

## Introduction à la Programmation par Contraintes

### Cours 6. Logiciels pour résoudre des CSPs

Ruslan Sadykov

INRIA Bordeaux—Sud-Ouest

21 Novembre 2011

- ▶ Payants
  - ▶ **IBM ILOG CP Optimizer**, interfaces : *C++*, *Java*, *IBM ILOG CPLEX Optimization Studio*.
  - ▶ **Artelys Kalis**, interfaces : *C++*, *Java*, *Xpress-Mosel*.
  - ▶ **Microsoft Solver Foundation**, interfaces : CLS-compliant languages (*C++*, *IronPython*, ...) *Add-in Designer for Excel 2007*.
  - ▶ **Comet** (gratuit pour un usage académique), interface : un langage spécial (type *C*)
- ▶ Gratuits
  - ▶ **Choco Solver**, interface : *Java*.
  - ▶ **MINION**, interfaces : un fichier texte spécial, ESSENSE'.
  - ▶ **Gecode**, interface : *C++*.

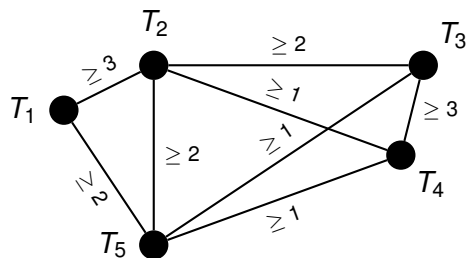
D'autres sont mentionnés sur Wikipédia (article « Programmation par contraintes »).

1/11

2/11

## Exemple : affectation des fréquences

- ▶ Variables :  $F_i$  — fréquence affectée à l'émetteur  $i$ .
- ▶ Variables additionnelles :  $S_i = 0$  si basse et 1 si haute.
- ▶ Contraintes :
  - ▶  $|F_i - F_j| \geq d_{ij}, \forall (i, j)$ ;
  - ▶  $\text{all-different}(F_1, \dots, F_5)$ .
- ▶ Contraintes additionnelles :
  - ▶  $\text{element}(S_i, \{0, 0, 0, 1, 1, 1, 1\}, F_i), \forall i$ .
  - ▶  $\text{gcc}(\{S_i\}_{\forall i}, \{0, 1\}, 2, 3, 2, 3)$ .



## IBM ILOG CPLEX Optimization Studio

Trial version :

<http://www-01.ibm.com/software/integration/optimization/cplex-cp-optimizer/>

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

3/11

4/11

## Déclaration des données et variables

```
/* *****  
 * OPL 6.0.1 Model  
 * File : frequenciest.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 ;
```

5/11

## Le fichier avec les données

```
/* *****  
 * OPL 6.0.1 Data  
 * File : frequenciest.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] ;
```

6/11

## Déclaration de la objective et des contraintes

```
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]) ;  
}
```

7/11

## Trouver toutes les 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() ;  
}
```

8/11

## Les contraintes existantes dans OPL

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

9/11

## Déclaration des heuristiques

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

Évaluateurs des variables :

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

Évaluateurs des valeurs :

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

10/11

## Paramétrage du solveur

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

Les options :

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

11/11