MONASH University

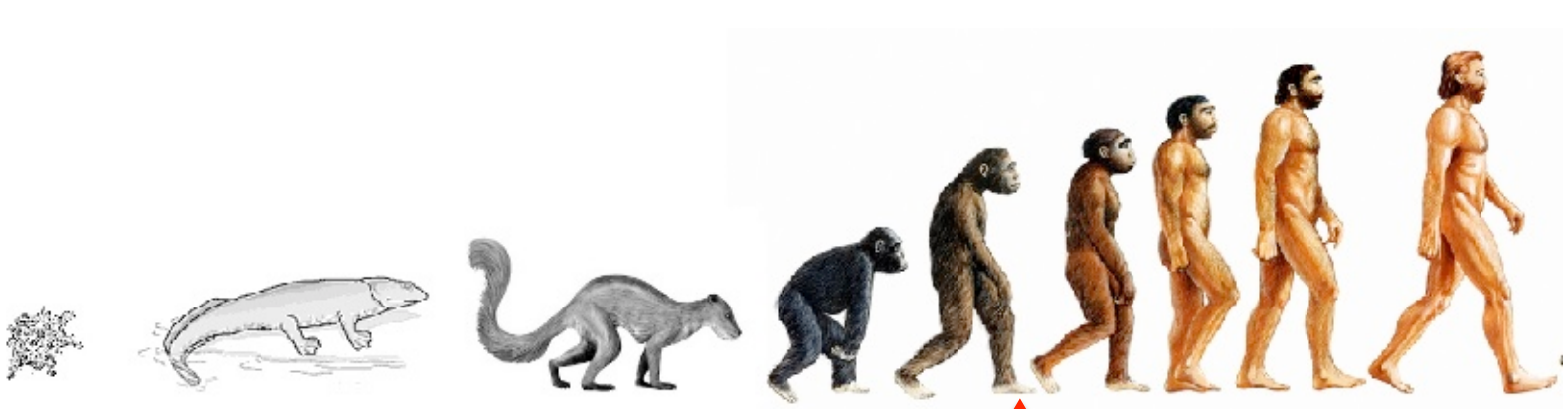
Visualising and profiling CP models: is the Holy Grail in sight?

Maria Garcia de la Banda - PTGH 2017

GROUP OF EIGHT AUSTRALIA

Evolution in profiling/visualistion for constraint programming

We are…



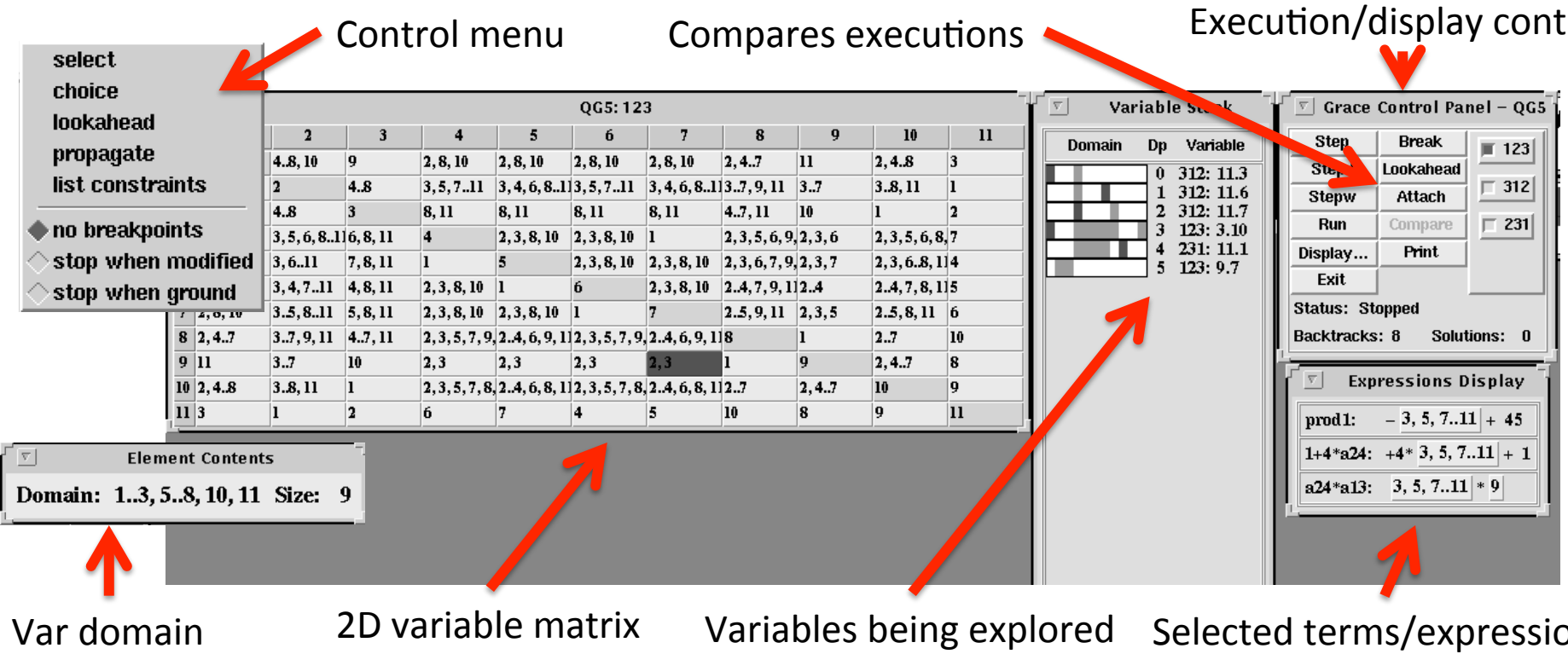
here?

MONASH University

A bit of history (definitely NOT exhaustive)

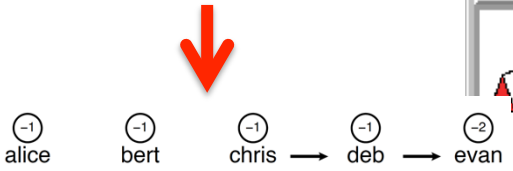
Early days…

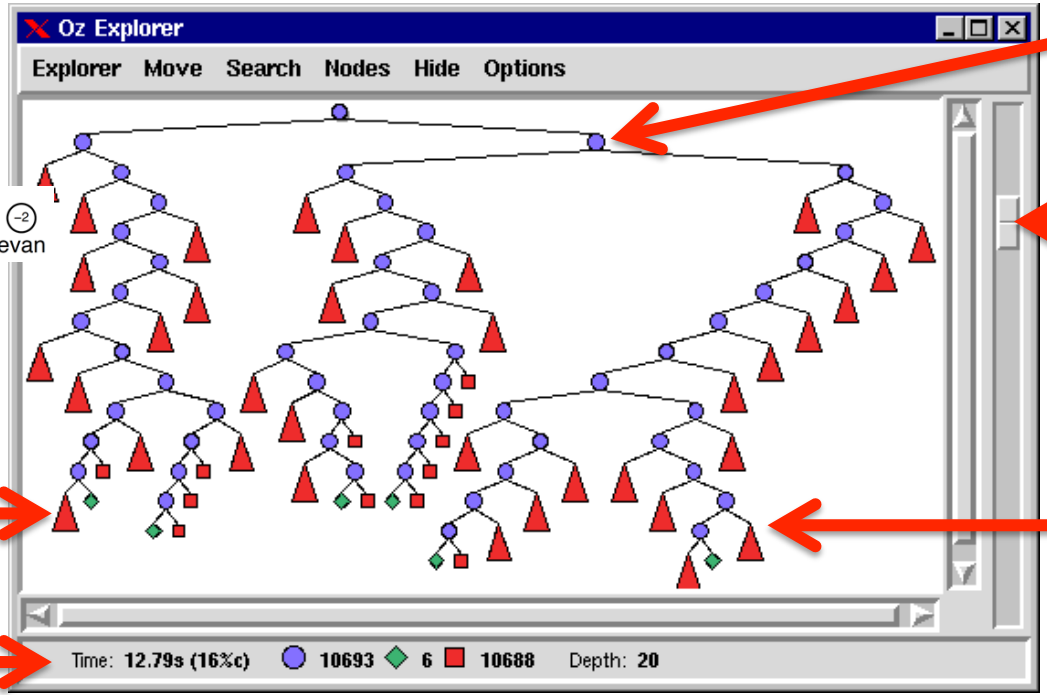
Grace: 1995 for ECLiPSe – interactive and variable focused



Explorer: 1997 for Oz – practical, interactive and search tree focused

User-defined display procedures





Double-click to explore node

Scale the tree

Collapsed trees

Textual or user-defined visualisation for node info

Basic stats

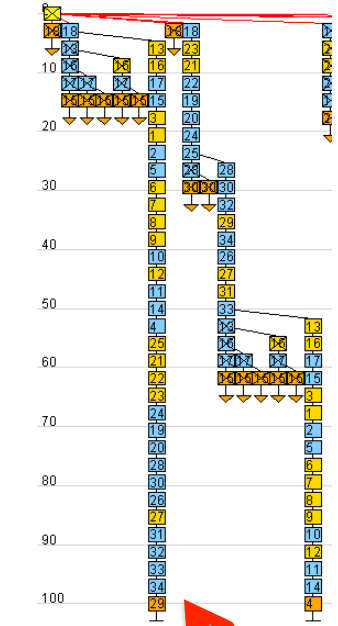
MONASH University

DiSCiPl project: 1996-1999

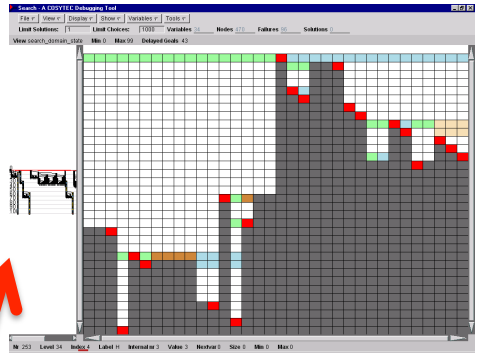
Debugging Systems for Constraint Programming

(an explosion of tools)

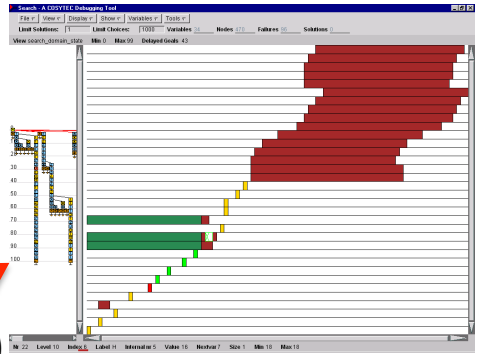
For CHIP – interactive, variable and constraint focused



Tree view (Oz inspired)

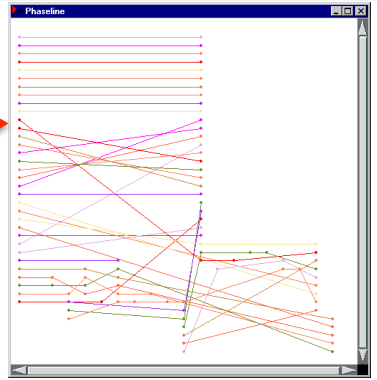


Variable update view

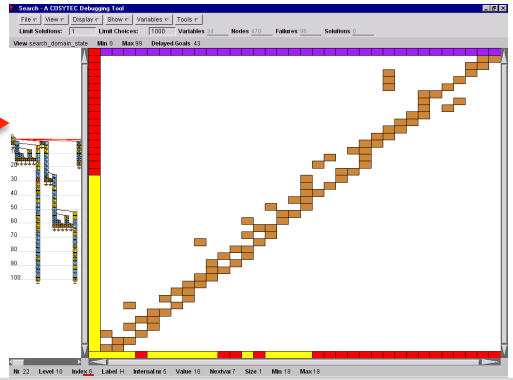


Domain state view

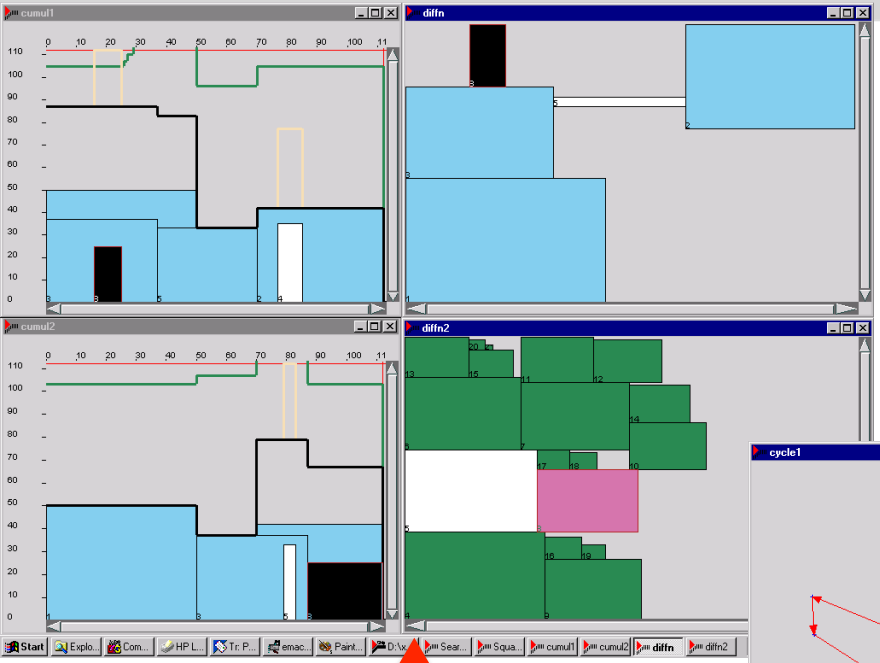
Phase line display



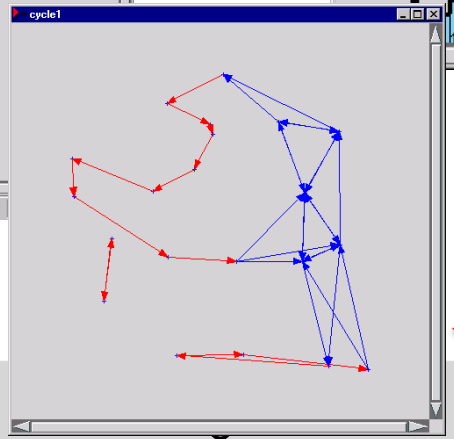
Constraint incidence matrix



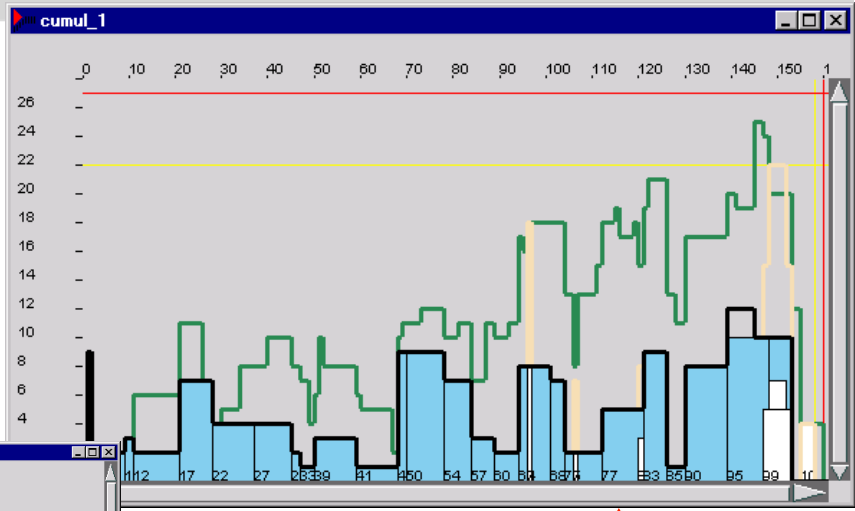
For CHIP – dedicated support for global constraints



Different views of a diffn constraint



Cycle constraint

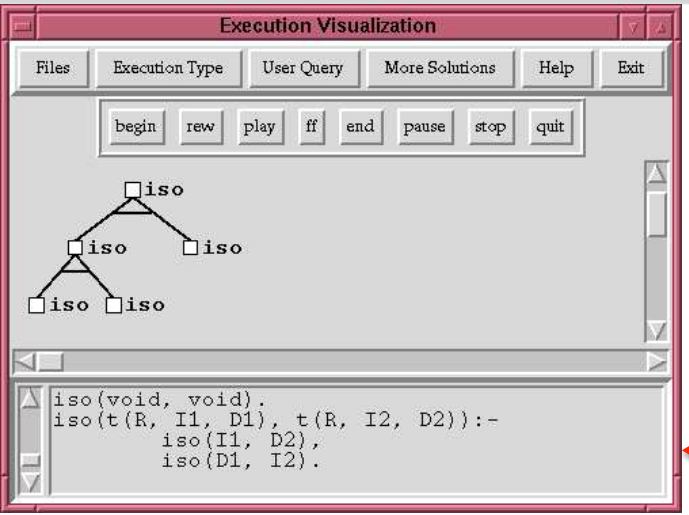


Cumulative constraint

MONASH University

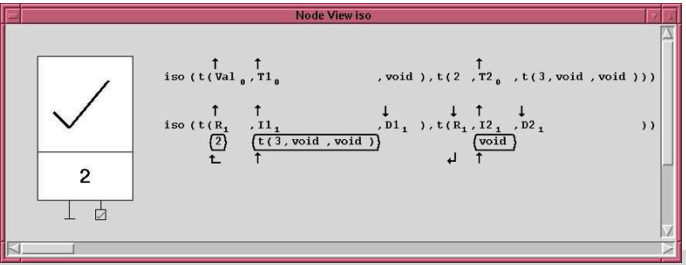
MONASH INFORMATION TECHNOLOGY

APT for CLP – interactive, decoupled & execution focused

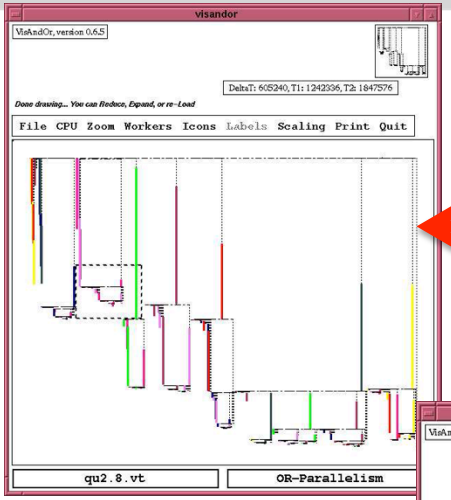


And-Or execution tree

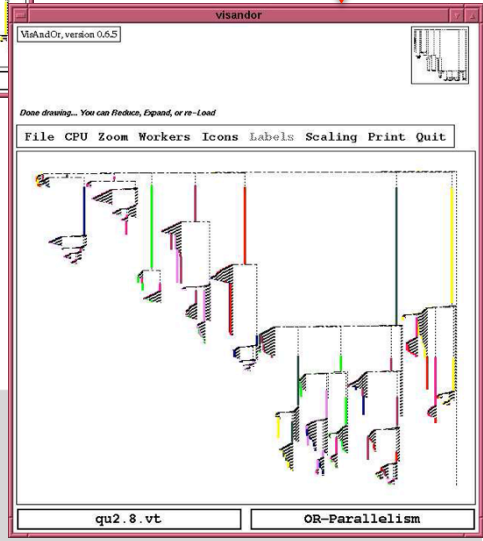
Source code



Variable update view

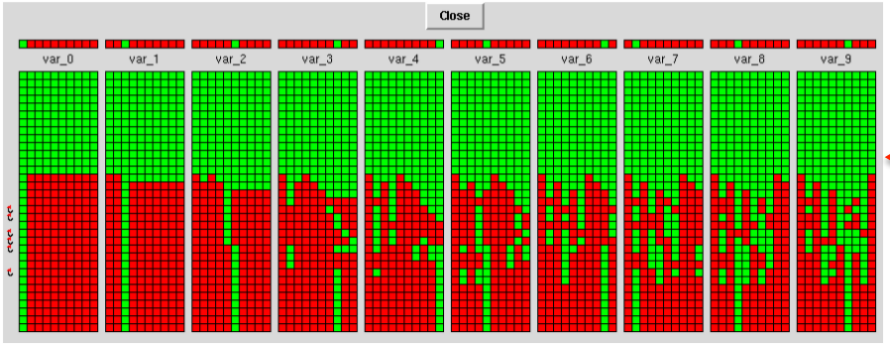


Parallel execution trees focusing on time or events

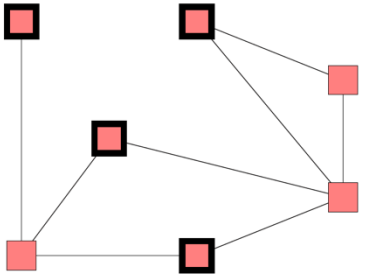


MONASH University

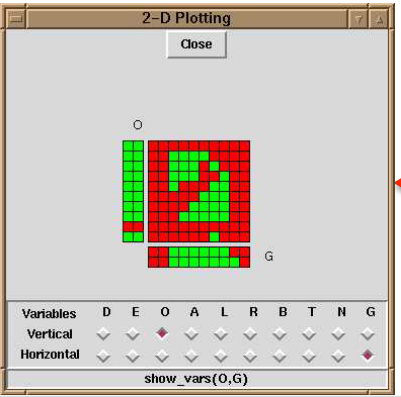
VIFID/TRIFID for CLP – domain and constraint focused



Detailed domain evolution

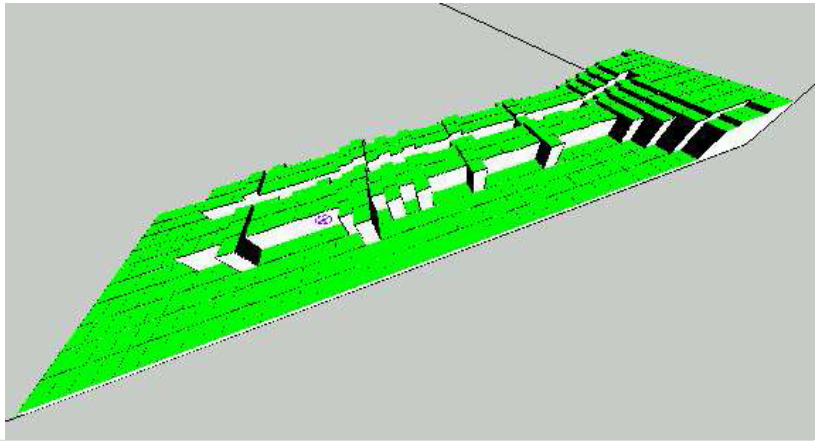


Constraint graph



Domain comparison for a given constraint

Domain size evolution



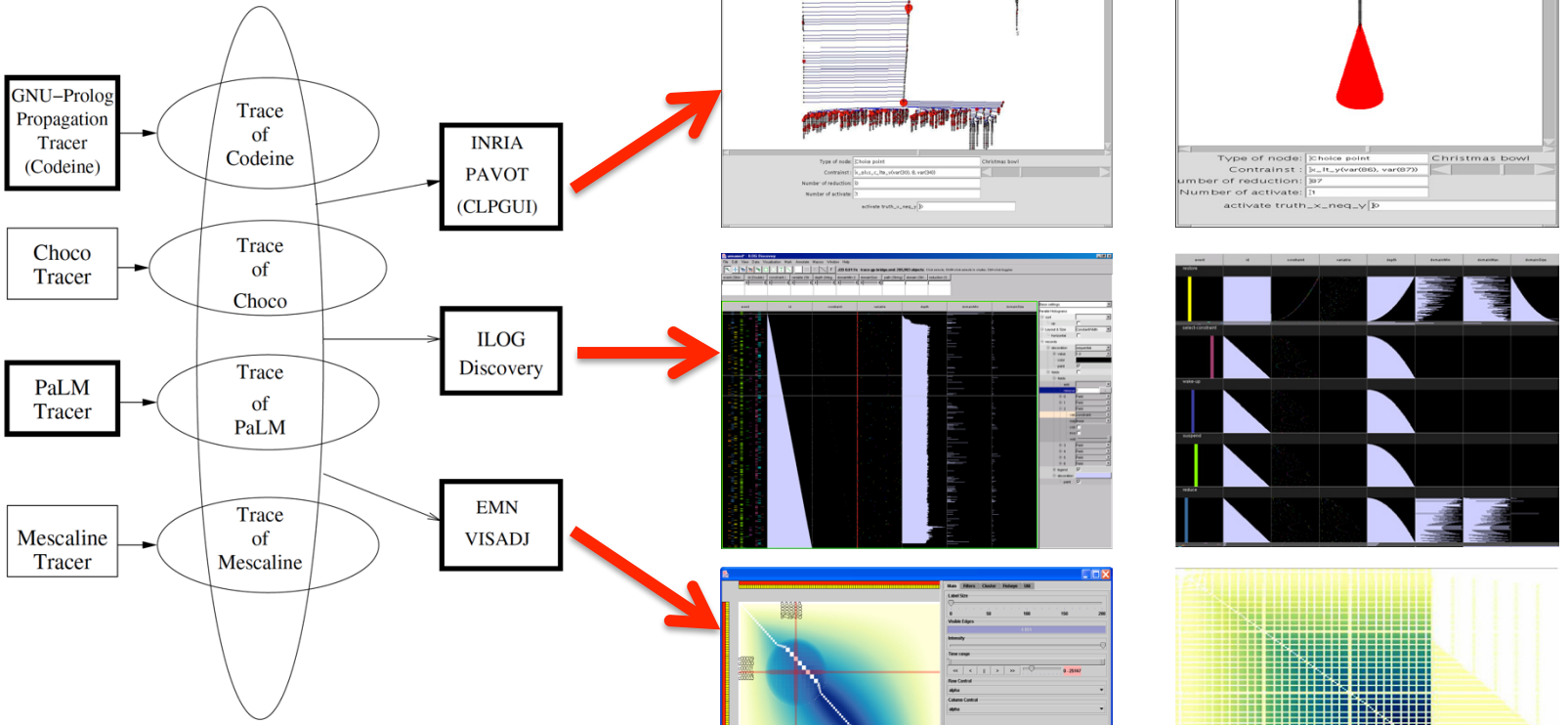
MONASH University

OADymPPaC project: 2000-2004

Tools for dynamic analysis and debugging of CP

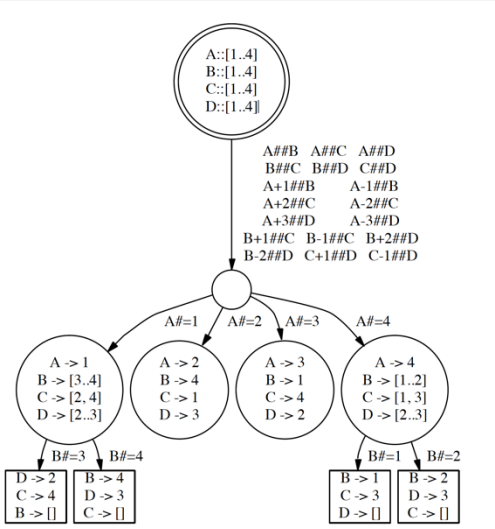
(the value of generic protocols)

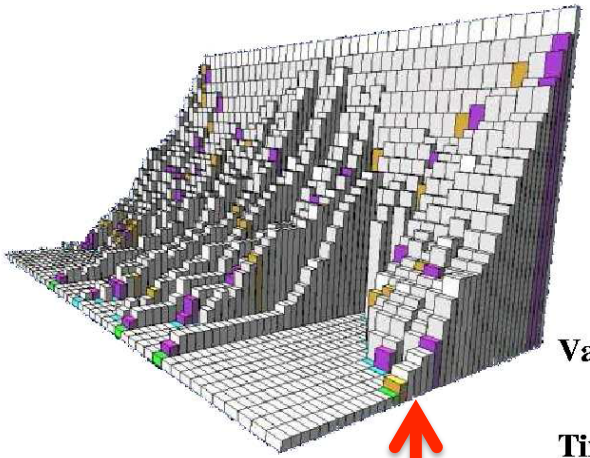
GenTra4CP: a generic tracer format for finite domain solvers

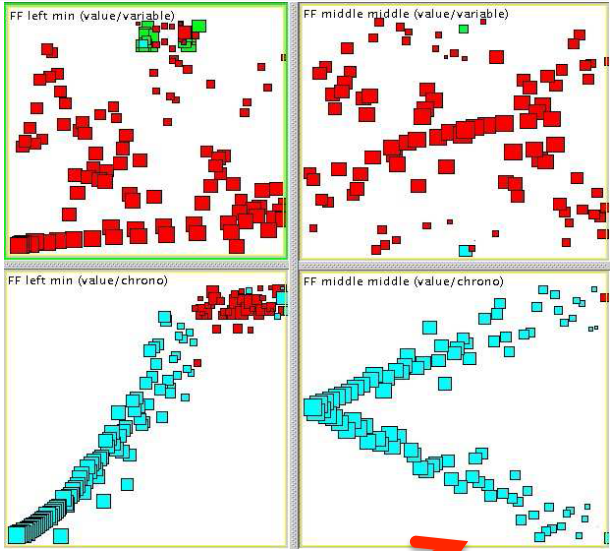


MONASH University

CLP(FD) visualiser via on-the-fly analysis of low-level trace events

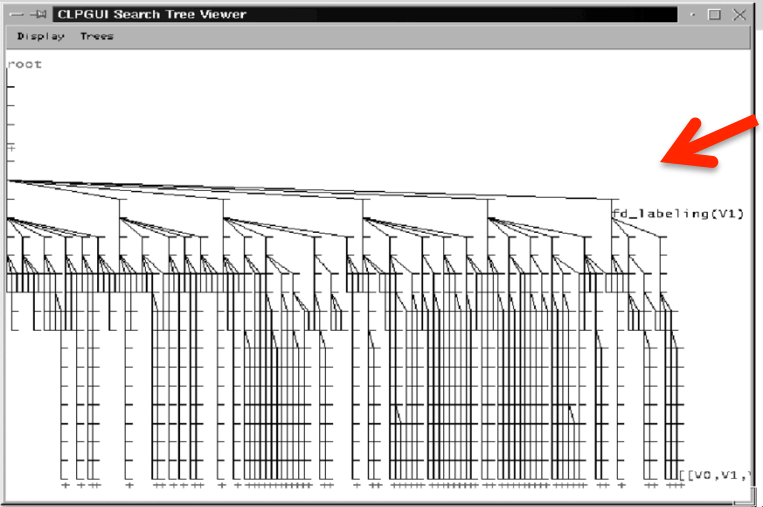






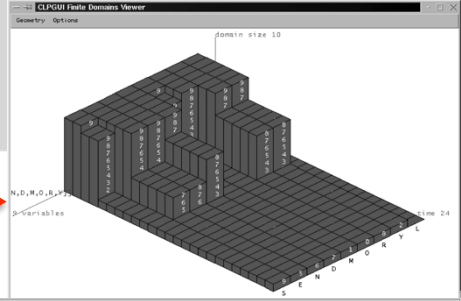
Programmable views including tree (dot) , 3D variable (C+VRML) and search (ILOG Esieve)

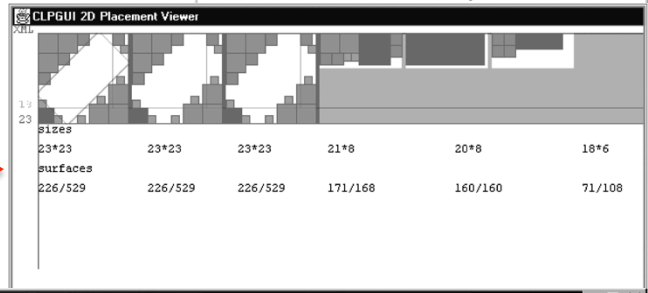
CLPGUI: generic for CLP FD via annotation predicates



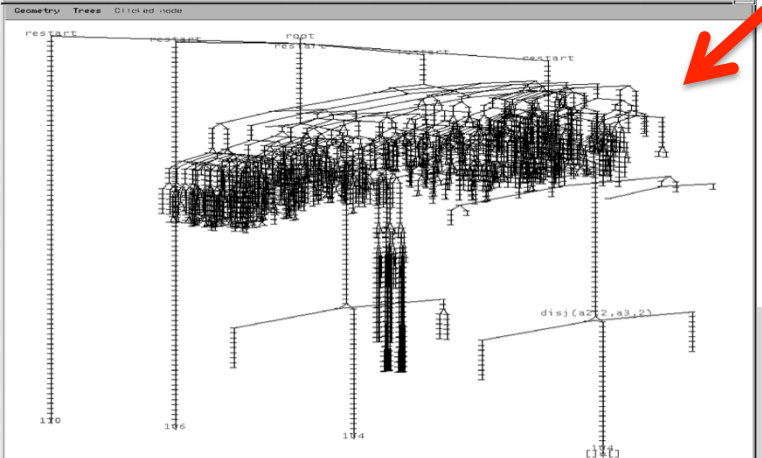
Incremental tree

3D domain



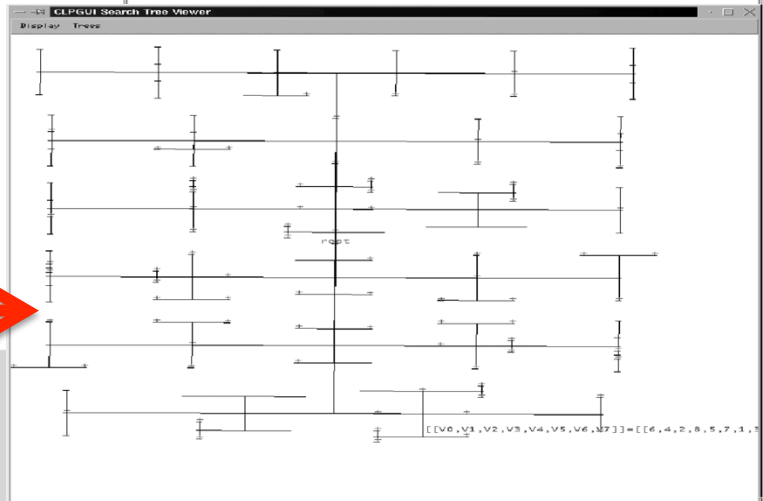


Solution view

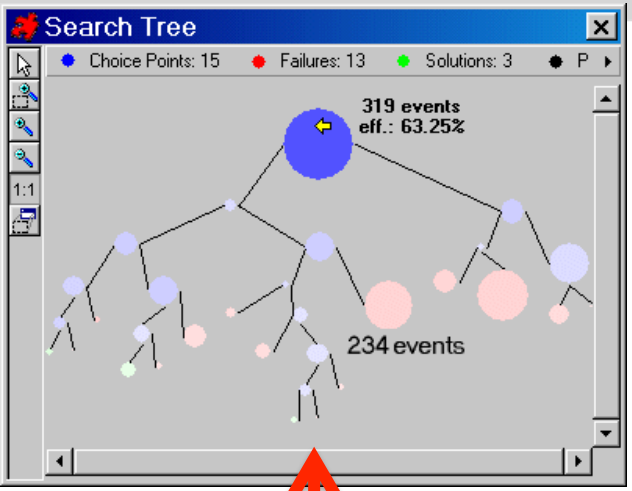


3D tree

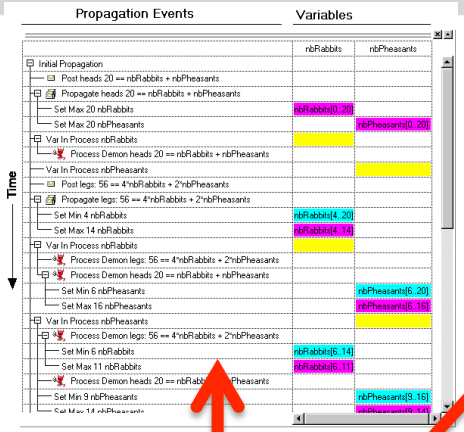
Dual tree



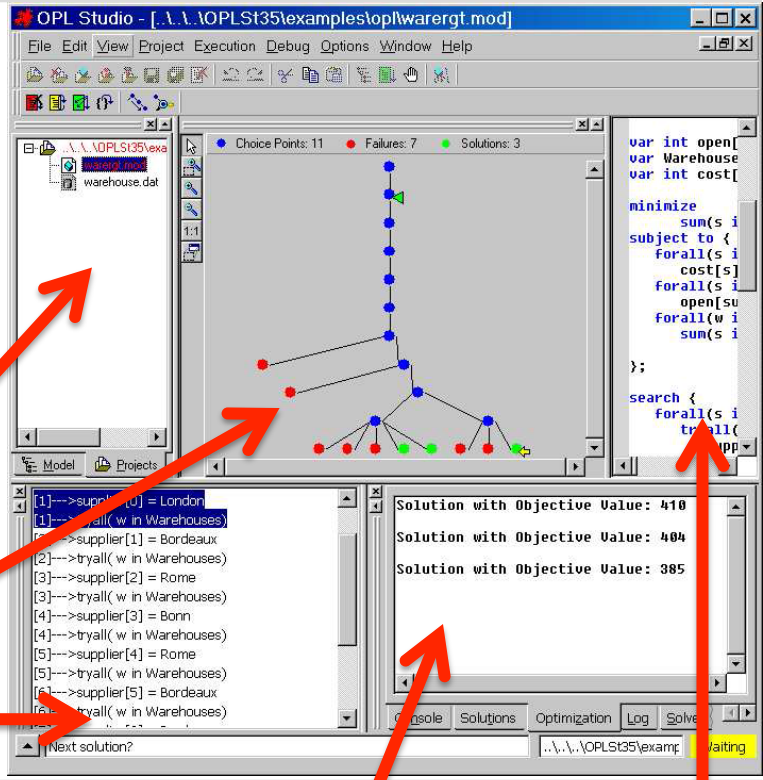
OPL Studio: 2001 for ILOG – interactive tree, domain & propagation focused



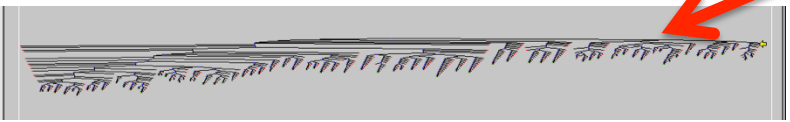
Xmas tree



Propagation spy



Tree view
(can collapse)



Choice stack

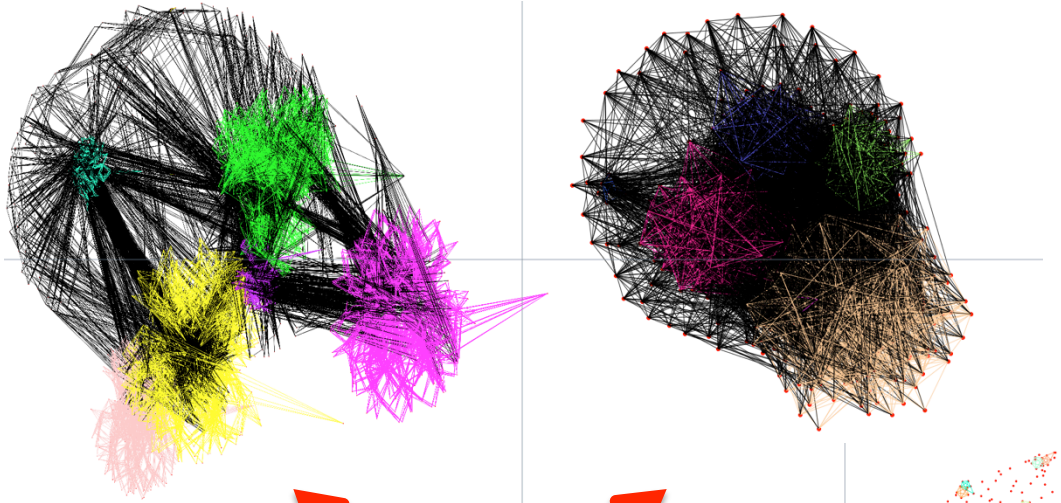
Results

Code

MONASH University

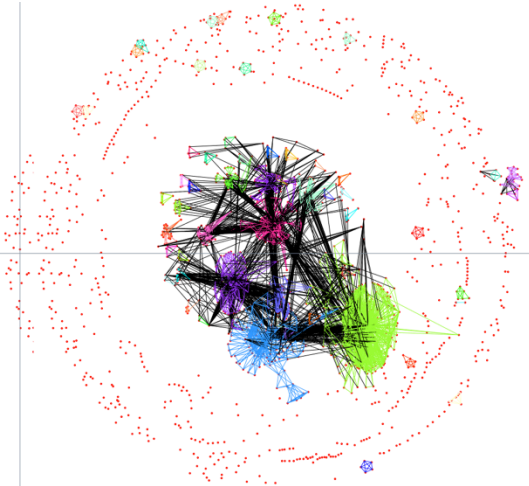
Similar tools being developed for SAT and Local Search (different purpose/insights)

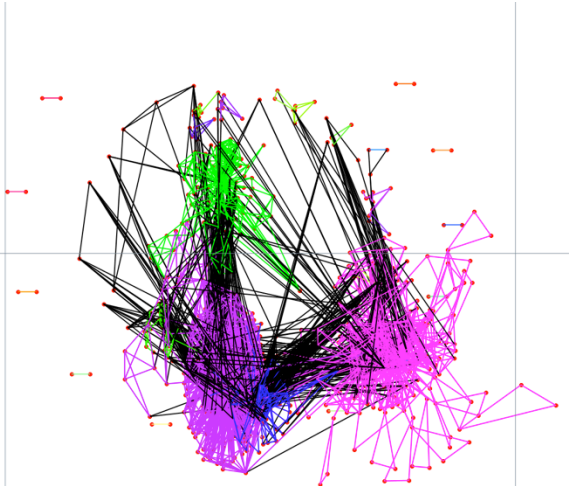
SATGraf for SAT: visualisation of the incidence-graph structure and evolution



Two different time points

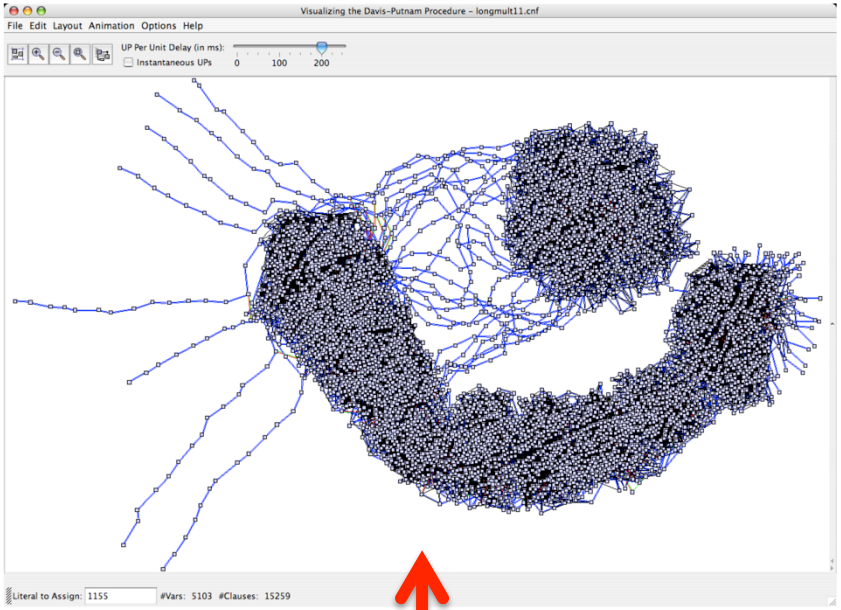
Two different problems





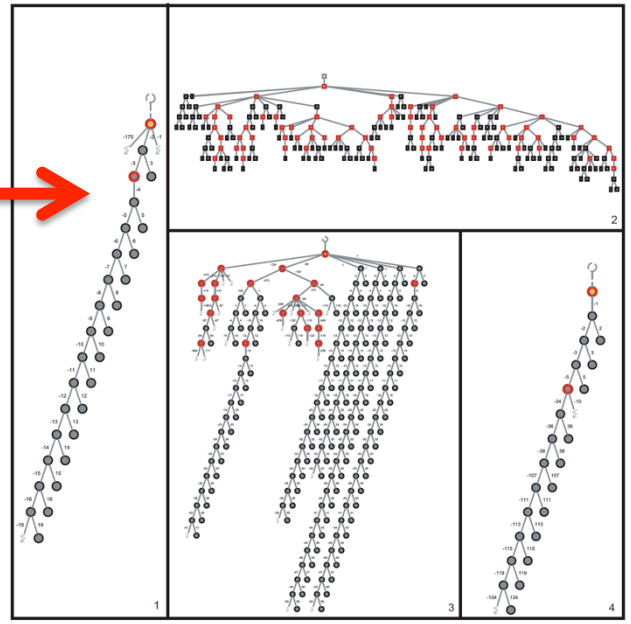
MONASH University

DPVis for SAT – visualising the constraint graph, its evolution & search tree



Constraint graph

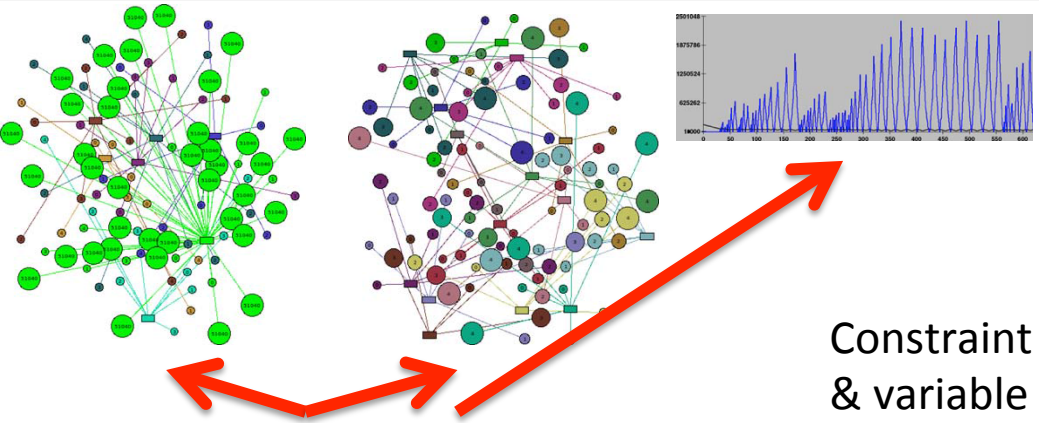
Search trees for several problems



Stats comparison

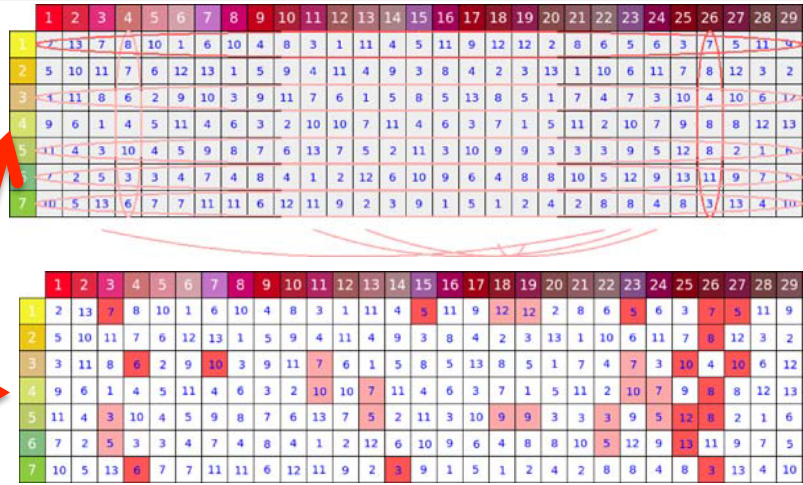
Instance	#Vars	#Clauses	#TLAs	#TLA/#Vars
longmult1	631	1611	151	23.93%
uuf50-0125	50	218	13	26.00%
ssa2670-141-d7	753	1619	336	44.62%
bw_large.b-d8	889	9949	233	26.21%

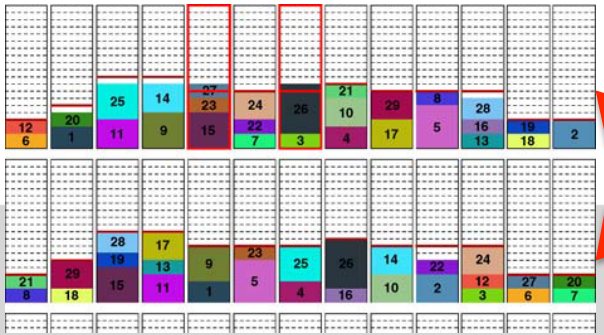
For CBLS: constraint violations, conflicts and evolution focused



Evolution of the objective

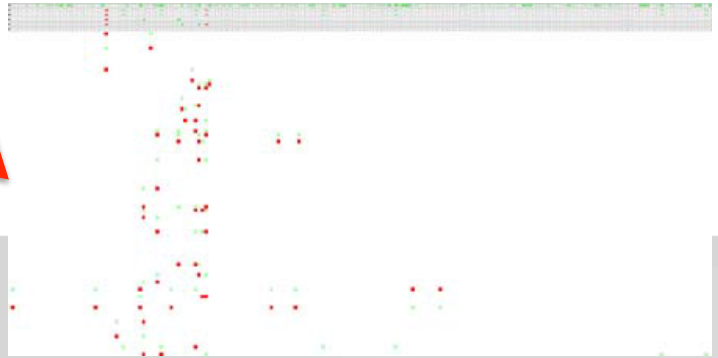
Constraint & variable violations





Evolution of knapsack

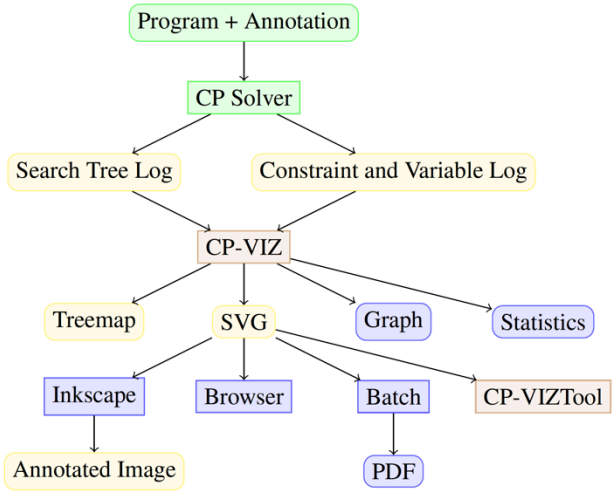
Variable evolution



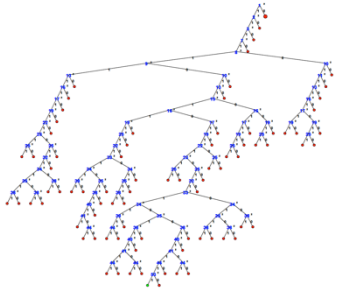
MONASH
University

Towards lightweight, practical, generic tools

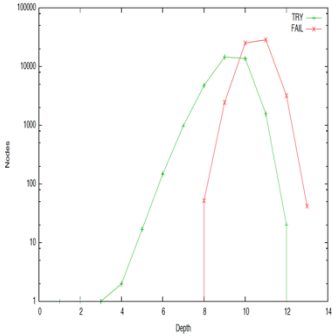
CP-VIZ: 2010 for ECliPSe & SICStus: generic, lightweight and versatile



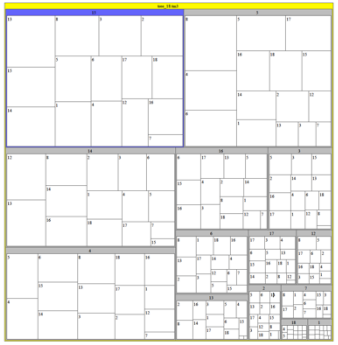
Tree Display

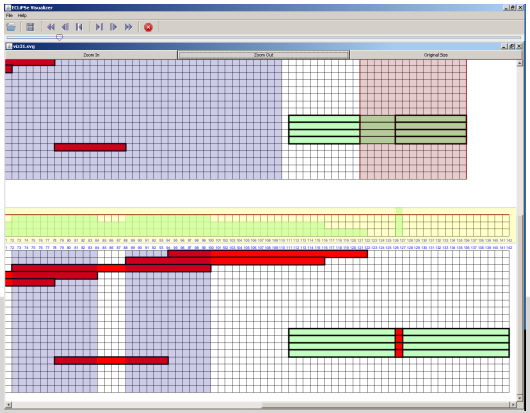


Failure Level

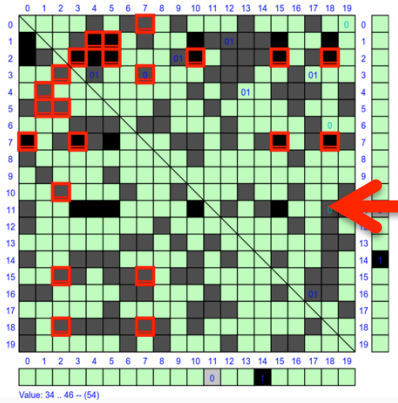


Failure Causes





Different search tree views



Customised globals

Invariant checking

MONASH
INFORMATION
TECHNOLOGY

MONASH University

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

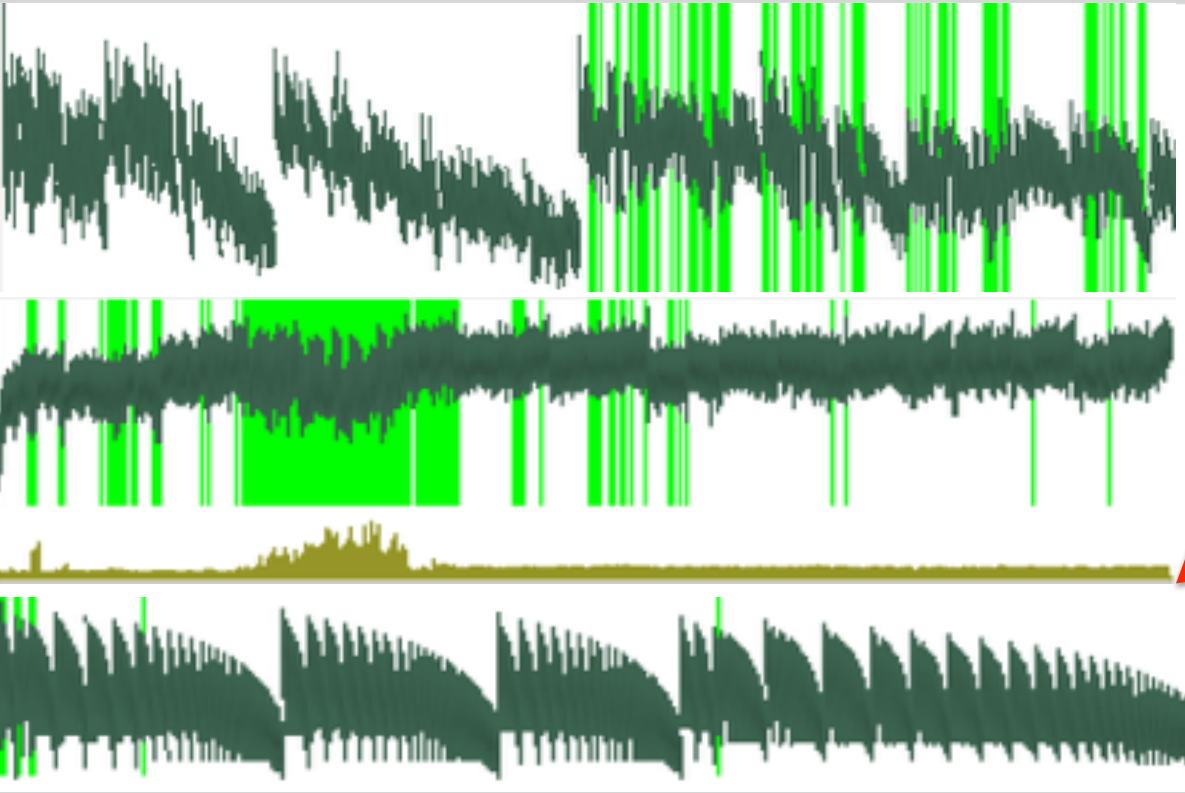
Maxim Shishmarev

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

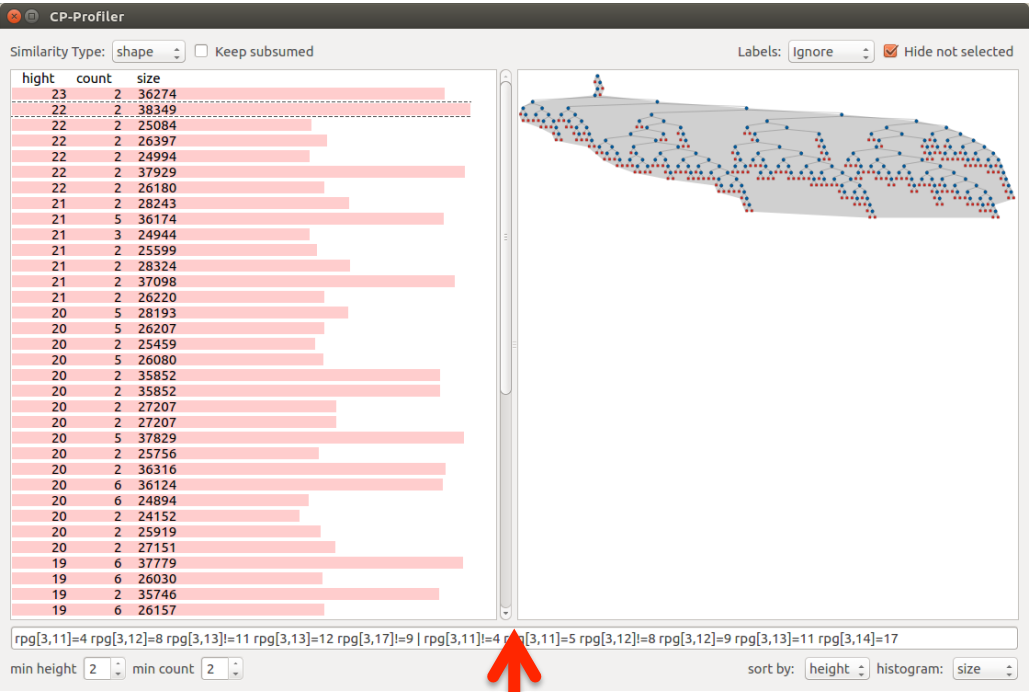


Solutions (light green) found after lots of search

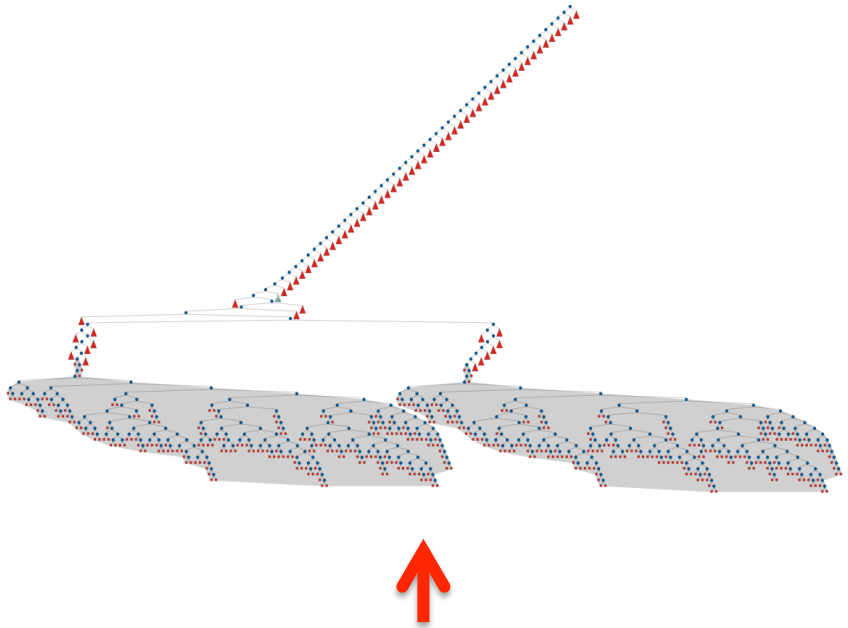
Other time-related info can be shown together

Visual patterns might indicate some property

Example: shape analysis to automatically focus attention

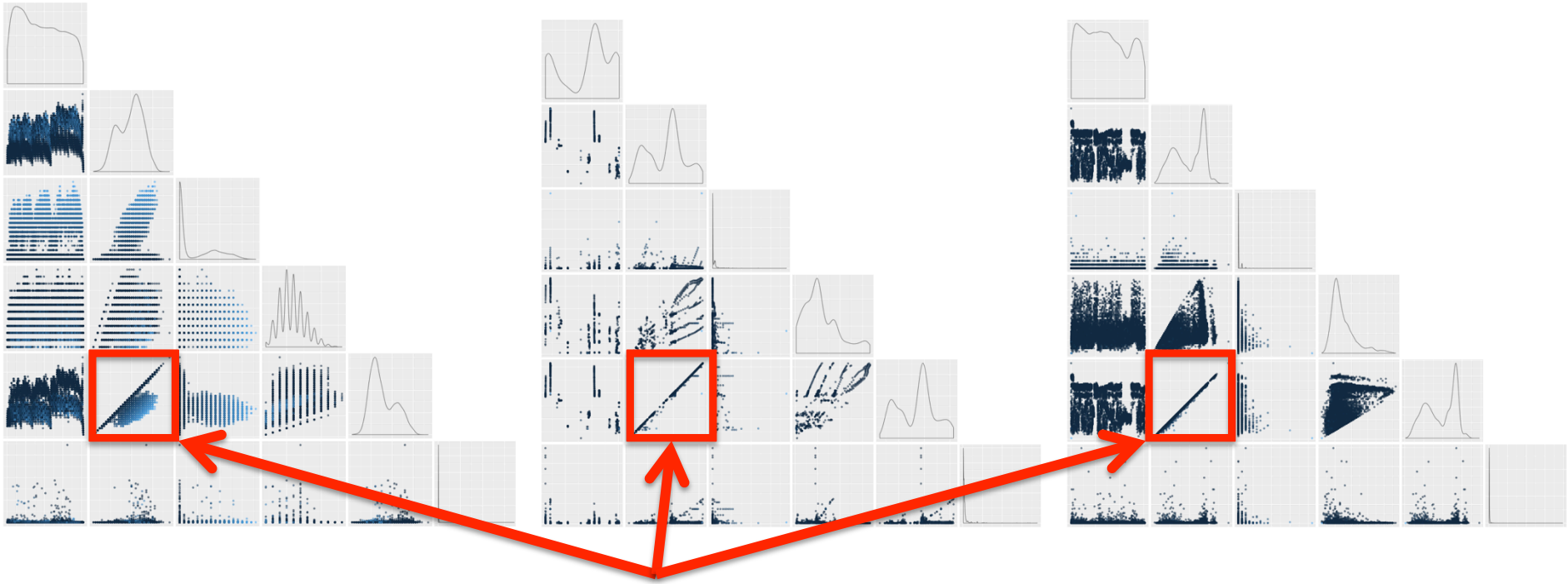


Histogram of “similar shape analysis” by size



Location of shapes within search tree

Example: statistical markers to automatically focus attention

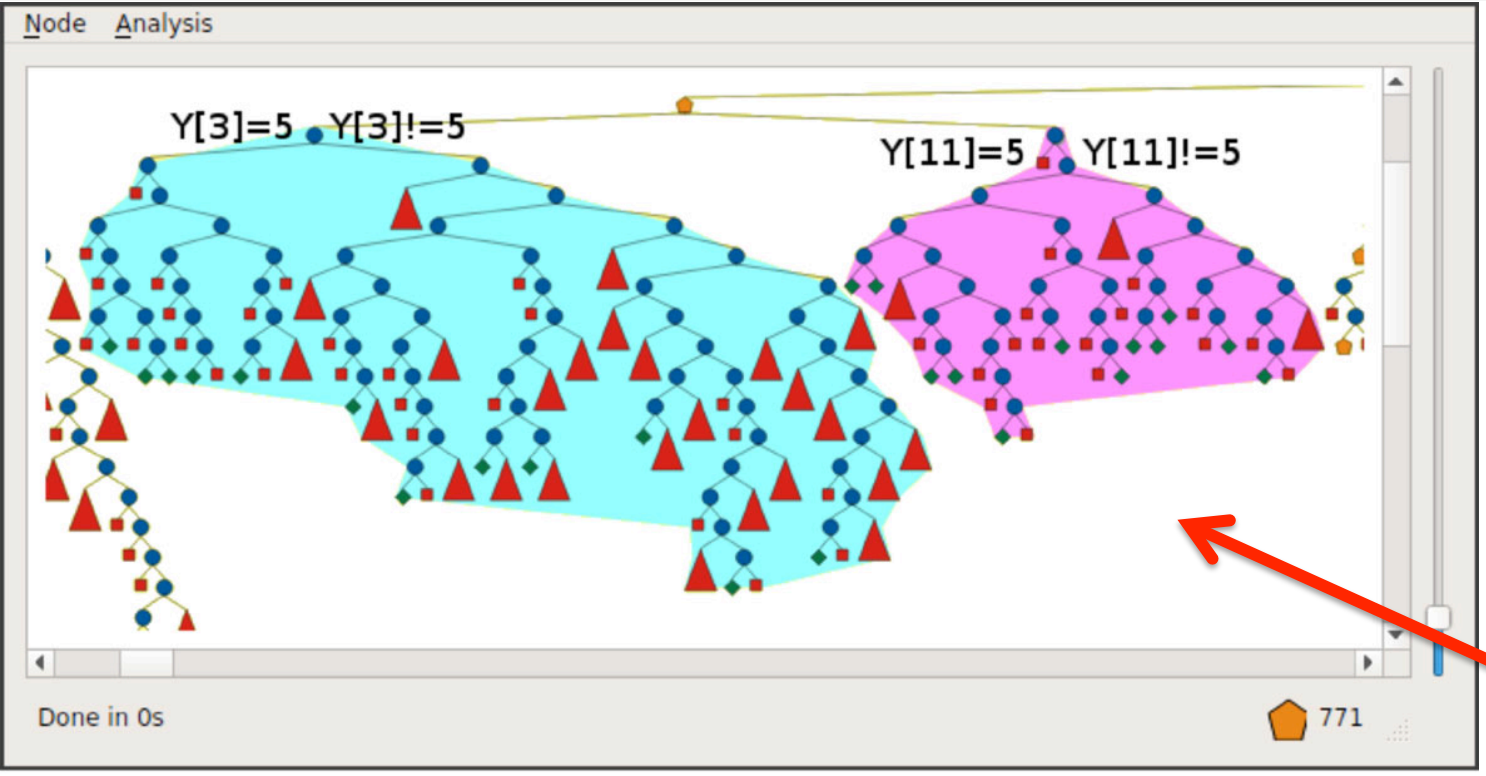


Effective versus ineffective backjumping

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

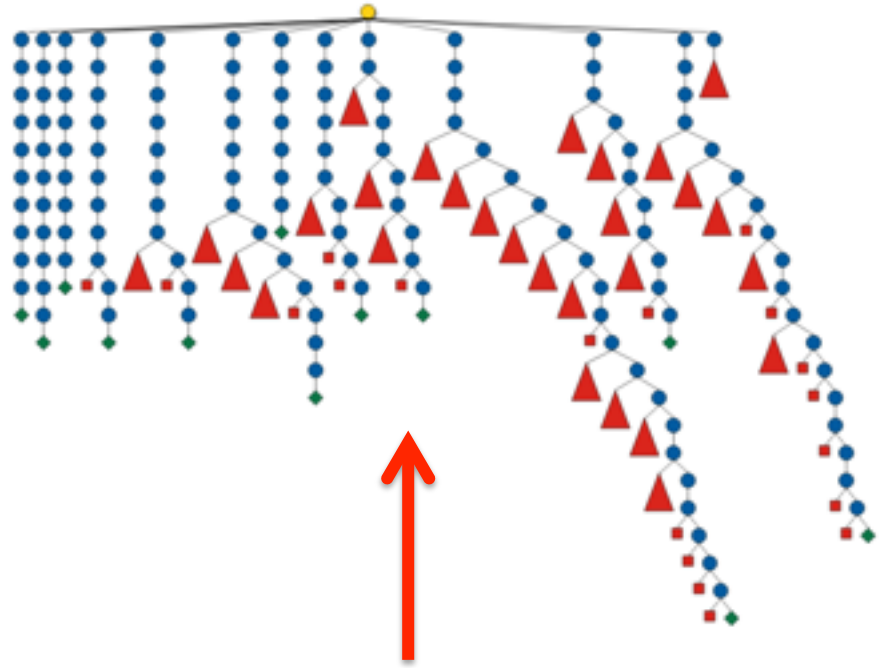
Example: comparing two executions of the same model via tree merging



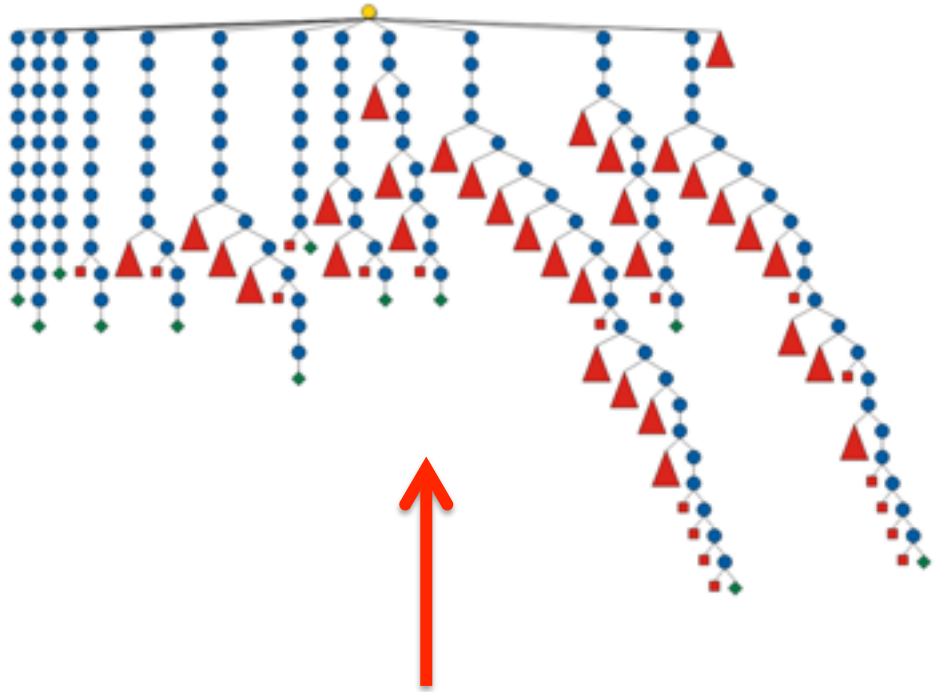
	1	2
27	21	15
28	21	17
29	25	15
30	29	3
31	1395	73
32	17	13
33	3	3
34	3	3
35	3	3
36	3	3
37	3	3

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

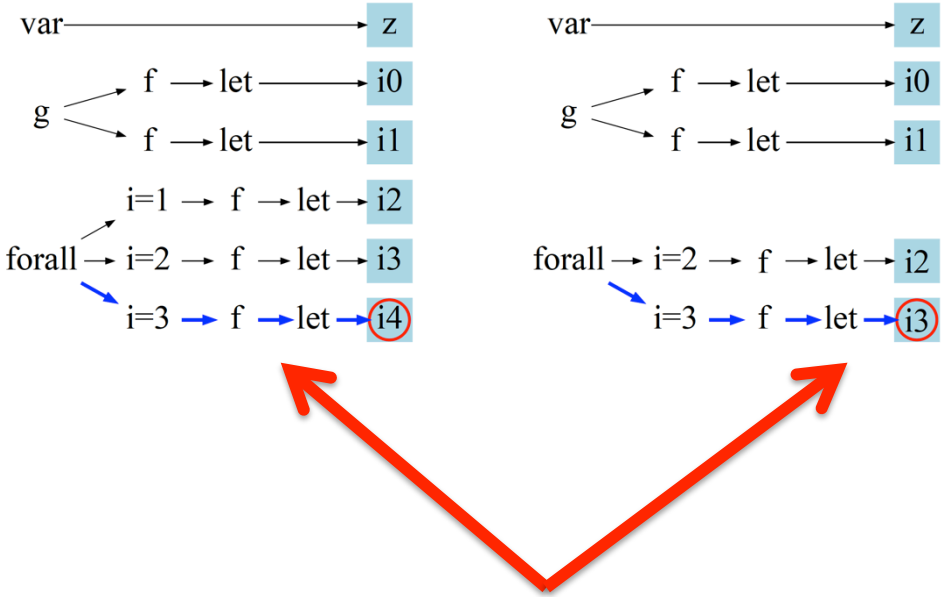
```
% a.mzn
predicate f(int: k) = let { var int: x } in h(x, k);
predicate h(var int: x1, int: x2);
```

```
% b.mzn
include "a.mzn";

predicate g(int: j) = f(j) /\ f(j+1);

constraint g(4);
var 1..3: z;
constraint forall(i in lb(z)..ub(z))(f(i));
```

i0 i1 i2 i3 i4 i=1 i=2 i=3

```
% b.fzn
var 1..3: z;
var int: i0;
var int: i1;
var int: i2;
var int: i3;
var int: i4;
constraint
    h(i0,5)
 /\ h(i1,4)
 /\ h(i2,1)
 /\ h(i3,2)
 /\ h(i4,3);
```

```
% b.fzn with paths
var 1..3: z; b:7.10-11
var int: i0; b:6.12-15 b:4.22-25 a:2.38-39
var int: i1; b:6.12-15 b:4.30-35 a:2.38-39
var int: i2; b:8.20-21 i=1 b:8.34-37 a:2.38-39
var int: i3; b:8.20-21 i=2 b:8.34-37 a:2.38-39
var int: i4; b:8.20-21 i=3 b:8.34-37 a:2.38-39
constraint
    h(i0,5)
 /\ h(i1,4)
 /\ h(i2,1)
 /\ h(i3,2)
 /\ h(i4,3);
```

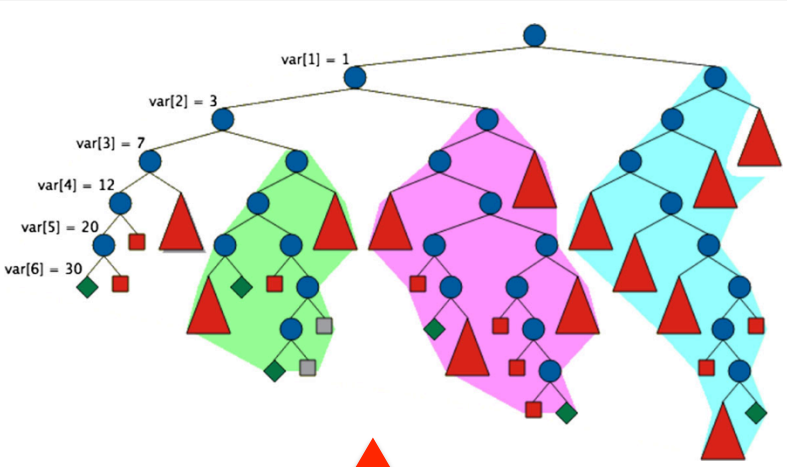


Same paths for variables named differently by the compiler

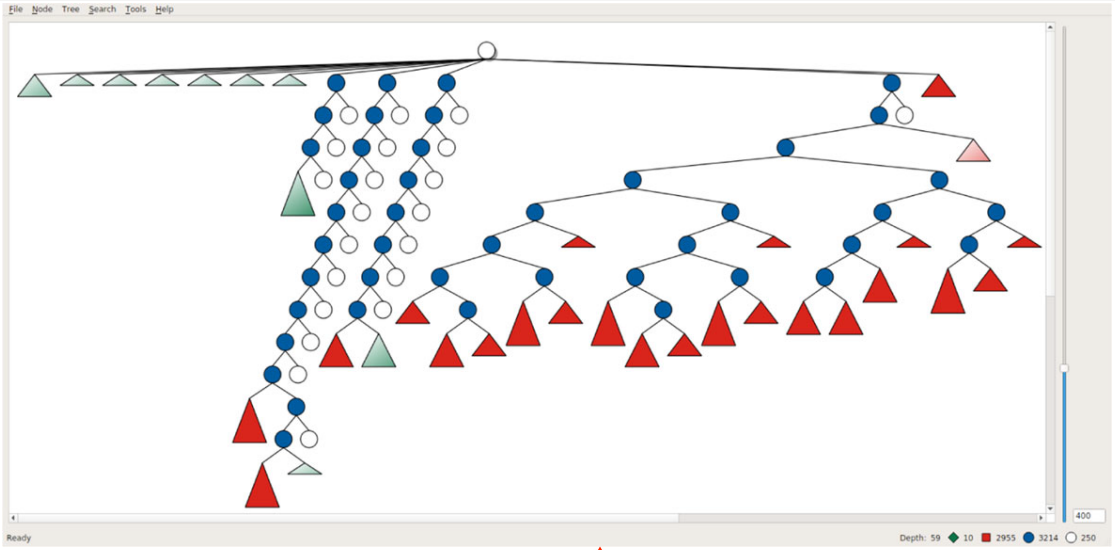
What are we missing? Supporting different searches and paradigms

- 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



Parallel search

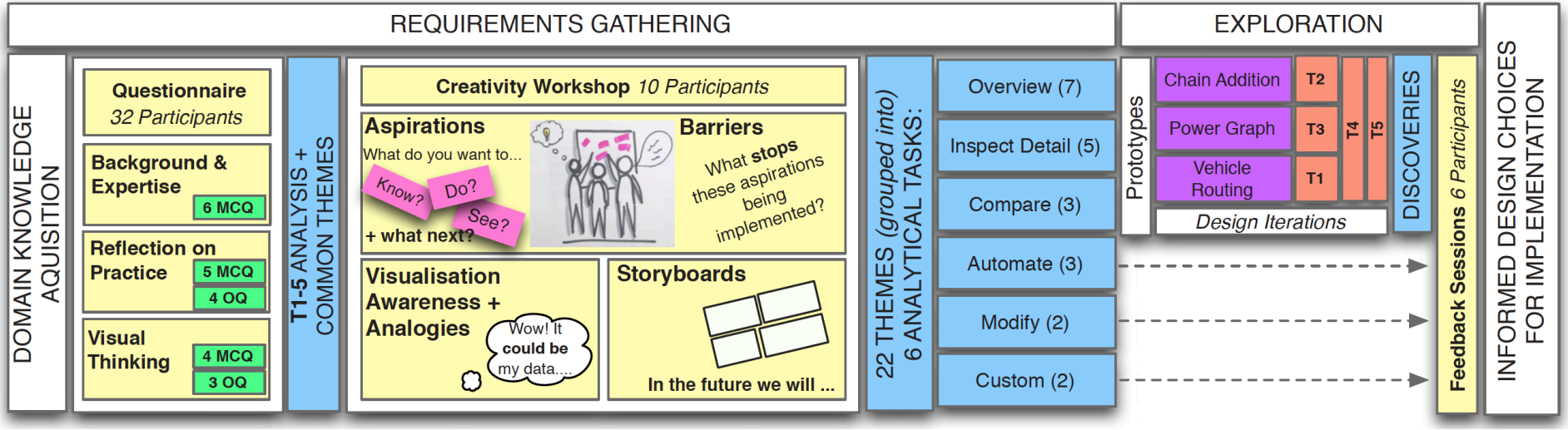


Restart search

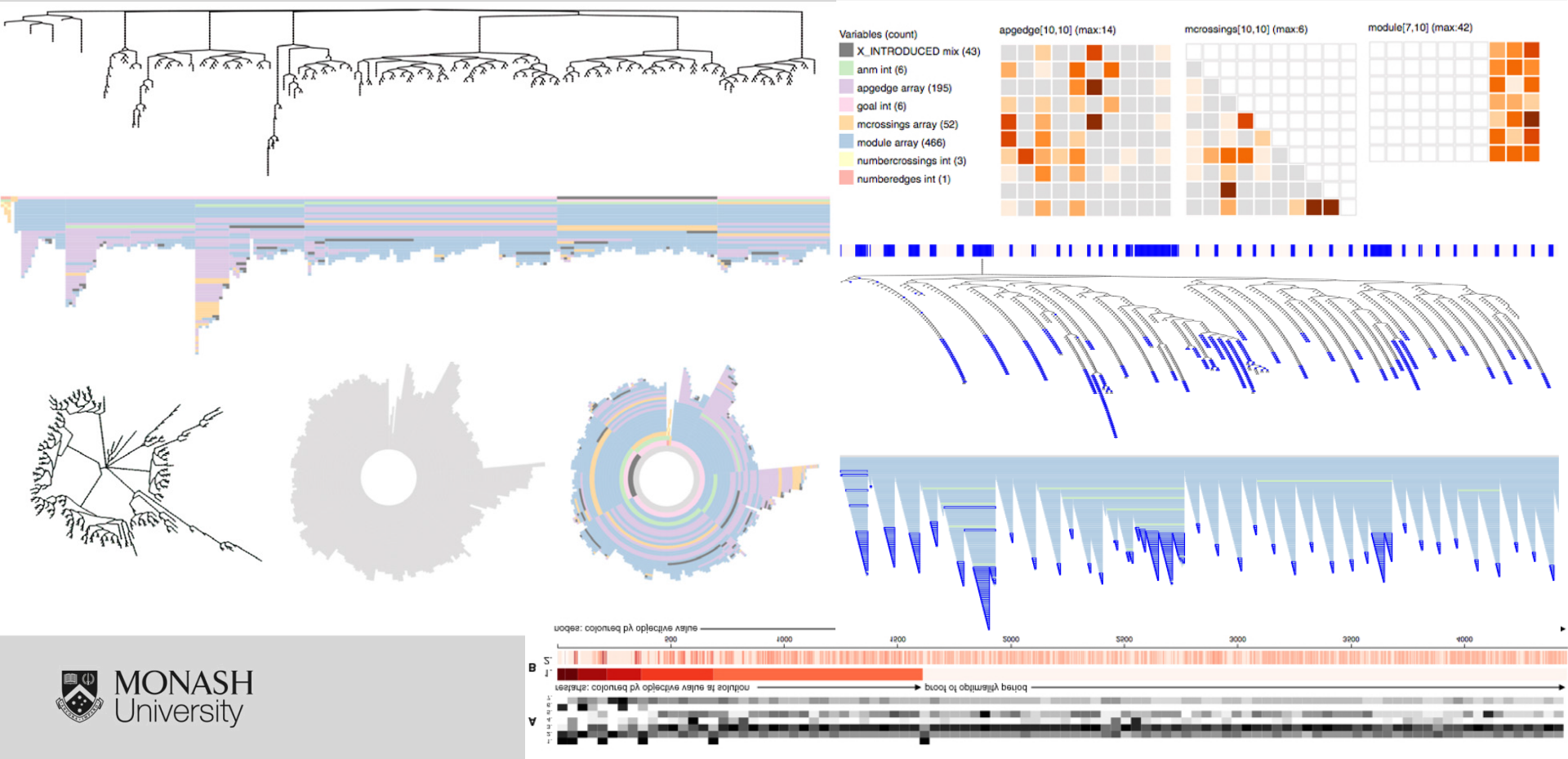
What are we missing? User testing

- 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

Example: user-centred design process for visual profiling tools in CP



Example: Many visualisation alternatives, many findings, but more needed



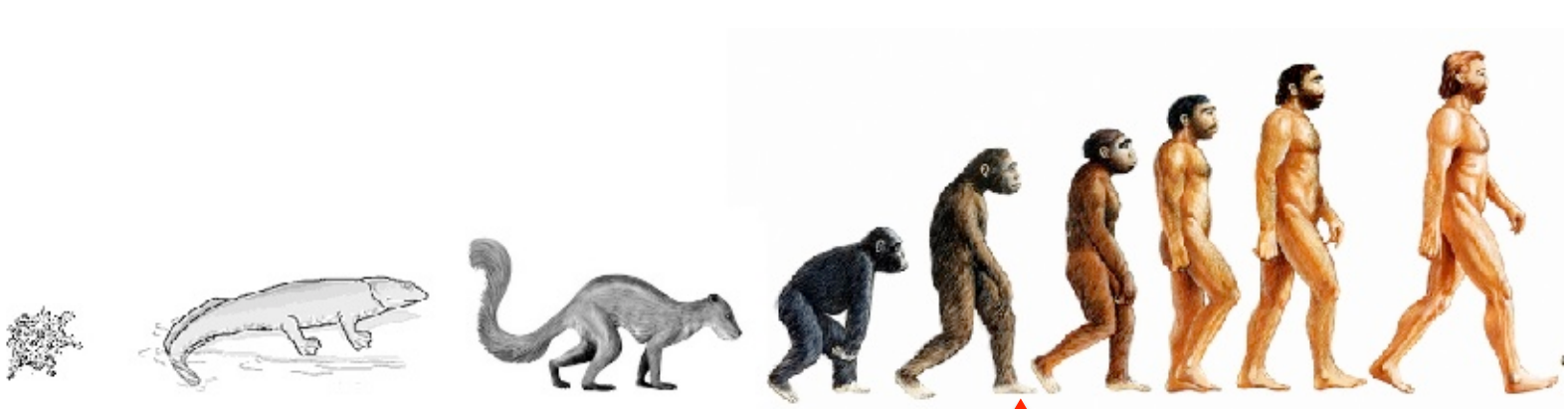
MONASH University

What are we missing? Connecting back to the model

- Since:
  - The ultimate goal is to figure out how 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/visualistion for constraint programming

We are…



here?

MONASH University

Thanks for listening!