Model Presolve, Warmstart and Conflict Refining in CP Optimizer

P. Laborie, P. Refalo, P. Shaw
IBM Software Group

Sept. 16, 2013
Overview

- A “model-and-run” optimizer must provide:
  - Efficient & robust automatic search
  - Easy-to-use functionalities to control the search
  - Productivity tools for model development
Overview

- A “model-and-run” optimizer must provide:
  CP Optimizer is a “model-and-run” optimizer providing:
  
  - Efficient & robust automatic search
    Restart, Learning, Portfolios, Relaxations, LNS, Presolve, ...

  - Easy-to-use functionalities to control the search
    Parameters, Search phases, Warmstart, ...

  - Productivity tools for model development
    Development Studio, Search log, Conflict Refiner, ...
Model Presolve

- Depending on the input data, models can have redundancies and can contain poorly formulated constraints

- Objective: Automatically improve the model in order to make stronger inferences faster

- Simplifications
  - Constraint compaction
  - Redundancy elimination
  - Constant propagation
  - Common sub-expression factorization

- Aggregations
  - Count expressions & difference constraints
  - Linear constraints over binary \{0,1\} variables
  - Alternative constraint combined with arithmetic expressions
Examples of Model Presolve

- Elimination of redundant constraints
  - Bound of variables and expressions computed by constraint propagation are used to eliminate redundant constraints before search

- Propagation of constant variables and expressions
  \[ 1 + 2x + 3y + xy - 3x + 7y \leq 8x - 7y + 10 \]
  \[ x == 4 \]
  \[ 21y \leq 45 \]

- Variable merge
  \[ x==y, y==z, z==t \rightarrow \text{merge the domains of } x, y, z \text{ and } t \text{ and replace } y, z, \text{ and } t \text{ by } x \text{ everywhere} \]
Examples of Model Presolve

- Aggregation of difference constraints
  - Binary $\neq$ constraints can be lifted to alldiff constraints

A problem with $\neq$

\[
\begin{align*}
x \neq y \\
x \neq z \\
x \neq t \\
y \neq z \\
y \neq t \\
z \neq v \\
t \neq z \\
t \neq v \\
u \neq v
\end{align*}
\]

The associated graph structure

The aggregated model

\[
\begin{align*}
alldiff(x, y, z, t) \\
alldiff(z, t, v) \\
u \neq v
\end{align*}
\]
Examples of Model Presolve

- Lifting difference constraints
  1) Pick up a binary constraint \( x \neq y \) not already in a lifted alldiff
  2) Find the largest clique containing \( x \neq y \) (in practice we use a greedy algorithm – add vertex one by one to the set starting with vertices having the maximum degree)
  3) Add a new lifted alldiff

- Experimentation on graph coloring
  - Color the vertices of a graph such that two adjacent vertices have a different color and the number of colors is minimized
  - The model is stated with a set of binary \( \neq \) constraints
  - Problems are from the DIMACS challenge (COLOR02 set)
    - 9 problems over 60 are solved to optimality without presolve
    - 39 problems over 60 are solved to optimality with presolve
## Results on graph coloring problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Time</th>
<th>Ch. Pts</th>
<th>Time</th>
<th>Ch. Pts</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>anna</td>
<td>1000.01</td>
<td>19308059</td>
<td>0.08</td>
<td>138</td>
<td>11</td>
</tr>
<tr>
<td>david</td>
<td>141.49</td>
<td>3628877</td>
<td>0.06</td>
<td>87</td>
<td>11</td>
</tr>
<tr>
<td>fpsol2.i.1</td>
<td>1000.01</td>
<td>6353782</td>
<td>4.11</td>
<td>496</td>
<td>65</td>
</tr>
<tr>
<td>fpsol2.i.2</td>
<td>1000.01</td>
<td>8731886</td>
<td>2.03</td>
<td>451</td>
<td>30</td>
</tr>
<tr>
<td>fpsol2.i.3</td>
<td>1000.01</td>
<td>8980436</td>
<td>1.90</td>
<td>425</td>
<td>30</td>
</tr>
<tr>
<td>games120</td>
<td>1000.01</td>
<td>20346758</td>
<td>0.08</td>
<td>120</td>
<td>9</td>
</tr>
<tr>
<td>homer</td>
<td>1000.02</td>
<td>10225244</td>
<td>0.05</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>huck</td>
<td>1000.01</td>
<td>2822739</td>
<td>0.05</td>
<td>74</td>
<td>11</td>
</tr>
<tr>
<td>inithx.i.1</td>
<td>1000.02</td>
<td>4744285</td>
<td>14.18</td>
<td>864</td>
<td>54</td>
</tr>
<tr>
<td>inithx.i.2</td>
<td>1000.01</td>
<td>5741171</td>
<td>10.06</td>
<td>645</td>
<td>31</td>
</tr>
<tr>
<td>inithx.i.3</td>
<td>1000.01</td>
<td>5671424</td>
<td>10.07</td>
<td>621</td>
<td>31</td>
</tr>
<tr>
<td>jean</td>
<td>1000.01</td>
<td>26779859</td>
<td>0.05</td>
<td>80</td>
<td>10</td>
</tr>
<tr>
<td>le450.15a</td>
<td>1000.02</td>
<td>3623748</td>
<td>8.41</td>
<td>1062</td>
<td>15</td>
</tr>
<tr>
<td>le450.15b</td>
<td>1000.02</td>
<td>6650123</td>
<td>4.70</td>
<td>847</td>
<td>15</td>
</tr>
<tr>
<td>le450.15c</td>
<td>1000.01</td>
<td>5723362</td>
<td>100.82</td>
<td>630981</td>
<td>15</td>
</tr>
<tr>
<td>le450.15d</td>
<td>1000.02</td>
<td>1457738</td>
<td>100.79</td>
<td>49765</td>
<td>15</td>
</tr>
<tr>
<td>le450.25a</td>
<td>1000.01</td>
<td>7289172</td>
<td>3.52</td>
<td>450</td>
<td>25</td>
</tr>
<tr>
<td>le450.25b</td>
<td>1000.02</td>
<td>7355812</td>
<td>3.25</td>
<td>450</td>
<td>25</td>
</tr>
<tr>
<td>le450.25c</td>
<td>1000.02</td>
<td>1969772</td>
<td>100.58</td>
<td>45443</td>
<td>25</td>
</tr>
<tr>
<td>le450.25d</td>
<td>1000.01</td>
<td>2313797</td>
<td>100.67</td>
<td>52443</td>
<td>25</td>
</tr>
<tr>
<td>le450.5a</td>
<td>1000.02</td>
<td>3009929</td>
<td>100.37</td>
<td>114321</td>
<td>5</td>
</tr>
<tr>
<td>le450.5b</td>
<td>1000.01</td>
<td>2379044</td>
<td>100.37</td>
<td>107031</td>
<td>5</td>
</tr>
<tr>
<td>le450.5c</td>
<td>1.48</td>
<td>522</td>
<td>10.17</td>
<td>450</td>
<td>5</td>
</tr>
<tr>
<td>le450.5d</td>
<td>1000.02</td>
<td>1635272</td>
<td>100.75</td>
<td>69670</td>
<td>5</td>
</tr>
<tr>
<td>miles1000</td>
<td>1000.01</td>
<td>1172583</td>
<td>0.47</td>
<td>128</td>
<td>42</td>
</tr>
<tr>
<td>miles1500</td>
<td>1000.01</td>
<td>9798720</td>
<td>100.02</td>
<td>136613</td>
<td>71</td>
</tr>
<tr>
<td>miles250</td>
<td>0.42</td>
<td>6961</td>
<td>0.05</td>
<td>128</td>
<td>8</td>
</tr>
<tr>
<td>miles500</td>
<td>1000.01</td>
<td>15628706</td>
<td>0.13</td>
<td>128</td>
<td>20</td>
</tr>
<tr>
<td>miles750</td>
<td>1000.01</td>
<td>13420578</td>
<td>0.27</td>
<td>128</td>
<td>31</td>
</tr>
<tr>
<td>mug100 1</td>
<td>1000.01</td>
<td>20055287</td>
<td>5.95</td>
<td>91443</td>
<td>3</td>
</tr>
<tr>
<td>mug88 1</td>
<td>1000.01</td>
<td>22220677</td>
<td>5.44</td>
<td>92818</td>
<td>3</td>
</tr>
<tr>
<td>mug88_25</td>
<td>1000.01</td>
<td>22069362</td>
<td>1.19</td>
<td>20104</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Problem</th>
<th>Time</th>
<th>Ch. Pts</th>
<th>Time</th>
<th>Ch. Pts</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>mulsol1.i.1</td>
<td>1000.01</td>
<td>13151679</td>
<td>0.66</td>
<td>197</td>
<td>49</td>
</tr>
<tr>
<td>mulsol1.i.2</td>
<td>1000.01</td>
<td>16062965</td>
<td>0.45</td>
<td>188</td>
<td>31</td>
</tr>
<tr>
<td>mulsol1.i.3</td>
<td>1000.01</td>
<td>16000296</td>
<td>0.45</td>
<td>184</td>
<td>31</td>
</tr>
<tr>
<td>mulsol1.i.4</td>
<td>1000.01</td>
<td>16442846</td>
<td>0.44</td>
<td>185</td>
<td>31</td>
</tr>
<tr>
<td>mulsol1.i.5</td>
<td>1000.01</td>
<td>11138968</td>
<td>0.59</td>
<td>186</td>
<td>31</td>
</tr>
<tr>
<td>myciel3</td>
<td>0.02</td>
<td>32</td>
<td>0.02</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>myciel4</td>
<td>0.09</td>
<td>2791</td>
<td>0.09</td>
<td>2791</td>
<td>0</td>
</tr>
<tr>
<td>myciel5</td>
<td>234.11</td>
<td>5734555</td>
<td>239.05</td>
<td>5734555</td>
<td>0</td>
</tr>
<tr>
<td>myciel6</td>
<td>1000.01</td>
<td>15070869</td>
<td>1000.00</td>
<td>14612561</td>
<td>0</td>
</tr>
<tr>
<td>myciel7</td>
<td>1000.01</td>
<td>10789203</td>
<td>1000.00</td>
<td>10699137</td>
<td>0</td>
</tr>
<tr>
<td>queen10_10</td>
<td>1000.02</td>
<td>7167824</td>
<td>1000.04</td>
<td>650963</td>
<td>10</td>
</tr>
<tr>
<td>queen11</td>
<td>1000.01</td>
<td>5350496</td>
<td>1000.04</td>
<td>564951</td>
<td>11</td>
</tr>
<tr>
<td>queen12_12</td>
<td>1000.01</td>
<td>4302509</td>
<td>1000.04</td>
<td>357267</td>
<td>12</td>
</tr>
<tr>
<td>queen13_13</td>
<td>1000.01</td>
<td>3613703</td>
<td>1000.08</td>
<td>259550</td>
<td>13</td>
</tr>
<tr>
<td>queen14_14</td>
<td>1000.01</td>
<td>3023318</td>
<td>1000.10</td>
<td>276221</td>
<td>14</td>
</tr>
<tr>
<td>queen15_15</td>
<td>1000.02</td>
<td>2544350</td>
<td>1000.15</td>
<td>1809290</td>
<td>15</td>
</tr>
<tr>
<td>queen16_16</td>
<td>1000.02</td>
<td>2114831</td>
<td>1000.15</td>
<td>166693</td>
<td>16</td>
</tr>
<tr>
<td>queen5_5</td>
<td>0.02</td>
<td>45</td>
<td>0.02</td>
<td>45</td>
<td>5</td>
</tr>
<tr>
<td>queen6_6</td>
<td>1.44</td>
<td>24392</td>
<td>1.45</td>
<td>7726</td>
<td>6</td>
</tr>
<tr>
<td>queen7_7</td>
<td>1.28</td>
<td>19528</td>
<td>0.09</td>
<td>111</td>
<td>7</td>
</tr>
<tr>
<td>queen8_12</td>
<td>1000.01</td>
<td>8415680</td>
<td>1000.04</td>
<td>890550</td>
<td>12</td>
</tr>
<tr>
<td>queen8_8</td>
<td>1000.01</td>
<td>11742448</td>
<td>1000.03</td>
<td>1826017</td>
<td>8</td>
</tr>
<tr>
<td>queen9_9</td>
<td>1000.01</td>
<td>9198487</td>
<td>1000.03</td>
<td>724638</td>
<td>9</td>
</tr>
<tr>
<td>school1</td>
<td>1000.02</td>
<td>1938387</td>
<td>28.52</td>
<td>543</td>
<td>14</td>
</tr>
<tr>
<td>school1_nsh</td>
<td>1000.02</td>
<td>4461519</td>
<td>1000.87</td>
<td>251760</td>
<td>14</td>
</tr>
<tr>
<td>zeroin.i.1</td>
<td>1000.01</td>
<td>11703893</td>
<td>0.55</td>
<td>211</td>
<td>49</td>
</tr>
<tr>
<td>zeroin.i.2</td>
<td>1000.01</td>
<td>11807063</td>
<td>0.47</td>
<td>211</td>
<td>30</td>
</tr>
<tr>
<td>zeroin.i.3</td>
<td>1000.01</td>
<td>12225170</td>
<td>0.50</td>
<td>206</td>
<td>30</td>
</tr>
</tbody>
</table>
Examples of Model Presolve

- Common sub-expression factorization
  - Eliminate multiple occurrences of the same expression and replace them by a new variable
    
    example: 
    
    \[
    \begin{align*}
    xy &\neq z + t \\
    z + xy &== a + b \\
    100 &\leq z + xy \\
    \end{align*}
    \]

  - The expressions \(xy\) and \(z + xy\) appear several times. We introduce two new variables \(u\) and \(v\) to replace these expressions and add the constraints: \(u == xy\), \(v == z + u\)

  - The model becomes

    \[
    \begin{align*}
    u &\neq z + t \\
    v &== a + b \\
    100 &\leq v \\
    \end{align*}
    \]

  - Communication of bound reduction on newly introduced variables achieves more domain reduction
  - It also reduces the number of expressions and thus involves less computations
Examples of Model Presolve: Golomb Ruler Example

**Golomb ruler naive formulation (in OPL)**

```plaintext
using CP;

int N = 10;
int L = ftoi(pow(2, N-1) - 1);
rangle R = 1..N;
dvar int x[R] in 0..L;

minimize x[N];
subject to {
    forall(i in 1..N-1) x[i] < x[i+1];
    forall(i in 1..N-1, j in i+1..N, k in i..N-1, l in k+1..N : l != j) {
        x[j] - x[i] != x[1] - x[k];
    }
}
```

Common sub-expression elimination regroups repeated differences `x[i] != x[j]` and introduces a new variable for each.

This allow aggregation of binary difference constraints on these variables to a single `alldifferent`

Presolve produces the well-known tight model for this problem

**Internal formulation after presolve**

```plaintext
let x0 = intVar(0..26);
let x1 = intVar(1..27);
let x2 = intVar(2..28);
let x3 = intVar(3..29);
let x4 = intVar(4..30);
let x5 = intVar(5..31);
let d1 = intVar(-25..27);
let d2 = intVar(-24..28);
let d3 = intVar(-23..29);
let d4 = intVar(-22..30);
let d5 = intVar(-21..31);
let d6 = intVar(-25..27);
let d7 = intVar(-24..28);
let d8 = intVar(-23..29);
let d9 = intVar(-22..30);
let d10 = intVar(-25..27);
let d11 = intVar(-24..28);
let d12 = intVar(-23..29);
let d13 = intVar(-25..27);
let d14 = intVar(-24..28);
let d15 = intVar(-25..27);

minimize(x5);

alldiff([d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15]);

x0 < x1;
x1 < x2;
x2 < x3;
x3 < x4;
x4 < x5;
x5;
```

© 2013 IBM Corporation
Conflict Refiner

- **Objective:** Identify a reason for an inconsistency by providing a minimal infeasible subset of constraints for an infeasible model

- **Use cases:**
  1) Model debugging (errors in model)
  2) Data debugging (inconsistent data)
  3) The model and data are correct, but the associated data represents a real-world conflict in the system being modeled
  4) You create an infeasible model to test properties of (or extract information about) a similar model
Conflict Refiner

- **Objective:** Identify a reason for an inconsistency by providing a minimal infeasible subset of constraints for an infeasible model

- **Use cases:**
  1. Model debugging (errors in model)
  2. Data debugging (inconsistent data)
  3. The model and data are correct, but the associated data represents a real-world conflict in the system being modeled
  4. You create an infeasible model to test properties of (or extract information about) a similar mode
Conflicting Refiner example: satellite scheduling problem

- USAF Satellite Control Network scheduling problem [1]
- $n$ communication requests for Earth orbiting satellites must be scheduled on a total of 32 antennas spread across 13 ground-based tracking stations
- In the instances, $n$ ranges from 400 to 1300

Communication requests
Alternative assignments to stations × time windows (opportunities)
Selected opportunity will use 1 antenna for communication with the satellite
Conflict Refiner example: model 1

```java
using CP;

tuple Station {
    string name; // Ground station name
    int id; // Ground station identifier
    int cap; // Number of available antennas
}

tuple Opportunity {
    string task; // Task
    int station; // Ground station
    int smin; // Start of visibility window of opportunity
    int dur; // Task duration in this opportunity
    int emax; // End of visibility window of opportunity
}

{Station} Stations = ...;
{Opportunity} Opportunities = ...;
{string} Tasks = { o.task | o in Opportunities };

dvar interval task[t in Tasks];
dvar interval opp[o in Opportunities] in o.smin..o.emax size o.dur;

subject to {
    forall(t in Tasks)
        opportunitySelection: alternative(task[t], all(o in Opportunities: o.task==t) opp[o]);
    forall(s in Stations)
        numberOfAntennas: sum(o in Opportunities: o.station==s.id) pulse(opp[o],1) <= s.cap;
```
Conflict Refiner example: running model 1

! Satisfiability problem - 2,980 variables, 851 constraints
! Workers = 2
! TimeLimit = 30
! Problem found infeasible at the root node
!
...

! Conflict refining - 851 constraints
!

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Number of constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>851</td>
</tr>
<tr>
<td>*</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>*</td>
<td>12</td>
</tr>
<tr>
<td>*</td>
<td>13</td>
</tr>
</tbody>
</table>

! Conflict refining terminated
!

! Conflict status : Terminated normally, conflict found
! Conflict size : 1 constraint
! Number of iterations : 13
! Total memory usage : 10.6 MB
! Conflict computation time : 0.04s
Conflict Refiner example: running model 1

- Conflict:

<table>
<thead>
<tr>
<th>Line</th>
<th>In conflict</th>
<th>Element (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>Yes</td>
<td>opportunitySelection[&quot;373A&quot;]</td>
</tr>
</tbody>
</table>

- Opportunities for task “373A”:

```value
<"373A", 2, 1191, 23, 1241>, 
<"373A", 3, 1191, 23, 1241>, 
<"373A", 11, 1191, 23, 1241>, 
<"373A", 13, 1191, 23, 1241>, 
<"373A", 5, 1191, 23, 1241>, 
<"373A", 7, 1191, 23, 1241>, 
<"373A", 8, 1191, 23, 1241>,
```
Conflict Refiner example: model 1

```plaintext
using CP;

tuple Station {
    string name; // Ground station name
    int id;     // Ground station identifier
    int cap;    // Number of available antennas
}

tuple Opportunity {
    string task; // Task
    int station; // Ground station
    int smin;   // Start of visibility window of opportunity
    int dur;    // Task duration in this opportunity
    int emax;   // End of visibility window of opportunity
}

{Station} Stations = ...;
{Opportunity} Opportunities = ...;
{string} Tasks = { o.task | o in Opportunities };

dvar interval task[t in Tasks];

dvar interval opp[o in Opportunities] in o.smin..o.emax size o.dur;

subject to {
    forall(t in Tasks)
        opportunitySelection: alternative(task[t], all(o in Opportunities: o.task==t) opp[o]);
    forall(s in Stations)
        numberOfAntennas: sum(o in Opportunities: o.station==s.id) pulse(opp[o],1) <= s.cap;
}
```
Conflict Refiner example: model 2

```plaintext
using CP;

tuple Station {
    string name; // Ground station name
    int id; // Ground station identifier
    int cap; // Number of available antennas
}

tuple Opportunity {
    string task; // Task
    int station; // Ground station
    int smin; // Start of visibility window of opportunity
    int dur; // Task duration in this opportunity
    int emax; // End of visibility window of opportunity
}

{Station} Stations = ...;
{Opportunity} Opportunities = ...;
{string} Tasks = { o.task | o in Opportunities };

dvar interval task[t in Tasks];

dvar interval opp[o in Opportunities] in o.smin..o.emax size o.dur;

subject to {
    forall(t in Tasks)
        opportunitySelection: alternative(task[t], all(o in Opportunities: o.task==t) opp[o]);
    forall(s in Stations)
        numberOfAntennas: sum(o in Opportunities: o.station==s.id) pulse(opp[o], 1) <= s.cap;
}
```
Conflict Refiner example: running model 2

Satisfiability problem - 2,980 variables, 851 constraints
Problem found infeasible at the root node

Conflict refining - 851 constraints

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Number of constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>851</td>
</tr>
<tr>
<td>2</td>
<td>426</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>5</td>
</tr>
<tr>
<td>59</td>
<td>5</td>
</tr>
</tbody>
</table>

Conflict refining terminated

Conflict status: Terminated normally, conflict found
Conflict size: 5 constraints
Number of iterations: 59
Total memory usage: 13.3 MB
Conflict computation time: 0.51s
Conflict Refiner example: running model 2

- Conflict:

<table>
<thead>
<tr>
<th>Line</th>
<th>In conflict</th>
<th>Element (5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>Yes</td>
<td>opportunitySelection[&quot;134A&quot;]</td>
</tr>
<tr>
<td>26</td>
<td>Yes</td>
<td>opportunitySelection[&quot;144&quot;]</td>
</tr>
<tr>
<td>26</td>
<td>Yes</td>
<td>opportunitySelection[&quot;146&quot;]</td>
</tr>
<tr>
<td>26</td>
<td>Yes</td>
<td>opportunitySelection[&quot;146A&quot;]</td>
</tr>
<tr>
<td>28</td>
<td>Yes</td>
<td>numberOfAntennas[&quot;LION&quot;,6,3]</td>
</tr>
</tbody>
</table>

- There is not enough antennas to accommodate all 4 tasks on their time-window on ground station “LION” (3 antennas):

<"134A", 6, 1232, 19, 1266>
<"144", 6, 1238, 31, 1272>
<"146", 6, 1228, 22, 1260>
<"146A", 6, 1230, 22, 1262>
Conflict Refiner example: model 2

```plaintext
using CP;

tuple Station {
  string name;  // Ground station name
  int id;       // Ground station identifier
  int cap;      // Number of available antennas
}

tuple Opportunity {
  string task;  // Task
  int station;  // Ground station
  int smin;     // Start of visibility window of opportunity
  int dur;      // Task duration in this opportunity
  int emax;     // End of visibility window of opportunity
}

{Station} Stations = ...;
{Opportunity} Opportunities = ...;
{string} Tasks = { o.task | o in Opportunities };

dvar interval task[t in Tasks];
dvar interval opp[o in Opportunities] optional in o.smin..o.emax size o.dur;

subject to {
  forall(t in Tasks)
    opportunitySelection: alternative(task[t], all(o in Opportunities: o.task==t) opp[o]);

  forall(s in Stations)
    numberOfAntennas: sum(o in Opportunities: o.station==s.id) pulse(opp[o],1) <= s.cap;
}
```
Conflict Refiner example: model 3

```
using CP;

tuple Station {
  string name; // Ground station name
  int id; // Ground station identifier
  int cap; // Number of available antennas
}

tuple Opportunity {
  string task; // Task
  int station; // Ground station
  int smin; // Start of visibility window of opportunity
  int dur; // Task duration in this opportunity
  int emax; // End of visibility window of opportunity
}

{Station} Stations = ...;
{Opportunity} Opportunities = ...;
{string} Tasks = { o.task | o in Opportunities };

dvar interval task[t in Tasks] optional;

dvar interval opp[o in Opportunities] optional in o.smin..o.emax size o.dur;

maximize sum(t in Tasks) presenceOf(task[t]);
subject to {
  forall(t in Tasks)
    opportunitySelection: alternative(task[t], all(o in Opportunities: o.task==t) opp[o]);
  forall(s in Stations)
    numberOfAntennas: sum(o in Opportunities: o.station==s.id) pulse(opp[o],1) <= s.cap;
}
```
Conflict Refiner example: running model 3

- Solution with 825 tasks executed for a total of 838 candidates (98.4%)
Warmstart

- **Objective:** Start search from a known (possibly incomplete) solution given by the user in order to further improve it or to help to guide the engine towards a first solution.

- **Use cases:**
  1) Restart an interrupted search with the current incumbent
  2) Start from an initial solution found by an available heuristic
  3) Goal programming for multi-objective problems
  4) When finding an initial solution is hard, solve an initial problem that maximizes constraint satisfaction and start from its solution
  5) Successively solving similar problems (e.g. dynamic scheduling)
  6) Hierarchical problem solving (e.g. planning → scheduling)
Warmstart

- **Objective:** Start search from a known (possibly incomplete) solution given by the user in order to further improve it or to help to guide the engine towards a first solution.

- **Use cases:**
  1. Restart an interrupted search with the current incumbent
  2. Start from an initial solution found by an available heuristic
  3. Goal programming for multi-objective problems
  4. When finding an initial solution is hard, solve an initial problem that maximizes constraint satisfaction and start from its solution
  5. Successively solving similar problems (e.g. dynamic scheduling)
  6. Hierarchical problem solving (e.g. planning → scheduling)
Warmstart example: satellite scheduling problem

- USAF Satellite Control Network scheduling problem [1]
- $n$ communication requests for Earth orbiting satellites must be scheduled on a total of 32 antennas spread across 13 ground-based tracking stations
- In the instances, $n$ ranges from to 400 to 1300
- Tasks have priorities: first maximize the number of scheduled high priority tasks, then the number of scheduled low priority tasks

Warmstart example: satellite scheduling problem

```plaintext
// STEP 1: MAXIMIZE NUMBER OF SCHEDULED HIGH-PRIORITY TASKS
var op1 = new IloOplModel(def, cp);
// Maximize number of high priority tasks:
data.BestHighPriorities = -1;
op1.addDataSource(data);
op1.generate();
cp.solve();

// STEP 2: MAXIMIZE NUMBER OF SCHEDULED LOW-PRIORITY TASKS
var cp2 = new IloCP();
var op2 = new IloOplModel(def, cp2);
// Maximize number of low priority tasks:
data.BestHighPriorities = op1.nbHighPriorities;
op2.addDataSource(data);
op2.generate();

// SETTING STARTING POINT
var sp = new IloOplCPSolution();
sp.setPresence(op2.opp, op1.opp);
sp.setStart(op2.opp, op1.opp);
cp2.setStartingPoint(sp);
cp2.solve();
```
Warmstart example: satellite scheduling problem

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory usage</td>
<td>26580488</td>
</tr>
<tr>
<td>Choice points</td>
<td>62614</td>
</tr>
<tr>
<td>Number of solutions</td>
<td>29</td>
</tr>
<tr>
<td>Number of branches</td>
<td>77707</td>
</tr>
<tr>
<td>Number of fails</td>
<td>15212</td>
</tr>
<tr>
<td>Number of intervals</td>
<td>2980</td>
</tr>
<tr>
<td>Objective</td>
<td>283</td>
</tr>
</tbody>
</table>

![Graph showing objective, solution, and proven solution over time](image-url)
Warmstart example: satellite scheduling problem

Steps:

- Memory usage: 26580488
- Choice points: 62614
- Number of solutions: 29
- Number of branches: 77707
- Number of fails: 15212
- Number of intervals: 2980
- Objective: 283
Overview

- A “model-and-run” optimizer must provide:
  CP Optimizer is a “model-and-run” optimizer providing:

  - Efficient & robust automatic search
    Restart, Learning, Portfolios, Relaxations, LNS, Presolve, ...

  - Easy-to-use functionalities to control the search
    Parameters, Search phases, Warmstart, ...

  - Productivity tools for model development
    Development Studio, Search log, Conflict Refiner, ...

Questions?