Constraint Programming

Lab 1. Introduction to OPL

Ruslan Sadykov

INRIA Bordeaux—Sud-Ouest

10 February 2022

Lignes directrices

Software overview

OPL Development Studio

Lab : steel mill inventory matching problem

Software to solve CSPs

- Commercial
  - **IBM ILOG CP Optimizer** (free for academic use), interfaces : *C++*, *Java*, *IBM ILOG CPLEX Optimization Studio (OPL)*.
  - **Artelys Kalis**, interfaces : *C++*, *Java*, *Xpress-Mosel*.
- Free
  - **Google OR-Tools CP-SAT solver**, interface : *C++, Python*.
  - **MiniZinc** (modeling language), interface : *MiniZincIDE*.
  - **Choco Solver**, interface : *Java*.
  - **Gecode**, interface : *C++*.
  - **PyCSP3**[3], interface : *Python*

Others : `http://www.constraint.org/en/tools/`

Software to solve CSPs

- Commercial
  - **IBM ILOG CP Optimizer** (free for academic use), interfaces : *C++*, *Java*, *IBM ILOG CPLEX Optimization Studio (OPL)*.
  - **Artelys Kalis**, interfaces : *C++*, *Java*, *Xpress-Mosel*.
- Free
  - **Google OR-Tools CP-SAT solver**, interface : *C++, Python*.
  - **MiniZinc** (modeling language), interface : *MiniZincIDE*.
  - **Choco Solver**, interface : *Java*.
  - **Gecode**, interface : *C++*.
  - **PyCSP3**[3], interface : *Python*

Others : `http://www.constraint.org/en/tools/`

Software to solve CSPs

- Commercial
  - **IBM ILOG CP Optimizer** (free for academic use), interfaces : *C++*, *Java*,
    IBM ILOG CPLEX Optimization Studio (OPL).
  - **Artelys Kalis**, interfaces : *C++*, *Java*, *Xpress-Mosel*.
- Free
  - **Google OR-Tools CP-SAT solver**, interface : *C++, Python*.
  - **MiniZinc** (modeling language), interface : *MiniZincIDE*.
  - **Choco Solver**, interface : *Java*.
  - **Gecode**, interface : *C++*.
  - **PyCSP3**[3], interface : *Python*

Others : `http://www.constraint.org/en/tools/`

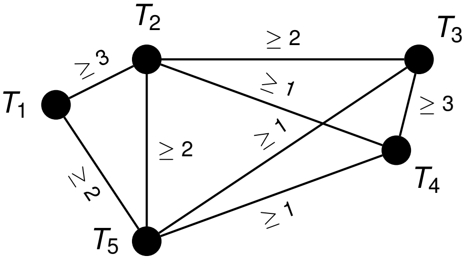
Software to solve CSPs

- Commercial
  - **IBM ILOG CP Optimizer** (free for academic use), interfaces : *C++*, *Java*, *IBM ILOG CPLEX Optimization Studio (OPL)*.
  - **Artelys Kalis**, interfaces : *C++*, *Java*, *Xpress-Mosel*.
- Free
  - **Google OR-Tools CP-SAT solver**, interface : *C++, Python*.
  - **MiniZinc** (modeling language), interface : *MiniZincIDE*.
  - **Choco Solver**, interface : *Java*.
  - **Gecode**, interface : *C++*.
  - **PyCSP3**[3], interface : *Python*

Others : `http://www.constraint.org/en/tools/`

Example : frequency assignment

- Variables : F_i — frequency assigned to transmitter i.
- Additional variables : $S_i = 0$ if low frequency 1 if high frequency.
- Constraints :
  - $\mid F_i - F_j \mid \geq d_{ij}, \forall (i,j)$;
  - $\texttt{all-different}(F_1, \ldots, F_5)$.
- Additional constraints :
  - $\texttt{element}(S_i, \{0,0,0,1,1,1,1\}, F_i), \forall i$.
  - $\texttt{gcc}(\{S_i\}_{\forall i}, \{0,1\}, 2, 3, 2, 3)$.



Lignes directrices

Software overview

OPL Development Studio

Lab : steel mill inventory matching problem

IBM ILOG CPLEX Optimization Studio

`https://www.ibm.com/products/ilog-cplex-optimization-studio`

(Windows, Linux, Mac OS, ...)

No-cost academic edition is available

Data and variables declaration

```
/*********************************************
 * OPL 6.0.1 Model
 * File : frequencies.mod
 *********************************************/

using CP; //!!!!!

int nbFreqs = ...;
int nbTrans = ...;
range Freqs = 1..nbFreqs;
range Trans = 1..nbTrans;
int Diffs[Trans,Trans] = ...;
int BasseHaute[Freqs] = ...;

dvar int F[Trans] in Freqs;
dvar int S[Trans] in 0..1;
```

Data file

```
/*********************************************
 * OPL 6.0.1 Data
 * File : frequencies.dat
 *********************************************/

nbFreqs = 7 ;
nbTrans = 5 ;
Diffs = [[0 3 0 0 2]
         [3 0 2 1 2]
         [0 2 0 3 1]
         [0 1 3 0 1]
         [2 2 1 1 0]] ;
BasseHaute = [0 0 0 1 1 1 1] ;
```

Objective and constraints declaration

```
minimize max(t in Trans) F[t];
subject to {
  forall (ordered t1, t2 in Trans : Diffs[t1,t2] > 0)
      abs( F[t1] - F[t2] ) >= Diffs[t1,t2];
  allDifferent(F);
  forall (t in Trans)
      S[t] == element(BasseHaute,F[t]);
  count(S,0) == 2;
  count(S,1) == 3;
}

execute {
  for (var t=1; t<=nbTrans; t++)
      writeln("F["+t+"]="+F[t]);
}
```

Find all solutions

```
main {
  thisOplModel.generate() ;
  cp.startNewSearch() ;
  var n=0 ;
  while (cp.next()) {
      n = n+1 ;
      write("Solution -> ") ;
      writeln(n) ;
      for (var t=1 ; t<=thisOplModel.nbTrans ; t++)
          writeln("\t F["+t+"]="+thisOplModel.F[t]) ;
  }
  cp.endSearch() ;
}
```

Some constraints available in OPL

- Arithmétiques
  (on peut utilisez `min, max, count, abs, element`).
- Logiques
  (`&&, ||, !, =>, !=, ==`).
- Explicites
  (`allowedAssignments, forbiddenAssignments`).
- Pour l'ordonnancement
  (`endBeforeStart, endAtStart, noOverlap, ...`)
- Specialisées
  (`allDifferent, allMinDistance, inverse, lex, pack`)

Declaration of heuristics

```
execute {
  var fc = cp.factory;
  var phase1 = fc.searchPhase(F,
      fc.selectSmallest(fc.varIndex(F)),
      fc.selectLargest(fc.value()));
  cp.setSearchPhases(phase1);
}
```

Variable evaluations :

```
varIndex(dvar int[])
domainSize()
domainMin()
regretOnMin()
successRate()
impact()
...
```

Value evaluations :

```
value()
valueImpact()
valueSuccessRate()
explicitValueEval(int[],int[])
valueIndex(int[])
```

Declaration of heuristics

```
execute {
  var fc = cp.factory;
  var phase1 = fc.searchPhase(F,
      fc.selectSmallest(fc.varIndex(F)),
      fc.selectLargest(fc.value()));
  cp.setSearchPhases(phase1);
}
```

Variable evaluations :

```
varIndex(dvar int[])
domainSize()
domainMin()
regretOnMin()
successRate()
impact()
...
```

Value evaluations :

```
value()
valueImpact()
valueSuccessRate()
explicitValueEval(int[],int[])
valueIndex(int[])
```

Solver parameters

```
execute {
  var p = cp.param;
  p.logPeriod = 10000 ;
  p.searchType = "DepthFirst";
  p.timeLimit = 600 ;
}
```

Options :

AllDiffInterenceLevel	Low, Basic, Medium, Extended
CountInferenceLevel	Low, Basic, Medium, Extended
ElementInferenceLevel	Low, Basic, Medium, Extended
BranchLimit	<number>
TimeLimit	<number>(in seconds)
LogVerbosity	Quiet, Terse, Normal, Verbose
PropagationLog	Quiet, Terse, Normal, Verbose
SearchType	depthFirst, Restart, MultiPoint

Solver parameters

```
execute {
  var p = cp.param;
  p.logPeriod = 10000 ;
  p.searchType = "DepthFirst" ;
  p.timeLimit = 600 ;
}
```

Options :

`AllDiffInterenceLevel`	`Low, Basic, Medium, Extended`
`CountInferenceLevel`	`Low, Basic, Medium, Extended`
`ElementInferenceLevel`	`Low, Basic, Medium, Extended`
`BranchLimit`	`<number>`
`TimeLimit`	`<number>(in seconds)`
`LogVerbosity`	`Quiet, Terse, Normal, Verbose`
`PropagationLog`	`Quiet, Terse, Normal, Verbose`
`SearchType`	`depthFirst, Restart, MultiPoint`

Using OPL in CREMI

1. Download and unarchive the model for frequency assignment problem :

   `www.math.u-bordeaux.fr/~rsadykov/`
   `teaching/MSE3315C/TP/frequencies.zip`

2. Run OPLIDE
3. In OPLIDE :
   - Make new project (File → New... → OPL project)
   - Copy downloaded files (.mod and .dat) to project (File → Copy Files to Project...)
   - Create a run configuration (File → New... → Run configuration)
   - Add .mod and .dat files to the new configuration (drag them there)
   - Run the configuration (right click on configuration → Run this)
4. Documentation : Help → Help Contents

Lignes directrices

Software overview

OPL Development Studio

Lab : steel mill inventory matching problem

Problem description

- A steel mill has an inventory of steel slabs of different sizes.
- From these steel slabs, we need to manufacture different types of steel coils.
- Every steel coil type requires a different production process encoded by a color.
- Every order is characterized by the weight and color of the steel coil.
- Every steel slab can be used for production of steel coils of at most two different colors.
- The total weight of steel coils produced from the same steel slab cannot exceed its capacity (or size).
- The objective is to minimize to total loss (unused capacity of steel slabs).

Practical information

Necessary files are available at

```
www.math.u-bordeaux1.fr/~rsadykov/
teaching/MSE3315C/TP/stillmill.zip
```

Help

- **or** : IDE and OPL > OPL > Language Quick Reference > OPL keywords > or
- **dexpr** : IDE and OPL > OPL > Language Quick Reference > OPL keywords > dexpr
- **pack** : IDE and OPL > OPL > Language Quick Reference > OPL functions > pack

Or you can just search them

Practical information

Necessary files are available at

```
www.math.u-bordeaux1.fr/~rsadykov/
teaching/MSE3315C/TP/stillmill.zip
```

Help

- **or** : IDE and OPL > OPL > Language Quick Reference > OPL keywords > or
- **dexpr** : IDE and OPL > OPL > Language Quick Reference > OPL keywords > dexpr
- **pack** : IDE and OPL > OPL > Language Quick Reference > OPL functions > pack

Or you can just search them