Chapter 13: Visualization Techniques

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

ECLiPSe ELearning

Licence

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.
# Introduction

1. What we want to introduce

- Why visualize?
- How to visualize constraint programs
- Visualization Interface
- Visualization Tool
Background

- Gift grant from Cisco Systems/Silicon Valley Community Foundation
- Cisco owns open-sourced ECLiPSe system
- How to expand user-base?
- Self-taught course in constraint programming
- Intended for Cisco engineers/programmers
- Open source/available to community
- Website
  
  http://4c.ucc.ie/~hsimonis/ELearning/index.htm

Format

- Video lectures
- Slides
- Handout
- Exercises
Problems Handled in Course

- Must have puzzles!
- Send+More=Money
- Sudoku
- N-queens
- Shikaku

Practical Example Problems

- Test plan generation (BIBD)
- Progressive party problem
- Routing and wavelength assignment
- Optical network design
- Car sequencing
- Costas arrays
- Sports scheduling
- Still to come
  - Production scheduling
  - Nurse rostering
  - Airport stand allocation
Intention

- Realistic, life like problems
- Must address scalability issues
- Often, problem not completely specified
- Issue: Hard to verify by hand
- Complexity still limited, not real problems
- No attempt at integration

How do we understand behavior?

- Mental model
- Formal analysis
- Debugging
- Tracing
- Life visualization
- Post-mortem analysis
Why Visualize?

- Understand what is done
- Understand what is done in which order
- Understand what is *not* done
- Understand when to give up

Design Choices

- No deep integration with solver
- Post-mortem visualization
- Intermediate file format
- No view of detailed propagation
  - Tool not intended for debugging constraint engine
**Conceptual Model**

- Stable state at defined program points
- Granularity
  - Assign value
  - Post constraint
- Show stable state after propagation
- Do not show individual propagation steps

**Visualizers**

- Search tree
- Variables
- Constraints
Visualization Tool

- Developed in Java
- Show two panes: tree and state
- Navigate along timeline
How many visualizers do we need?

- Develop few primitives
  - Cell based view
  - Domain vector
- Allow aggregation
  - Vector/matrix
  - General layout
- Which global constraints require more?
  - Task based view for \textit{cumulative}
  - Matching/flow based representation does not scale

How to Interpret Visualization

- Search tree
  - Good/bad choices
  - Place of backtracking
- State
  - Missing propagation
Costas Array Search tree (Size 16)

- Deep backtracking
- Third choice wrong
- Last choice wrong
- Value selection strategy useless

Missing Propagation

The model is doing this
Missing Propagation

It could be doing that!

Comparison (Search Tree, size 16)

Initial Model

Improved Model
Progressive Party Problem, 9 Time Periods

2 Restarts Before Solution Found
Clear: impossible to explore search space

- Either many solutions or good value selection
- Value selection at end rather poor
- Probably many solutions
### Introduction
Visualization by Annotation
Visualization Interface
Conclusions

#### Missing Propagation: Shikaku

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   | 1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10| 11| 12| 13| 14| 15| 16| 17| 18| 19| 20| 21| 22| 23| 24| 25| 26| 27| 28| 29| 30| 31| 32| 33| 34| 35| 36| 37| 38| 39| 40| 41| 42| 43| 44| 45| 46| 47| 48| 49| 50| 51| 52| 53| 54| 55| 56| 57| 58| 59| 60| 61| 62| 63| 64| 65| 66| 67| 68| 69| 70| 71| 72| 73| 74| 75| 76| 77| 78| 79| 80| 81| 82| 83| 84| 85| 86| 87| 88| 89| 90| 91| 92| 93| 94| 95| 96| 97| 98| 99| 100|

#### Sendmore Program Annotated

```prolog
sendmory(L, Output, IgnoreFixed):-
  L=[S,E,N,D,M,O,R,Y],
  L :: 0..9,
  create_visualization([output:Output, 
                        ignore_fixed:IgnoreFixed, 
                        width:8, 
                        height:10],Handle),
  add_visualizer(Handle, 
                vector(L), 
                [display:expanded]),
  alldifferent(L),draw_visualization(Handle),
  S #\= 0,draw_visualization(Handle),
  M #\= 0,draw_visualization(Handle),
```

Helmut Simonis | Visualization Techniques
Sendmore Program Annotated

\[
1000 \times S + 100 \times E + 10 \times N + D + \\
1000 \times M + 100 \times O + 10 \times R + E \neq \\
10000 \times M + 1000 \times O + 100 \times N + 10 \times E + Y,
\]

\text{name\_variables(Handle,L,} \\
['S','E','N','D','M','O','R','Y'], \\
\text{Pairs),} \\
\text{root(Handle),} \\
\text{search(Pairs,1,input\_order,} \\
\text{tree\_indomain(Handle,_),} \\
\text{complete,[]}}, \\
\text{solution(Handle),} \\
\text{close\_visualization(Handle).}

Sudoku Program Annotated

\text{model(Matrix,Method,Output):-} \\
\text{Matrix[1..9,1..9] :: 1..9,} \\
\text{create\_visualization([output:Output,} \\
\text{width:9,} \\
\text{height:9],Handle),} \\
\text{add\_visualizer(Handle,} \\
\text{domain\_matrix(Matrix),} \\
\text{[display:text])}, \\
\text{draw\_visualization(Handle),} \\
\text{(for(I,1,9),} \\
\text{param(Matrix,Method,Handle) do} \\
\text{Method:alldifferent(Matrix[I,1..9]),} \\
\text{draw\_visualization(Handle,[focus:row(I)])}, \\
\text{Method:alldifferent(Matrix[1..9,I]),} \\
\text{draw\_visualization(Handle,[focus:col(I)])}} \\
\),
(multifor([I,J],[1,1],[7,7],[3,3]),
  param(Matrix,Method,Handle) do
    draw_visualization(Handle, 
      [focus:block(I,J,3,3)])
  ),
  extract_array(Handle,row,Matrix,NamedList),
  root(Handle),
  search(NamedList,1,input_order,
    tree_indomain(Handle,_),
    complete,[]),
  solution(Handle),
  close_visualization(Handle).
After Setup (Forward Checking)

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Propagation Steps (Bounds Consistency)

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Helmut Simonis  Visualization Techniques  33
After Setup (Bounds Consistency)

Propagation Steps (Domain Consistency)
After Setup (Domain Consistency)

Forward Checking

Bounds Consistency

Domain Consistency
Instrumented indomain

```prolog
tree_indomain_generic(Term,Handle,Handle,Type):-
  Handle = visualization{ignore_fixed:IgnoreFixed,
                        var_arg:VarArg,
                        name_arg:NameArg,
                        focus_arg:FocusArg},
  arg(VarArg,Term,X),
  ((integer(X),IgnoreFixed = yes) ->
   true
  ;
   arg(NameArg,Term,Name),
   arg(FocusArg,Term,Focus),
   get_domain_as_list(X,L),
   get_domain_size(X,Size),
   reorganize_domain(X,L,Type,K),
   try_value(Handle,X,K,Name,Size,Focus)
  ).
```

```
try_value(Handle,X,[V|_],Name,Size,Focus):-
  ((X = V, true) ->
   try(Handle,Name,Size,V),
   focus_option(Focus,FocusOption),
   draw_visualization(Handle,FocusOption)
  ;
   failure(Handle,Name,Size,V),
   fail_option(Focus,V,FailOption),
   draw_visualization(Handle,FailOption),
   fail
  ).
try_value(Handle,X,[_|R],Name,Size,Focus):-
  try_value(Handle,X,R,Name,Size,Focus).
```
Architecture (Current)

Program + Annotation

ECLiPSe

TreeLog

VisualizationLog

Viz

Treemap

SVG

Graph

Statistics

Inkscape

Browser

Batch

VizTool

Annotated Image

PDF

Architecture (Planned)

Program + Annotation

ECLiPSe

TreeLog

VisualizationLog

Viz

Treemap

SVG

Graph

Statistics

Inkscape

Browser

Batch

VizTool

Annotated Image

PDF
**CP-Inside**

Program + Annotation

CP-Inside

TreeLog

VisualizationLog

Viz

Treemap

SVG

Graph

Statistics

Inkscape

Browser

Batch

VizTool

Annotated Image

PDF

**Generic Tool**

Program + Annotation

Your Favourite Tool

TreeLog

VisualizationLog

Viz

Your Favourite Analysis

Treemap

SVG

Graph

Statistics

Inkscape

Browser

Batch

VizTool

Annotated Image

PDF
TreeLog Format

- XML based description
- Record information about nodes in search tree
  - Choices
  - Failures
  - Success
- Redundant information to ease generation

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<tree version="1.0" >
  <root id="0" />
  <try id="1" parent="0" name="S" size="1" value="9" />
  <fail id="2" parent="1" name="E" size="4" value="4" />
  <try id="3" parent="1" name="E" size="4" value="5" />
  <try id="4" parent="3" name="N" size="1" value="6" />
  <try id="5" parent="4" name="D" size="1" value="7" />
  <try id="6" parent="5" name="M" size="1" value="1" />
  <try id="7" parent="6" name="O" size="1" value="0" />
  <try id="8" parent="7" name="R" size="1" value="8" />
  <try id="9" parent="8" name="Y" size="1" value="2" />
  <succ id="9" />
  <fail id="10" parent="1" name="E" size="4" value="6" />
  <fail id="11" parent="1" name="E" size="4" value="7" />
</tree>
```
**VisualizerLog Format**

- XML based description
- Describe state of variables and/or constraints at specific stages
  - Where annotated in program
  - For every node in tree
- Linked to search tree log

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<visualization version="1.0" >
<visualizer id="1" type="vector" display="expanded" x="0" y="0"
  width="8" height="10" group="1" min="0" max="9" />
<state id="1" tree_node="-1" >
<visualizer_state id="1" >
<dvar index="1" domain="0 .. 9" />  
<dvar index="2" domain="0 .. 9" />  
<dvar index="3" domain="0 .. 9" />  
<dvar index="4" domain="0 .. 9" />  
<dvar index="5" domain="0 .. 9" />  
<dvar index="6" domain="0 .. 9" />  
<dvar index="7" domain="0 .. 9" />  
<dvar index="8" domain="0 .. 9" />
</visualizer_state>
</state>
...</visualization>
```
VisualizerLog Example

...<state id="2" tree_node="-1" >
<visualizer_state id="1" >
<dvar index="1" domain="1 .. 9" />
<dvar index="2" domain="0 .. 9" />
<dvar index="3" domain="0 .. 9" />
<dvar index="4" domain="0 .. 9" />
<dvar index="5" domain="0 .. 9" />
<dvar index="6" domain="0 .. 9" />
<dvar index="7" domain="0 .. 9" />
<dvar index="8" domain="0 .. 9" />
</visualizer_state>
</state>
...

...<state id="5" tree_node="1" >
<visualizer_state id="1" >
<integer index="1" value="9" />
<dvar index="2" domain="4 .. 7" />
<dvar index="3" domain="5 .. 8" />
<dvar index="4" domain="2 .. 8" />
<integer index="5" value="1" />
<integer index="6" value="0" />
<dvar index="7" domain="2 .. 8" />
<dvar index="8" domain="2 .. 8" />
<focus group="-" index="1" />
</visualizer_state>
</state>
...
VisualizerLog Example

...<state id="6" tree_node="2">
<visualizer_state id="1">
<integer index="1" value="9" />
<dvar index="2" domain="4 .. 7" />
<dvar index="3" domain="5 .. 8" />
<dvar index="4" domain="2 .. 8" />
<integer index="5" value="1" />
<integer index="6" value="0" />
<dvar index="7" domain="2 .. 8" />
<dvar index="8" domain="2 .. 8" />
<failed group="-" index="2" value="4" />
</visualizer_state>
</state>
...

...<state id="14" tree_node="9">
<visualizer_state id="1">
<integer index="1" value="9" />
<integer index="2" value="5" />
<integer index="3" value="6" />
<integer index="4" value="7" />
<integer index="5" value="1" />
<integer index="6" value="0" />
<integer index="7" value="8" />
<integer index="8" value="2" />
</visualizer_state>
</state>
...
</visualization>
Conclusions

- New ELearning course for ECLiPSe
- Open source material, Creative Commons BY-NC-SA license
  - Application driven
  - Modelling with global constraints
  - Customizing search
- Effort only justifiable through Cisco grant

Design choice: System independent
- Provide enough information to user of system, not to tool developer
- Relatively few primitives, extensible for specific global constraints
- XML intermediate format, open for specific analysis