Constraint Programming for Sequencing Problems

Willem-Jan van Hoeve

Tepper School of Business
Carnegie Mellon University

Joint work with Gilles Pesant, Louis-Martin Rousseau, and Ashish Sabharwal

Université Nice Sophia Antipolis
June 29, 2009

Outline

• Constraint programming
• The sequence constraint
• Experimental results
Constraint Programming

Constraint Programming Overview

Constraint Programming is a way of modeling and solving combinatorial optimization problems
- CP combines techniques from artificial intelligence, logic programming, and operations research
- There exist several industrial solvers (e.g., ILOG, Eclipse, Xpress-Kalis, Comet), and academic solvers (e.g., Gecode, Choco, Minion)
- Many industrial applications, e.g., Dutch Railways uses CP inside their Edelman-award-winning approach for timetabling
Example:

variables/domains $x_1 \in \{1,2\}$, $x_2 \in \{0,1,2,3\}$, $x_3 \in \{2,3\}$

constraints

$x_1 > x_2$

$x_1 + x_2 = x_3$

alldifferent$(x_1, x_2, x_3)$
Example:
variables/domains
- \( x_1 \in \{1\}, \ x_2 \in \{0,1\}, \ x_3 \in \{2,3\} \)

constraints
- \( x_1 > x_2 \)
- \( x_1 + x_2 = x_3 \)
- \( \text{alldifferent}(x_1,x_2,x_3) \)

Example:
variables/domains
- \( x_1 \in \{2\}, \ x_2 \in \{0,1\}, \ x_3 \in \{2,3\} \)

constraints
- \( x_1 > x_2 \)
- \( x_1 + x_2 = x_3 \)
- \( \text{alldifferent}(x_1,x_2,x_3) \)
**Goal:** Remove as many inconsistent values as efficiently possible for each constraint individually

**More filtering:** Group constraints together?

**Problem:** Solving arbitrary conjunction of constraints is NP-hard

**Solution:**
- group constraints together that occur frequently in applications, and capture tractable structure
- result is called a global constraint

(Alternative: keep NP-hard subproblem, but don’t require to filter all inconsistent values)
Filtering algorithm for sequence


Motivation: Nurse rostering

• find feasible working pattern for each employee
• restrictions:
  – every calendar-week 4 or 5 working days
  – every 9 consecutive days at most 7 working days
  – every 30 consecutive days at least 20 working days

...
**Sequence constraint**

**Example:** every 9 consecutive days at most 7 working days

variable $x_i \in \{0,1\}$ for each day $i$

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>$x_6$</td>
<td>$x_7$</td>
<td>$x_8$</td>
<td>$x_9$</td>
<td>$x_{10}$</td>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
</tr>
</tbody>
</table>

$$\{0 \leq x_1 + x_2 + x_3 \leq 1, \ldots, 0 \leq x_{11} + x_{12} \leq 7\}$$

$: sequence(x_1, x_2, \ldots, x_{12}, q=9, \min=0, \max=7)$

sequence$(x_1, x_2, \ldots, x_n, q, \min, \max)$:
the sum of every $q$ consecutive variables is between $\min$ and $\max$

A sequence constraint groups together the individual constraints

**Sequence constraint**

sequence is more powerful than individual constraints filtered separately

**Example:** $sequence(x_1, x_2, \ldots, x_7, q=5, \min=2, \max=3)$

$x_1=1, x_2=1, x_6=0$

$$\begin{array}{ccccccc}
  x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\
  1 & 1 & 0/1 & 0/1 & 0/1 & 0 & 0/1 \\
\end{array}$$

$2 \leq x_1 + x_2 + x_3 + x_4 + x_5 \leq 3$
$2 \leq x_2 + x_3 + x_4 + x_5 + x_6 \leq 3$
$2 \leq x_3 + x_4 + x_5 + x_6 + x_7 \leq 3$
Sequence constraint

History:
1988: car sequencing (Dincbas, Simonis & Van Hentenryck, 1988)
1994: sequence introduced (Beldiceanu & Contejean, 1994) as conjunction of overlapping cardinality constraints
1997: filtering algorithm (Régin & Puget, 1997) tailored to car sequencing, no complete filtering
2001: filtering algorithm (Beldiceanu & Carlsson, 2001) instance of generic class of cardinality-path constraints, no complete filtering

Goal: efficient (polynomial-time) complete filtering for sequence

First attempt: DP

Example: sequence \(x_1, x_2, x_3, x_4, x_5, x_6\), \(q=3\), \(min=1\), \(max=2\)

\[1 \leq x_1 + x_2 + x_3 \leq 2\]
\[1 \leq x_2 + x_3 + x_4 \leq 2\]
\[1 \leq x_3 + x_4 + x_5 \leq 2\]
\[1 \leq x_4 + x_5 + x_6 \leq 2\]

\[x = 1 \ 0 \ 1 \ 0 \ 0 \ 1\]

Problem: in the worst case we have to evaluate exponentially many paths
Accumulative solutions

Accumulate variables: $y[i] = x_1 + x_2 + ... + x_i$

Example: sequence $(x_1, x_2, x_3, x_4, x_5, x_6)$, $q=3$, $min=1$, $max=2$

Observation: for any two accumulate solutions, their pointwise minimum and maximum are also solutions

$x_{blue} = 1 0 1 0 0 1$
$x_{red} = 0 1 0 1 1 0$

This is not true for binary $x$ representation!
Accumulative solutions

Accumulate variables: \( y[i] = x_1 + x_2 + ... + x_i \)

Example: sequence \((x_1,x_2,x_3,x_4,x_5, x_6, q=3, \text{ min}=1, \text{ max}=2)\)

Corollary: absolute minimum and maximum solutions envelope all solutions

Find minimum solution

Algorithm:
- initialize \( y \)
- while some subsequence violated
  - push-up endpoint minimally
  - repair on left and right (using push-ups)

Invariant: \( y[i+1] - y[i] \) is 0 or 1

Example:
- sequence \((x_1,x_2,...,x_6, q=3, \text{ min}=2, \text{ max}=2)\)
- \( D(x_i) = \{0,1\} \) for all \( i \neq 5 \)
- \( D(x_5) = \{1\} \)

\[ 2 \leq y[3] - y[0] \leq 2 \]
**Find minimum solution**

**Algorithm:**
- initialize $y$
- while some subsequence violated
  - push-up endpoint minimally
  - repair on left and right (using push-ups)

**Invariant:** $y[i+1] - y[i] \text{ is 0 or 1}$

**Example:**
sequence($x_1, x_2, ..., x_6$, $q=3$, min=2, max=2)$
$D(x_i) = \{0, 1\}$ for all $i \neq 5$
$D(x_5) = \{1\}$
Find minimum solution

Properties:
- repair keeps $y[i] \leq y_{\min}[i]$ for all $i$ (by induction)
  hence, if minimum solution exists, algorithm finds it
  otherwise, $y[i] > i$ leads to unsatisfiability
- total number of push-ups bounded by $n^2$
  algorithm runs in $O(n^2)$ time

Filtering algorithm

Basic algorithm:
- for every domain value:
  - compute minimum solution (using this domain value)
    if no solution, remove domain value

Complete filtering: we remove all inconsistent values

Time complexity: $O(n^3)$

Improvements:
- maintain supports for domain values
  - each solution provides support for $n$ values
- for each value, restart from $y_{\min}$
- also compute maximum solution $y_{\max}$
  - detect violation if $y[i] > y_{\max}[i]$
- maintain $y_{\min}$ and $y_{\max}$ during search (both are monotone)
  amortize complexity: $O(n^2)$ on any path from root to a leaf
Additional remarks

• filtering algorithm also applies to generalized sequence:
  length, min, max, varies for each subsequence

  Example: nurse rostering problem
  – every calendar-week 4 or 5 working days
  – every 9 consecutive days at most 7 working days
  – every 30 consecutive days at least 20 working days

• ‘sequence’ on non-consecutive sets is NP-hard [Régin, 2005]

Single sequence constraint

<table>
<thead>
<tr>
<th>q</th>
<th>(max – min)</th>
<th>ILOG Basic</th>
<th>ILOG Extended</th>
<th>our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>limit</td>
<td>limit 34K 18</td>
<td>0 0.01</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>362K 54</td>
<td>19K 6</td>
<td>0 0.01</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>381K 55</td>
<td>113K 48</td>
<td>0 0.01</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>265K 54</td>
<td>7K 4</td>
<td>0 0.02</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>287K 48</td>
<td>0 0.5</td>
<td>0 0.02</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>limit</td>
<td>limit 61K 42</td>
<td>0 0.01</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>195K 43</td>
<td>0 0.7</td>
<td>0 0.02</td>
</tr>
</tbody>
</table>

ILOG = ILOG/IBM Constraint Programming Library

times in seconds (time limit is 1 minute)

10 instances per class
Single sequence constraint

<table>
<thead>
<tr>
<th>max – min = 1</th>
<th>ILOG Basic</th>
<th>ILOG Extended</th>
<th>our algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>n</td>
<td>back-tracks</td>
<td>CPU</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>459K</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>192K</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>48K</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>1000</td>
<td>1K</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>210K</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>221K</td>
<td>18</td>
</tr>
<tr>
<td>7</td>
<td>500</td>
<td>80K</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>1000</td>
<td>30K</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>50</td>
<td>18K</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>3K</td>
<td>0.3</td>
</tr>
<tr>
<td>9</td>
<td>500</td>
<td>49K</td>
<td>18</td>
</tr>
<tr>
<td>9</td>
<td>1000</td>
<td>17K</td>
<td>20</td>
</tr>
</tbody>
</table>

Generalized sequence constraint

Instances:
- inspired by nurse rostering problems
- two sequence constraints
- find all solutions

<table>
<thead>
<tr>
<th>instance type</th>
<th>horizon</th>
<th>#solutions</th>
<th>our sequence constraints individually backtracks</th>
<th>time</th>
<th>our generalized-sequence constraint backtracks</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>max6/8-min22/30</td>
<td>40</td>
<td>2248</td>
<td>185K</td>
<td>4 min</td>
<td>0</td>
<td>0.77 s</td>
</tr>
<tr>
<td>80</td>
<td>730</td>
<td>198k</td>
<td>18 min</td>
<td>0</td>
<td>0.61 s</td>
<td></td>
</tr>
<tr>
<td>max6/9-min20/30</td>
<td>40</td>
<td>3</td>
<td>394k</td>
<td>7 min</td>
<td>0</td>
<td>0.01 s</td>
</tr>
<tr>
<td>80</td>
<td>3</td>
<td>394k</td>
<td>30 min</td>
<td>0</td>
<td>0.05 s</td>
<td></td>
</tr>
<tr>
<td>max7/9-min22/30</td>
<td>40</td>
<td>138k</td>
<td>328k</td>
<td>7 min</td>
<td>0</td>
<td>34 s</td>
</tr>
<tr>
<td>80</td>
<td>23k</td>
<td>1847k</td>
<td>2 hours</td>
<td>0</td>
<td>15 s</td>
<td></td>
</tr>
</tbody>
</table>
Summary

- Global constraints are driving force behind successful application of constraint programming
- This talk: first polynomial-time complete domain filtering algorithm for the sequence constraint