX in programming sequences

Interfaces

- command line applications, especially `oplrun`
- ILOG Concert-API
- CPLEX Callable Library
- CPLEX-Interfaces
- ILOG Script

Integrating OPL/CPLEX in programming sequences

Interfaces

- command line applications, especially `oplrun`
- ILOG Concert-API
- CPLEX Callable Library
- CPLEX-Interfaces
- ILOG Script

Selection of applications

- automated construction and solution of model instances
- reading values for decision variables after solution
- adding automatically generated data to a model instance

Preparations in the cyclic staffing problem example

Add the following constraints:

$$\sum_{s \in S} x_s \leq ub$$

$$\sum_{s \in S} x_s \geq lb$$

If both bounds have the same value, the effective result is:

$$\sum_{s \in S} x_s = ub = lb$$

Advantages and disadvantages of the command line application

Advantages

- useable with any programming language, which supports command line applications
- quite versatile
- well suited for academic research and commercial prototypes

Advantages and disadvantages of the command line application

Advantages

- useable with any programming language, which supports command line applications
- quite versatile
- well suited for academic research and commercial prototypes

Disadvantages

- has to be combined with given data source interfaces or data input and output has to be coded seperately
- susceptible to error in platform configuration
- not well suited for productive systems

Example: VBA

Listings excerpt: command invocation

```
'Solve P1
Shell ("oplrun " & modPath & " " & datPath)
```

Listings excerpt: Workaround for empty solution space

```
'Solution not integer
'prepare step 2
dataSheet.Range("Obj").Value = "n"
```

7.1 Optimization routine in OPL
7.2 Data sources
Excel spreadsheets
Data bases
7.3 OPL in programming sequences
Command line with data sources
The Concert-API, example: Java
ILOG Script

Class packages

ilog.concert classes providing interfaces to ILOG Concert

ilog.cplex classes providing interfaces to ILOG CPLEX

ilog.cp classes providing interfaces to ILOG CP Optimizer

ilog.opl classes providing interfaces to OPL

Important classes

IloOplFactory the central class for constructing other OPL objects

IloOplModelSource a model file in the file system

IloOplErrorHandler pipeline for output of OPL errors and warnings (standard: System.out)

IloOplModelDefintion internal representation of the model

IloCplex represents an instance of the CPLEX solver

IloOplModel a problem instance

IloOplDataSource data source for a model instance

Instantiating and solving models

Example: Cyclic-Staffing-Problem

```java
IloOplFactory oplF = new IloOplFactory();

IloOplModelSource modelSource =
  oplF.createOplModelSource("CyclicStaffingProblem.mod");

IloOplErrorHandler err = oplF.createOplErrorHandler();
IloOplModelDefinition def = oplF.createOplModelDefinition(
  modelSource, oplF.createOplSettings(err));

IloCplex cplex = oplF.createCplex();

IloOplModel opl = oplF.createOplModel(def, cplex);

IloOplDataSource dataSource =
 oplF.createOplDataSource("CyclicStaffingProblem.dat");
opl.addDataSource(dataSource);

opl.generate();

opl.getCplex().solve();

opl.printSolution(System.out);
```

Accesing model elements

- access with class IloOplElement and method getElement(String s).
- data type translation with as-methods

OPL data type	Concert data type keyword	Java data type
int	Int	int
float	Num	double
string	Symbol	String
Set	Set	
Array	Map	
tuple	Tuple	
range	Range	
dvar	Var	
dexpr	Expr	

7.1 Optimization routine in OPL
7.2 Data sources
Excel spreadsheets
Data bases
7.3 OPL in programming sequences
Command line with data sources
The Concert-API, example: Java
ILOG Script

Define custom data sources

- abstract class `IloCustomOplDataSource`
- data delivery via `customRead`

Example: cyclic staffing problem

```java
@Override
public void customRead() {
  //Initialize Datahandler object
  IloOplDataHandler handler = getDataHandler();

  //deliver ub
  handler.startElement("ub");
  handler.addIntItem(this.ub);
  handler.endElement();

  //deliver lb
  handler.startElement("lb");
  handler.addIntItem(this.lb);
  handler.endElement();
}
```

7.1 Optimization routine in OPL
7.2 Data sources
Excel spreadsheets
Data bases
7.3 OPL in programming sequences
Command line with data sources
The Concert-API, example: Java
ILOG Script

ILOG Script

- ILOG Script is an extension of JavaScript
- based on Concert
- Scripts are written directly into to the model file
- execute blocks can be used in pre- and postprocessing
  - simplified syntax
  - OPL variables can be used like script variables
- The main block serves as procedure control. Here we can create and and solve problem instances.