Visualising and profiling CP models: is the Holy Grail in sight?

Maria Garcia de la Banda - PTGH 2017
Evolution in profiling/visualisation for constraint programming

We are…

here?
A bit of history (definitely NOT exhaustive)
Early days…
Grace: 1995 for ECLiPSe – interactive and variable focused

- **Control menu**
- **Compares executions**
- **Execution/display control**

- **Var domain**
- **2D variable matrix**
- **Variables being explored**
- **Selected terms/expressions**
Explorer: 1997 for Oz – practical, interactive and search tree focused

- **User-defined display procedures**
- **Collapsed trees**
- **Basic stats**
- **Double-click to explore node**
- **Scale the tree**
- **Textual or user-defined visualisation for node info**

![Image of Oz Explorer window]

- **Explorer**
- **Move**
- **Search**
- **Nodes**
- **Hide**
- **Options**

**User-defined display procedures**

- alice
- bert
- chris
- deb
- evan

**Collapsed trees**

**Basic stats**

- Time: 12.79s (16%)  
  - 10693  
  - 6
  - 10688  
  - Depth: 20
DiSCIPI project: 1996-1999
Debugging Systems for Constraint Programming
(an explosion of tools)
For CHIP – interactive, variable and constraint focused

- Tree view (Oz inspired)
- Variable update view
- Constraint incidence matrix
- Domain state view
- Phase line display
For CHIP – dedicated support for global constraints

Different views of a diffn constraint

Cumulative constraint

Cycle constraint
APT for CLP – interactive, decoupled & execution focused

- And-Or execution tree
- Source code
- Variable update view

Parallel execution trees focusing on time or events
VIFID/TRIFID for CLP – domain and constraint focused

- Detailed domain evolution
- Constraint graph

- Domain comparison for a given constraint
- Domain size evolution
OADymPPaC project: 2000-2004
Tools for dynamic analysis and debugging of CP
(the value of generic protocols)
GenTra4CP: a generic tracer format for finite domain solvers
CLP(FD) visualiser via on-the-fly analysis of low-level trace events

Programmable views including tree (dot), 3D variable (C+VRML) and search (ILOG Esieve)
CLPGUI: generic for CLP FD via annotation predicates

- Incremental tree
- Solution view
- 3D domain
- 3D tree
- Dual tree
OPL Studio: 2001 for ILOG – interactive tree, domain & propagation focused
Similar tools being developed for SAT and Local Search (different purpose/insights)
SATGraf for SAT: visualisation of the incidence-graph structure and evolution

Two different problems

Two different time points
DPVis for SAT – visualising the constraint graph, its evolution & search tree

Constraint graph

Search trees for several problems

Stats comparison
For CBLS: constraint violations, conflicts and evolution focused

Evolution of the objective

Constraint & variable violations

Variable evolution

Evolution of knapsack
Towards lightweight, practical, generic tools
CP-VIZ: 2010 for ECLiPSe & SICStus: generic, lightweight and versatile
What had we learned?
Lessons learned: it is good to be ....

- **Generic**: not tightly coupled to any solver
- **Lightweight**: only require small changes to any solver
- **Versatile**: provide interface to other visualisation tools
- **Intuitive**: clearly visualise what you mean to
- **Built-in**: not everything must be user-defined
- **Efficient
- **Open source**

CP-VIZ is most of these, so why is not shipped with every solver?
The following is not just my work! It is mostly the work of...

Kevin Leo

Chris Mears

Guido Tack

Maxim Shishmarev

Mark Wallace
What are we missing? Programming in the large

- Since:
  - Most tools work well for small problems (many developed for education)
  - **Visual insight** is hard for Ks of variables, constraints, and Ms of search nodes

- Need:
  - Visualisations that can be meaningful with **scale**
  - **Focus** the user’s attention to on the interesting parts
  - Automatically find these interesting parts (statistical markers)
Example: pixel-trees – they scale on time and easily show solution density

Solutions (light green) found after lots of search

Other time-related info can be shown together

Visual patterns might indicate some property
Example: shape analysis to automatically focus attention

Histogram of “similar shape analysis” by size

Location of shapes within search tree
Example: statistical markers to automatically focus attention

Effective versus ineffective backjumping
What are we missing? Comparing the execution after model changes

- **Since:**
  - Most tools focus on a **single** execution
  - This does not help the **iterative** development process

- **Need:**
  - Visualisations that can meaningfully **compare** several executions
  - **Focus** the user’s attention to on the modified parts
Example: comparing two executions of the same model via tree merging

Merged trees and stats on divergence
Example: replaying a model’s search with a different solver

Learning solver: ~2K failures

Non-learning solver: ~18K failures
Comparing needs linking “same” vars and constraints across executions

Same paths for variables named differently by the compiler
Since:
- Most tools focus on **one kind of search** (mostly depth-first)
- Nowadays we have parallel search, restarts, learning solvers, local search, LNS, SAT, MIP

Need:
- Visualisations that can **support** all these searches and paradigms
- And they can help **compare** them
- **Focus** the user’s attention to on the modified parts
Example: parallel searches, restarts and learning solvers
What are we missing? User testing

■ Since:
  – Most tools are designed by solver developers
  – Application users might require different kinds of information

■ Need:
  – Understand what application users need
  – Develop possible solutions
  – Test the solutions to see if they work for the users

What are we missing? User testing
Example: user-centred design process for visual profiling tools in CP
Example: Many visualisation alternatives, many findings, but more needed
What are we missing? Connecting back to the model

- **Since:**
  - The ultimate goal is to **modify the model** to improve it
  - In CP the execution is **quite far** from the original model:
    - Different variables, constraints, expressions..

- **Need:**
  - **Connect** the findings made during the execution to the model variables and constraints
  - Again: connect variables and constraints **across executions**
Evolution in profiling/visualisation for constraint programming

We are…

here?
Thanks for listening!