Visualization for Constraint Programming

Helmut Simonis

CRT-AI CP Week 2025

Licence

Acknowledgments

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 12/RC/2289-P2 at Insight the SFI Research Centre for Data Analytics at UCC, which is co-funded under the European Regional Development Fund.
A version of this material was developed as part of the ECLiPSe ELearning course: `https://eclipseclp.org/ELearning/index.html`. Support from Cisco Systems and the Silicon Valley Community Foundation is gratefully acknowledged.

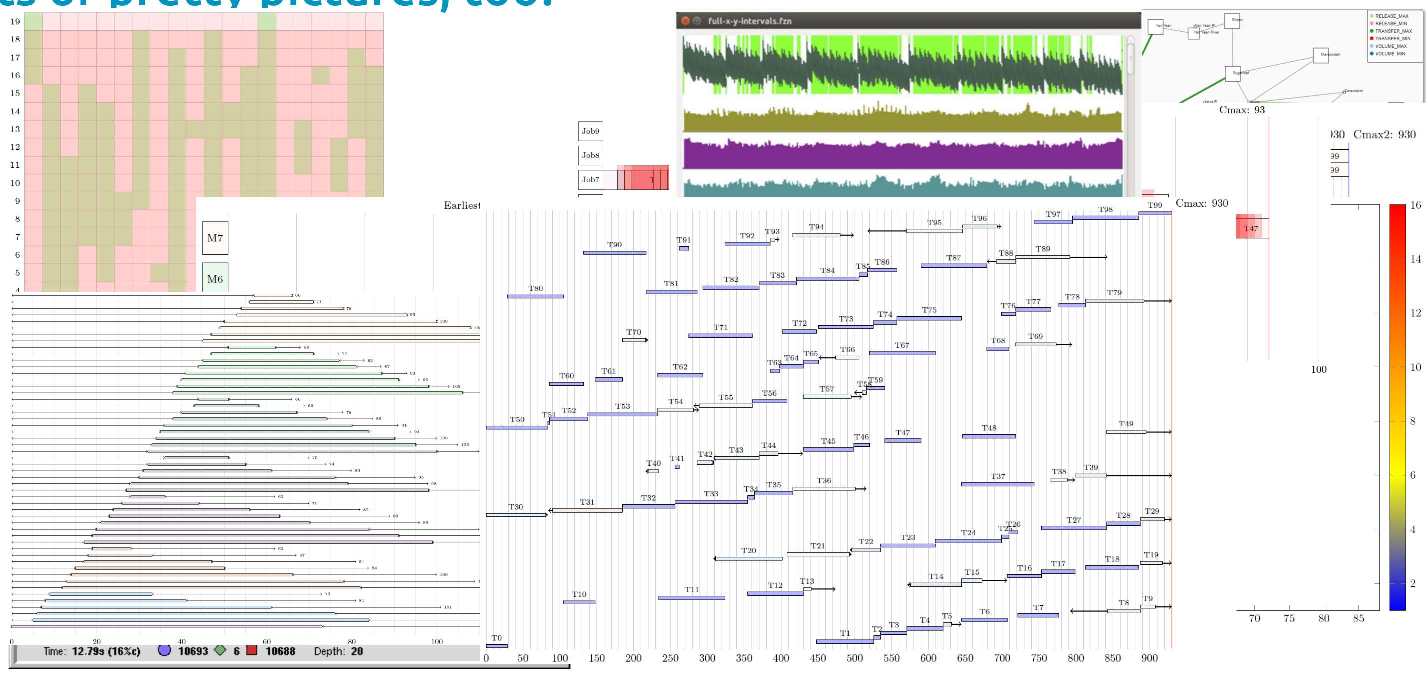
Origins

- Derived from Tutorial at CP 2021 (`https://cp2021.a4cp.org/tutorials.html`)
- By Helmut Simonis, Insight/UCC and Guido Tack, Monash University
- Video at `https://www.youtube.com/watch?v=AI-ZfQtMLAU`

Take-Away Message

- Visualization can help at different stages of the development process
- Discover problems early
- Understand what is happening without drowning in details
- Easily communicate with stakeholders without CP experience
- Decide how much integration with solver you need
- Complex use cases require visualization to be practical
- Generic vs. application specific visualization

Lots of pretty pictures, too!



Outline

Background (Simonis)

- Partner in the ASSISTANT (`https://assistant-project.eu/`) H2020 ICT-38 project
- Visualization and Constraint Acquisition are part of WP4 (Scheduling and Production Planning)
- Focused on industrial case studies from Siemens Energy and Atlas Copco (flow-shop variants)

Background (Tack)

- Monash University, Melbourne, Australia
- MiniZinc (`https://www.minizinc.org`) and Gecode (`https://www.gecode.org`)
- Implemented Gist (search tree visualisation), MiniZinc IDE (includes visualisation API), industry projects

Full Scale Development Process

- Considering the full process of building a CP based application
- Not just solving a given benchmark problem
- Involves many people, a lot of different tools

Stakeholders (One person may have multiple roles)

- Application end user
- Domain expert
- Management
- Customer IT department
- System designer
- CP model developer
- Integration/front-end developer
- Solver developer

Layers of Description (Scheduling Example)

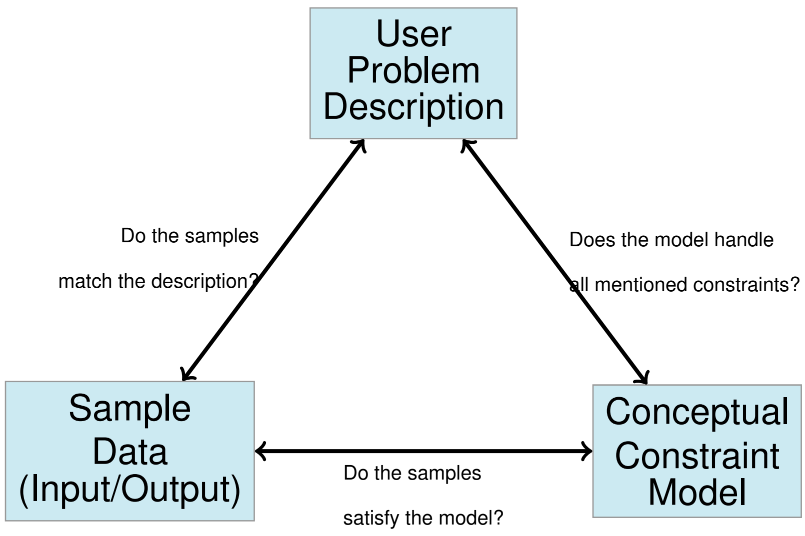
Modelling Layer	Physical Plant	Enterprise IT	Application Data Model	User Constraint Model	Backend Model
Stakeholder	End User	Customer IT	System Designer	CP Modeler	Solver Developer
Example Technology	Excel	SAP	Java	MiniZinc	Chuffed

Layers of Description (Academic Focus)

Modelling Layer	Physical Plant	Enterprise IT	Application Data Model	User Constraint Model	Backend Model
Stakeholder	End User	Customer IT	System Designer	CP Modeler	Solver Developer
Example Technology	Excel	SAP	Java	MiniZinc	Chuffed

Most CP Papers

Triangle of Information



Roles of Visualization Considered Here

- Help to build model
- Explain results to different stakeholders
- Build confidence in system
- Allow communication between stakeholders
- Present results to management/funding agency/general public

Focus: Building and Maintaining the Model

- We will focus on using visualization to help develop the model
- For this we consider a problem classification of typical issues arising during development
- Visualization helps with resolving the issues, but does not solve them itself
- We are concentrating on using visualization as a tool for the developer
- Also used when maintaining a working system
  - You may not remember all the details of the model!
  - You should still understand the information in the visualization

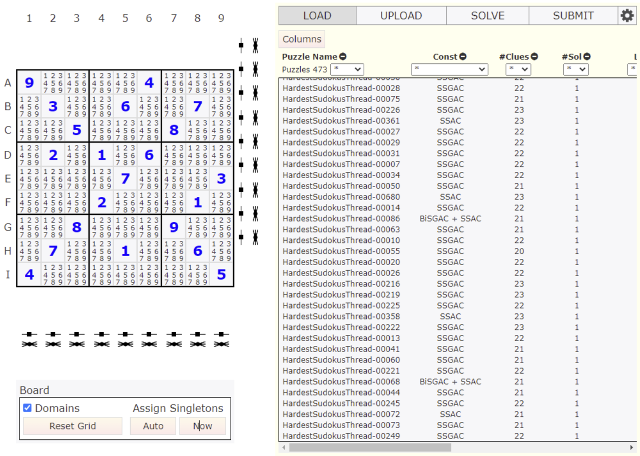
Other Roles of Visualization (not covered in detail)

- Improve the solver itself
- Understand what the solver is doing
- Teaching aid
- Outreach

Important Distinction

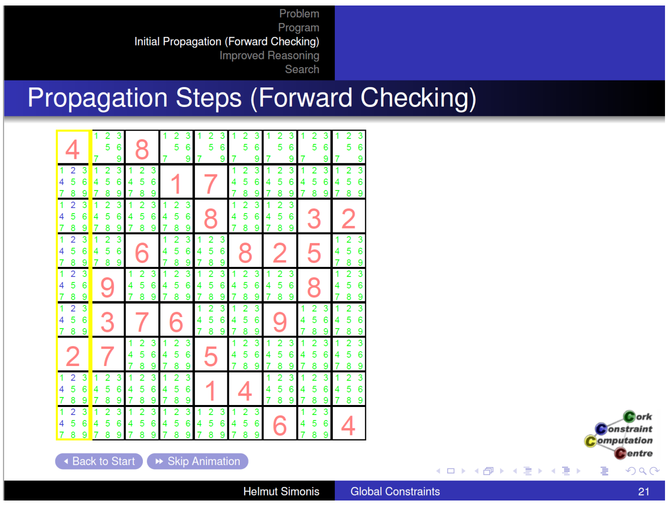
- A generic visualization toolkit (used for multiple problems)
  - May need adaptation/ does not handle full range of problems encountered
  - Available at start of project
  - Reuse of components/improvement across multiple projects
- A problem specific visualization
  - Can be customized to use/handle specific problem features
  - Not available at start of project
  - Development cost can be prohibitive for single project

Problem Specific: Sudoku Tool [Howell et al., 2018]



- User control of solving process
- Understand the impact of consistency techniques
- Very much based on specific problem structure

Generic Tool: CP-Viz [Simonis et al., 2010]



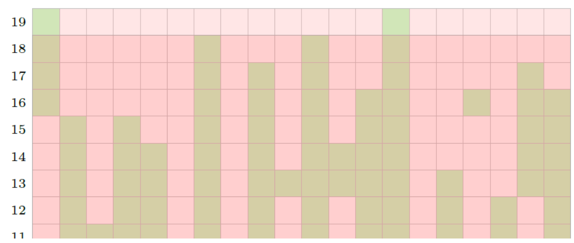
- Less interaction/user control
- Available from start of development
- Integrated into report/slide generation

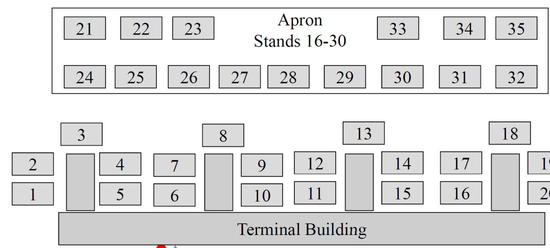
Common to Both Examples

- Emphasis on solution process, not on solution
- Understand how CP solves the problem
- Ways of configuring solver to change process
- Not focussed on
  - Modelling the problem
  - Checking that the solution is correct

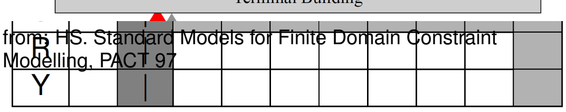
Fundamental Types of Visualization for CP

- Check an assignment
  - May or may not satisfy constraint
- Capacity view
  - How tight is the constraint, no assignment needed
- Explain failure
  - No assignment, constraint cannot be satisfied
- Explain progress during search
  - Partial assignment, show propagation
- Show solution
  - Often application specific





from: HS. Standard Models for Finite Domain Constraint Modelling, PACT 97



from: HS. ECLiPSe ELearning Course

Used at Different Stages

Assignment Checker check manual or external solutions

Capacity View check for input data consistency

Failure Explanation model setup failed

Propagation View during search, detailed understanding

- Often too much information

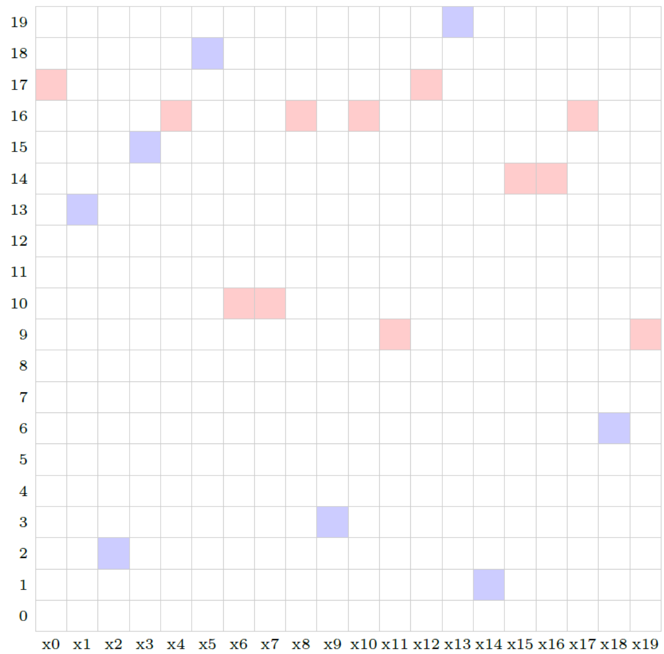
Solution View displaying results for end-user

- Often useful to translate into end-user concepts

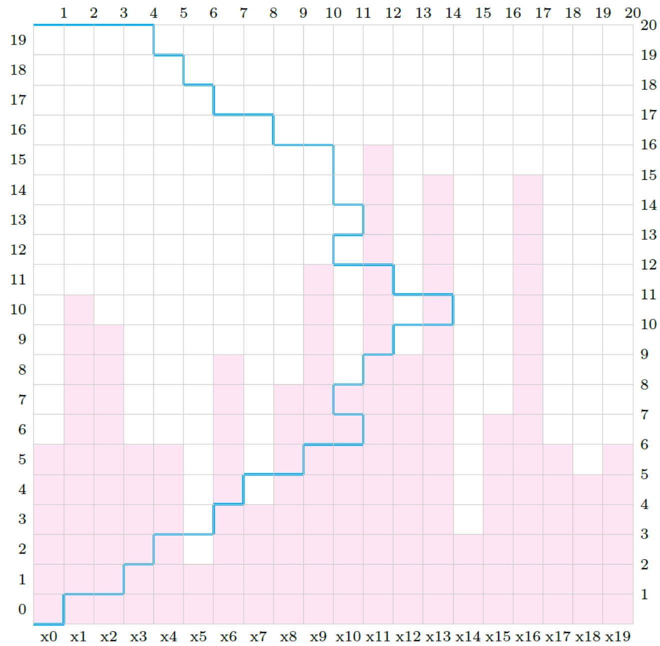
Example: Alldifferent Visualizations

- We use the same representation as variable/value matrix
  - Columns: variables
  - Rows: values
- Other visualization forms possible (vector, bi-partite graph)
- Different visualizations use different APIs
  - Values only
  - Domain bounds/explicit domains
  - Requires information about methods used in solver (domain/bound consistency)

Alldifferent: Inconsistent Assignment

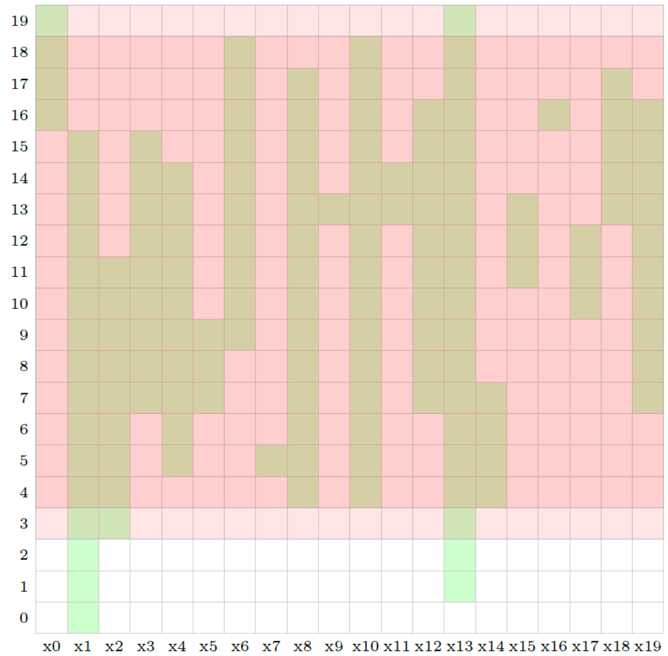


Alldifferent: Capacity View



- Show many values are in the domain of each variable
- How many variables contain a value in their domain
- Highlights potential bottlenecks

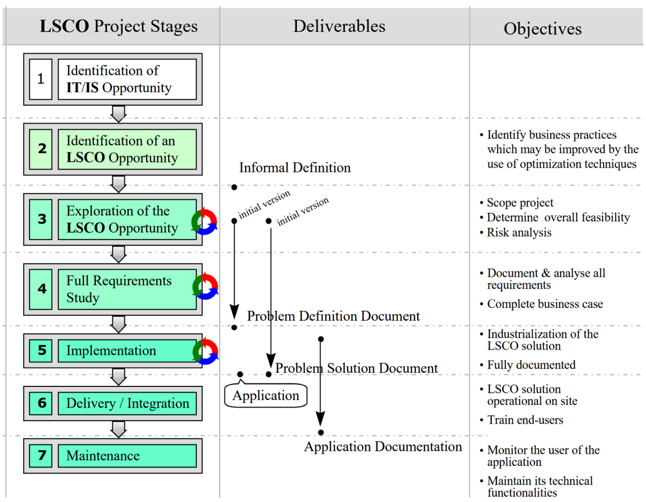
Alldifferent: Explaining Infeasibility



- Show value range that contains too many competing variables (bound consistency)
- All or only one explanation?
- Largest or smallest infeasible set?
- Dual explanation possible

Development Process

- Surprisingly little work describing how to develop and maintain models
- Foundation: European CHIC-2 project [Gervet, 1998]
- Not aware of papers in the last 10 years
- Most training material focused on
  - How to use a specific system
  - Explaining the principles behind CP
- Problems occur/re-occur at different points in project timeline



Classifications of Problems with Models

1. It does not compile
2. A known solution is not accepted by the model (covered in slides)
3. *The model is inconsistent and rejected at startup* (covered in presentation)
4. *The model is inconsistent and fails after some search*
5. *The solver starts, but does not finish, returning neither yes or no*
6. A solution is found, but the user rejects it, because it violates some constraint that was not discussed before
7. *An initial solution is found, but no further improvements are found in a reasonable time*

Classifications of Problems with Models (II)

8. The model has been working for some time,
   but suddenly it stops finding a solution, or violates a constraint
9. When the best solution is shown to the users,
   they immediately find a change that improves the quality
10. The solver says it is optimal, but there is a better solution
11. *The solver finds a solution, which satisfies all constraints, the user accepts it as correct, but doesn't like it, and wants a different one*

Comments

- For some solvers, certain problems are not differentiated
  - MiniZinc: (3) and (4) look the same, as there is no separate consistency check at root node
  - Choco: You can ask to propagate constraints after setup, to separate (3) and (4)
  - Prolog based: Individual constraint may fail at posting, allowing fine grain analysis of failure (3)
- Access to internal state during search may not be possible/is difficult
- Reason for failure of constraint rarely provided

A known solution is not accepted by the current model

- This can have different reasons
  - The data models do not match
  - A constraint stated was misunderstood
  - A constraint is soft, not hard
  - The known solution is not the plan, but the actual operation
- Role of visualization

Running Example: Scheduling Problem

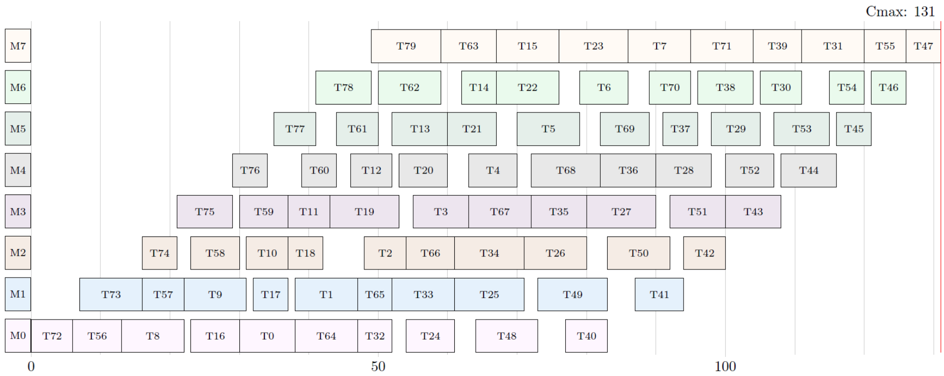
- Flowshop type problem
- Jobs consist of series of tasks processed on machines in same order
- Precedence between tasks of same job
- Disjunctive machines
- Tasks require operator during first part of run
  - Overall cumulative manpower limit

Conceptual Model (MiniZinc)

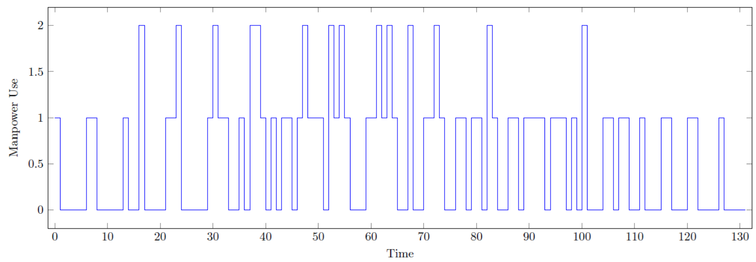
```
% variables
  array[T] of var 0..ub:start;
  var lb..ub:cmax;

% constraints
  constraint forall (i in T)
             (cmax >= start[i]+duration[i]);
  constraint forall (p in P)
             (start[prec[p,2]] >= start[prec[p,1]]+
                                  duration[prec[p,1]]);
  constraint forall(s in S)
      (cumulative([start[i]|i in T where stage[i]=s],
                  [duration[i]|i in T where stage[i] = s],
                  [1|i in T where stage[i] = s],1));
  constraint
      cumulative([start[i]|i in T],
                 [manpowerDuration[i]|i in T],
                 [1|i in T],manpower);

solve minimize cmax;
```

- Data definition not shown for brevity

Example Solution (10 Jobs/8 Machines)



- Typical visualization: Gantt Chart
- Machine view
- Other views available



- Typical visualization: Resource Profile
- Show operator use over time

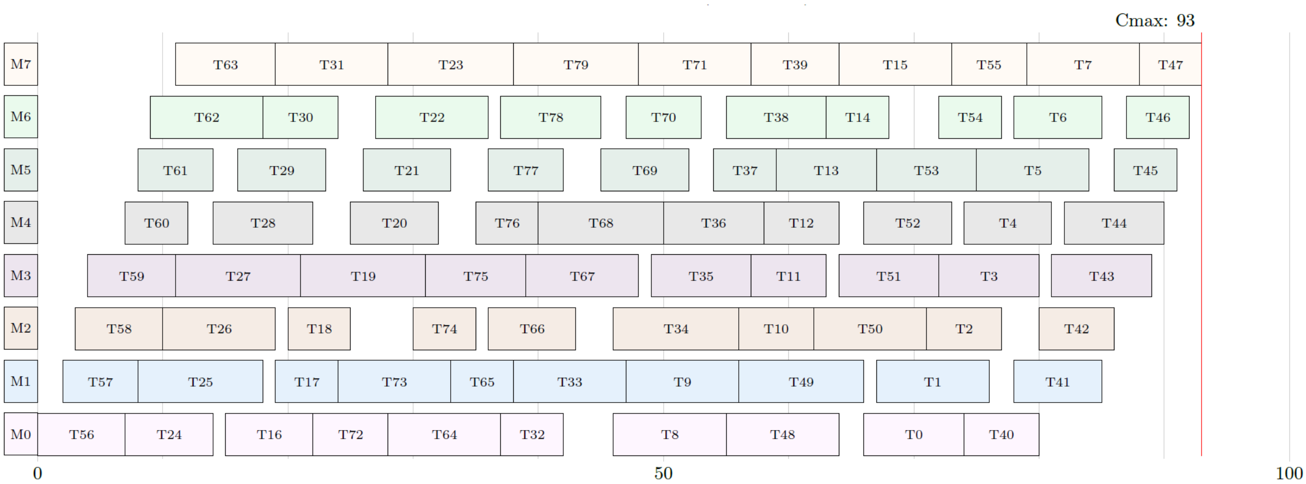
Conceptual Models do not Match

- Different concept of time
  - Calendar time
  - Working time
- Rework tasks (added tasks)
- Skipped tasks for some jobs
- Preemption allowed
- Task stretching over downtime

A Constraint was Misunderstood

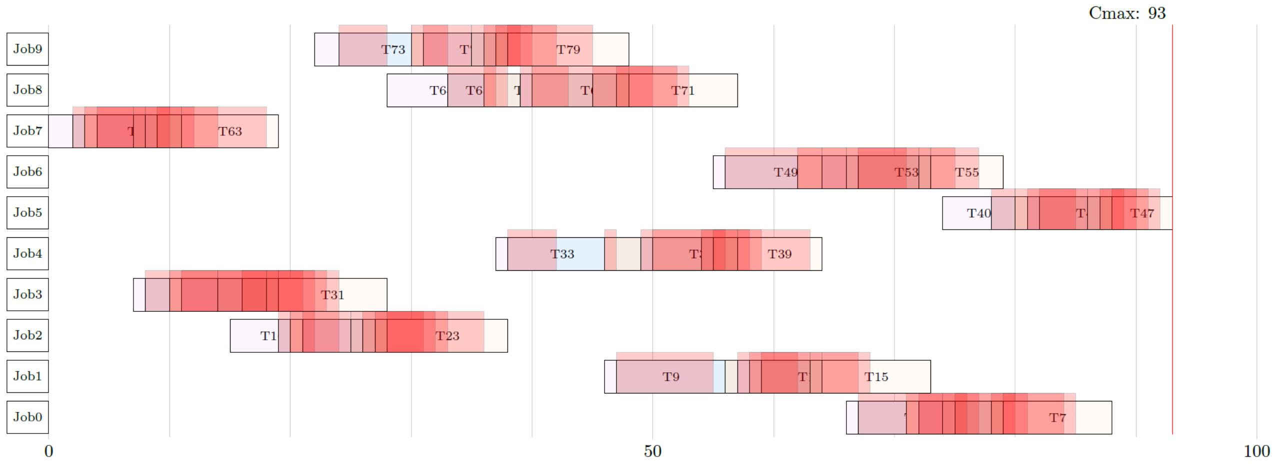
- Very rare that a formal description of constraints is given
- In most cases, problem derived from interviews with domain experts/end users
- Language barrier:
  - CP Modeller does not (yet) speak language of application domain
  - End users are not familiar with CP/optimization
- Example in Manufacturing:
  - Pipelined production, not strict end-to-start precedence

Modified Problem: Gantt Chart with Pipelining (Machine View)



- Nothing to see here, we need other views to identify problems

Modified Problem: Gantt Chart with Pipelining (Job View)



- Display in job view highlights numerous task overlaps
- Shades of red show how many tasks are active at the same time
- Uses semi-opaque overlay to indicate problems

Modified Problem: Gantt Chart with Pipelining (Task View)

- Task view separates tasks, shows pipelining of production steps
- Next task of job can start as soon as some items of previous task are completed
- Must run until last items of previous tasks are finished

Role of Visualization

- Check that conceptual model matches sample data
- Quickly highlight conflicts in sample data
- Visual Checker
  - Does not replace automatic checks on data or solutions
  - If nobody checks the visualization, no alarm is raised
- Understand problems with units
  - Time resolution
  - Stock levels/consumption

The model is inconsistent and rejected at startup

- Two main concepts:
- Find minimal correction sets (MCS) / minimal unsatisfiable set (MUS)
  - Each MUS (there can be many) explains why the model does not provide a solution
  - Each MCS (there can be many) explains how the model can be made satisfiable
  - Overview on explanations in CP during PTHG21 workshop `http://www.cs.ucc.ie/~bg6/data/pthg2021.pdf`[Gupta et al., 2021]
- Explain unsatisfiable global constraints
  - Even if you know that a single global constraint is unsatisfiable, you need to understand why.
  - This is not trivial

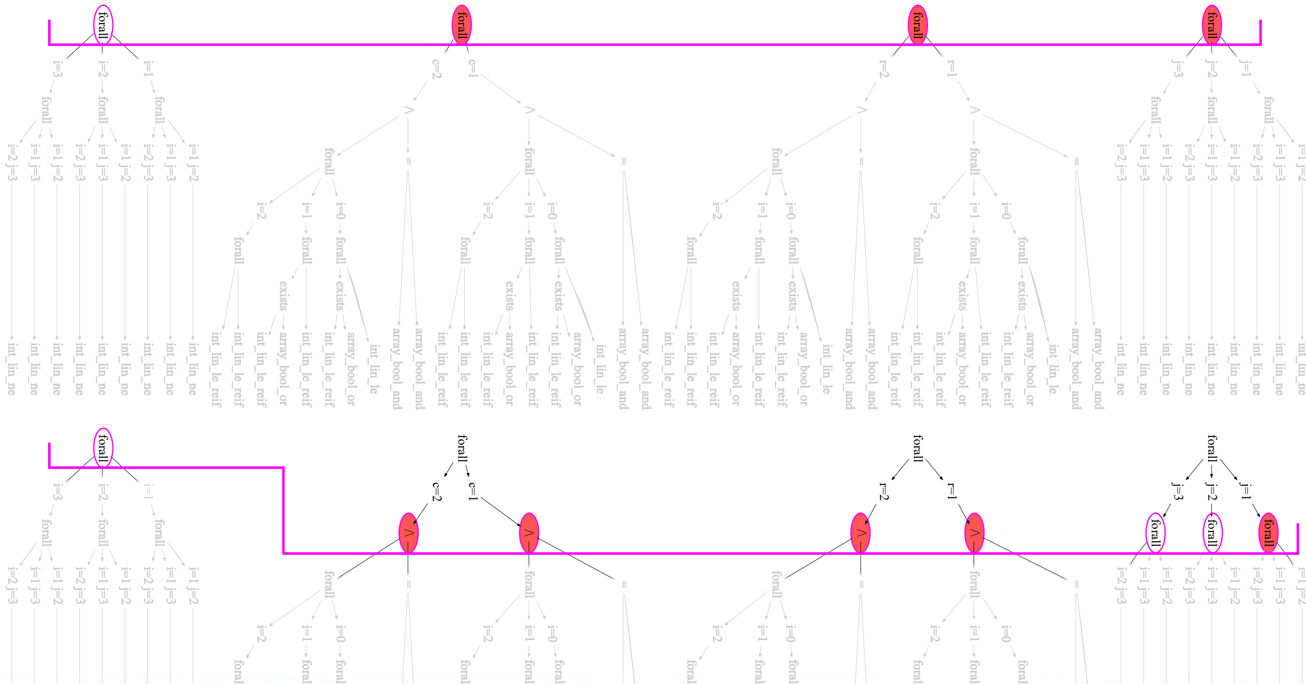
Finding MCS/MUS

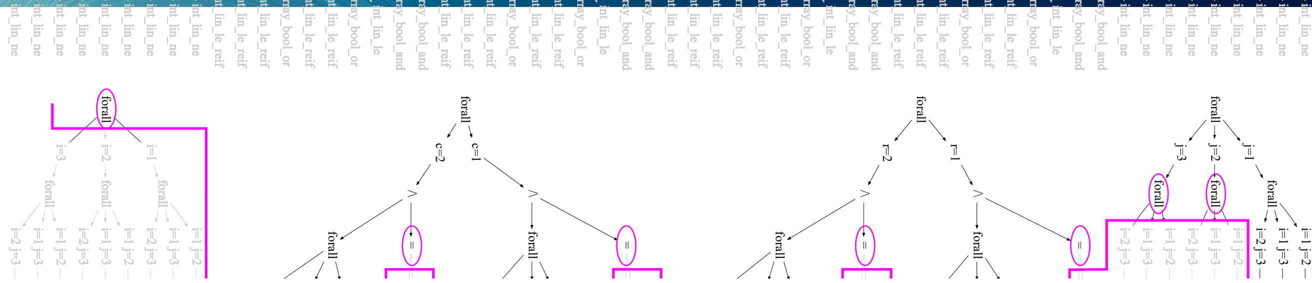
- Algorithms typically use *search*
- Enable/disable constraints systematically
- Problem: how to interpret results?
- Need to understand MCS/MUS at the level of the conceptual model, not the solver-level model

Example: Latin Squares MUS

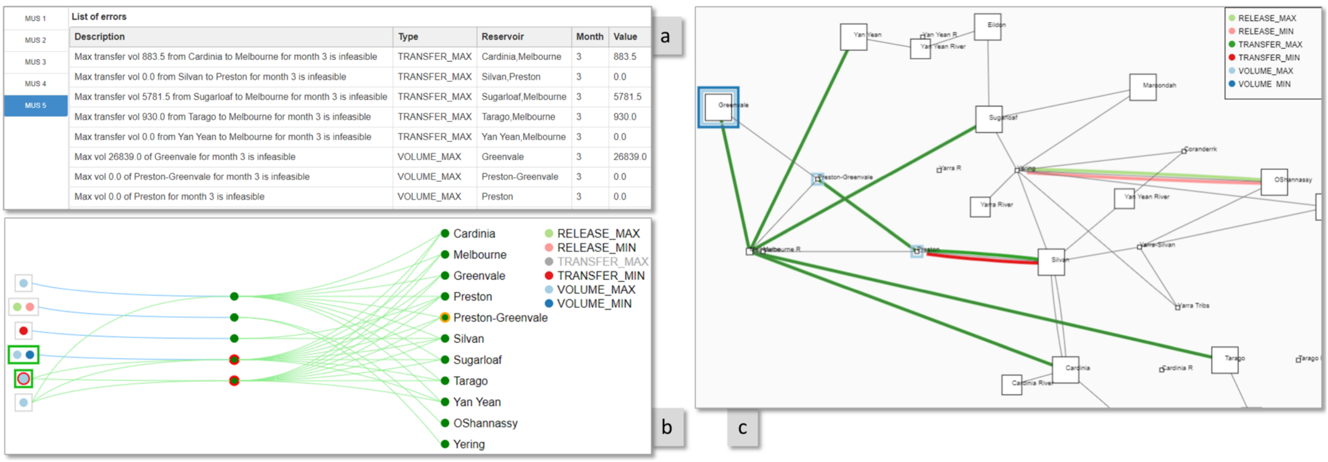
(This slide is a placeholder for a live demo in the presentation)

Visualising MUS in terms of the model



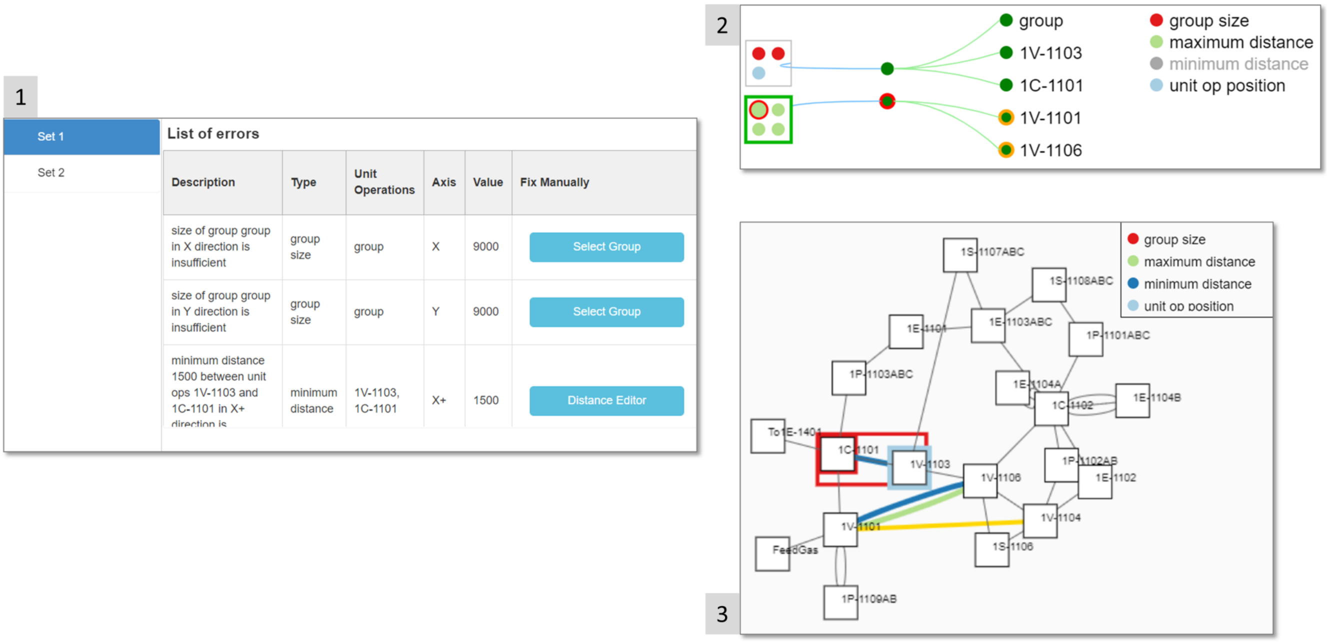


Example: Visual MUS exploration I [Senthooran et al., 2021]



Description	Type	Reservoir	Month	Value
Max transfer vol 883.5 from Cardinia to Melbourne for month 3 is infeasible	TRANSFER_MAX	Cardinia,Melbourne	3	883.5
Max transfer vol 0.0 from Silvan to Preston for month 3 is infeasible	TRANSFER_MAX	Silvan,Preston	3	0.0
Max transfer vol 5781.5 from Sugarloaf to Melbourne for month 3 is infeasible	TRANSFER_MAX	Sugarloaf,Melbourne	3	5781.5
Max transfer vol 930.0 from Tarago to Melbourne for month 3 is infeasible	TRANSFER_MAX	Tarago,Melbourne	3	930.0
Max transfer vol 0.0 from Yan Yean to Melbourne for month 3 is infeasible	TRANSFER_MAX	Yan Yean,Melbourne	3	0.0
Max vol 26839.0 of Greenvale for month 3 is infeasible	VOLUME_MAX	Greenvale	3	26839.0
Max vol 0.0 of Preston-Greenvale for month 3 is infeasible	VOLUME_MAX	Preston-Greenvale	3	0.0
Max vol 0.0 of Preston for month 3 is infeasible	VOLUME_MAX	Preston	3	0.0

Example: Visual MUS exploration II [Senthooran et al., 2021]



Explaining Failure of Single Global Constraints [Simonis et al., 2000]

- Approach 1: Integrate (parts of) constraint filtering into visualization
  - Example: Detect Hall sets for alldifferent, as shown above
- Approach 2: Provide debugging information about failure
  - Example: CHIP propagation events, oaDymPac trace format
- Approach 3: Use generic (SAT-based) methods to describe failures
  - Research challenge: How to present internal problem representation and learned clauses back to the user

The model is inconsistent and fails after some search

- Constraints are not strong enough to detect infeasibility
- Good chance of using stronger lower bounds
- MUS based methods work, but can be slow

The solver starts, but does not return yes or no

- The model may well be correct, but the search does not work
- The model is too tight, but the search space is too large to detect infeasibility
- Relaxing some resource limits should allow to find solutions
- Understanding what is happening requires some access to solver internals

Why does the search not find a solution?

- Two aims of analysis
  - Understand what is happening
  - Suggest change to improve behaviour

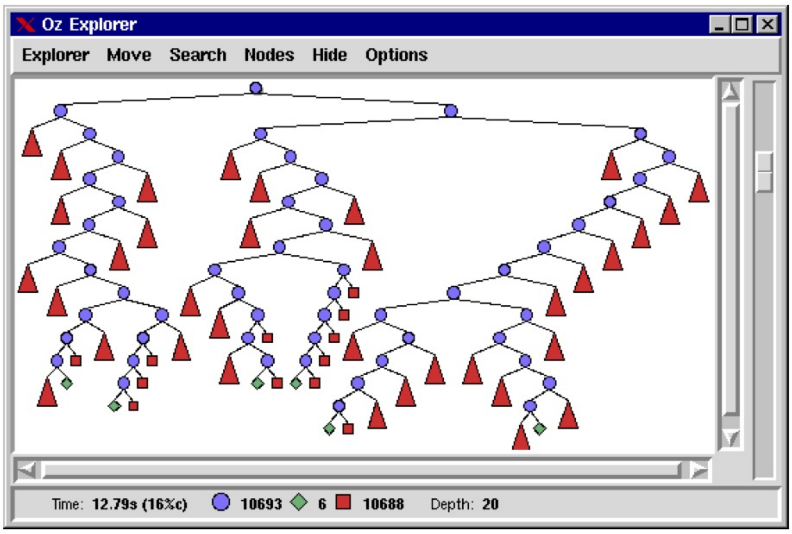
Visualizations

- Search tree display
- Search depth display
- Heatmap

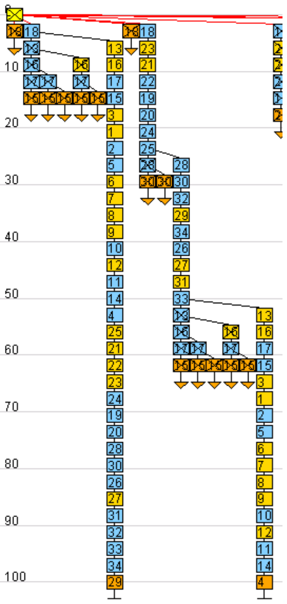
What can we learn?

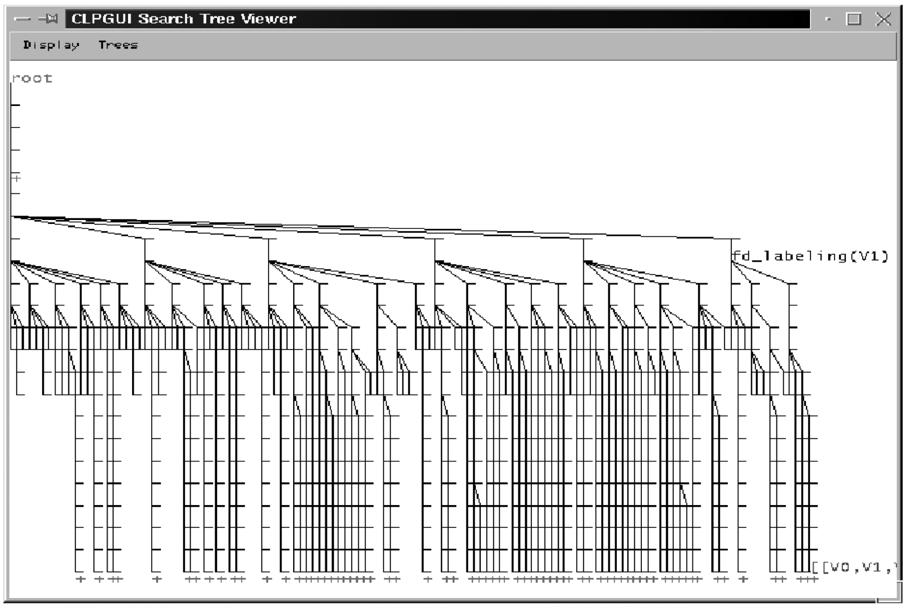
- How far do we progress in the search?
- How far do we backtrack?
- Is the value selection method working?
- Do we get stuck in an infeasible branch?
- Can we cut off infeasible branches by propagation?

Search Tree Display [Schulte, 1997]

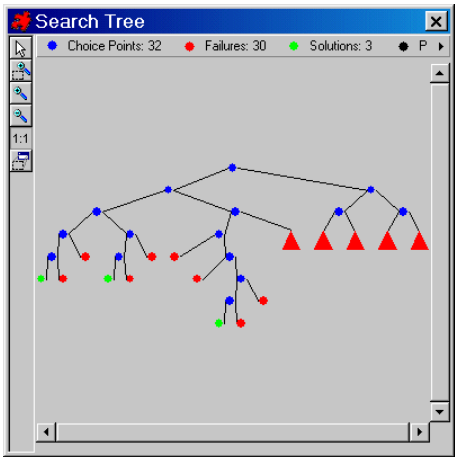


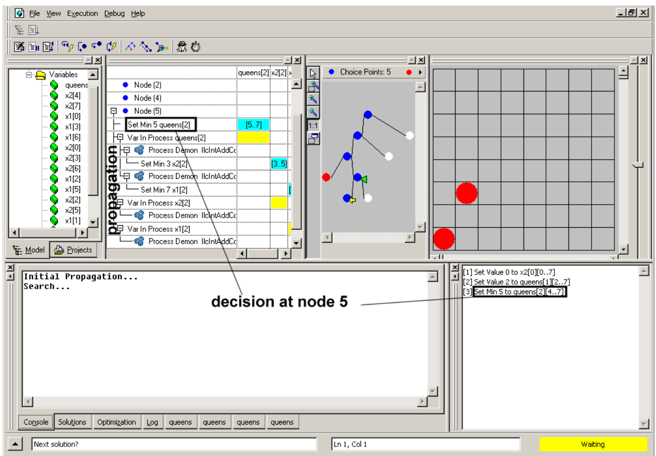
Search Tree Display [Simonis and Aggoun, 2000] [Fages et al., 2004]



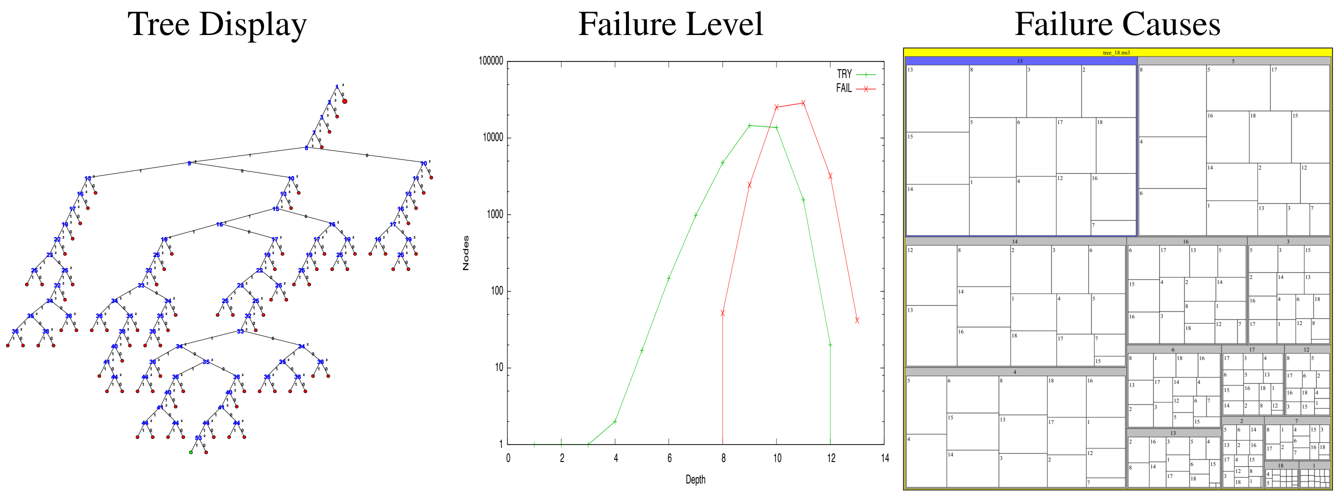


Search Tree Display: ILOG Solver Debugger (2009)

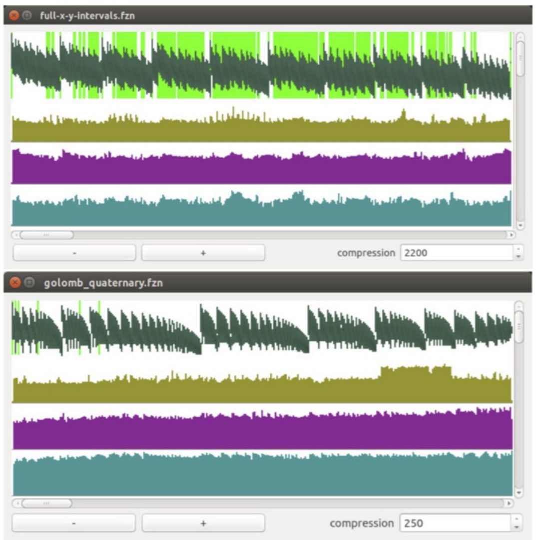




Search Tree Display [Simonis et al., 2010]



Pixel Tree [Shishmarev et al., 2016]



- See large-scale structure in very large trees
- Plot histograms on same horizontal scale (e.g. avg. failure depth, avg. domain reduction)

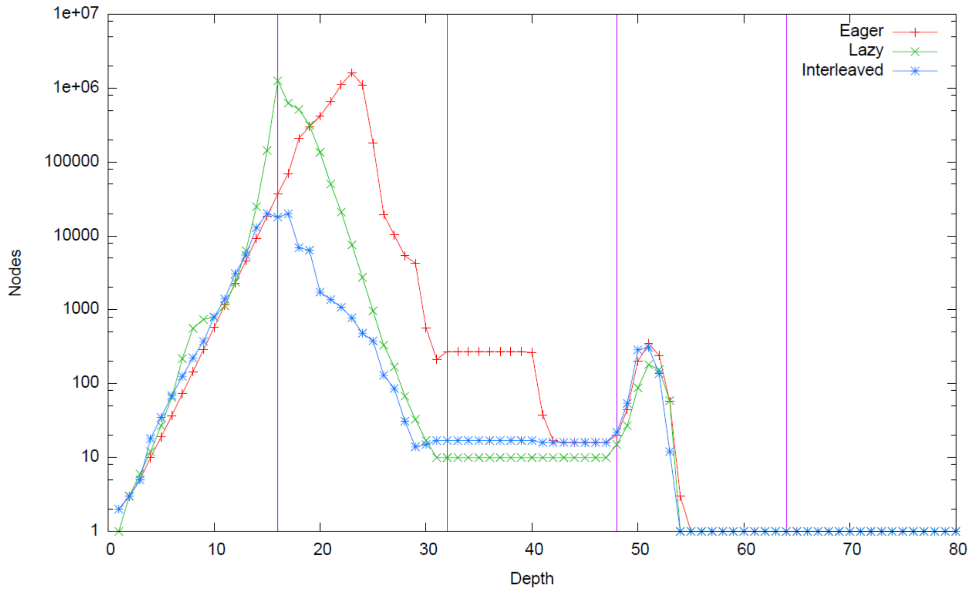
Example: Similar Subtree Analysis

(This slide is a placeholder for a live demo in the presentation)

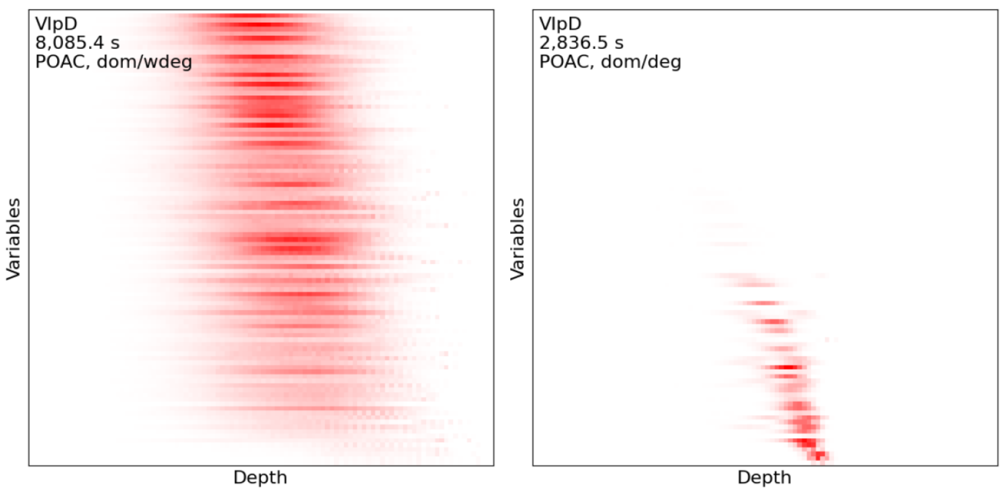
Search Tree Visualization: Challenges

- How to visualize learning, diving, restarts, backjumps, non-chronological search, best-first search, ... ?
- What can we learn from very large trees?

Search Depth Display [Simonis and O'Sullivan, 2011]



Heatmap [Howell et al., 2020]



- Which variables are assigned at which depth of the search
- Where does backtracking occur

A solution is found, but the user rejects it

- There may be a constraint that is missing in the description
- Visualization is key for discussion with users
- Users are very good at spotting problems
- Not so good at describing all constraints of problem

Examples

- Task stretching over downtime not allowed
- Task starting just before shift-change not allowed

An initial solution is found, but no further improvements are made

- Only for optimization problems
- Two basic scenarios
  - We are not finding better solutions due to limitations of search method
  - We have found the optimal solution, but have difficulty proving optimality
- Different approaches for both cases

Limit of Search Routine

- Some search methods work for finding good initial solutions, but are poor exploring the search space
- Other methods are weak getting initial good solutions, but are effective to explore search space
- Reasons
  - Bad update of cost function
  - Making choices that are dominated
  - Causes deep backtracking in search tree
- Adding lower bound is not helping
- Solutions:
  - Adding constraints to update cost earlier in tree
  - Explore search tree only partially

Running Example: Scheduling Problem

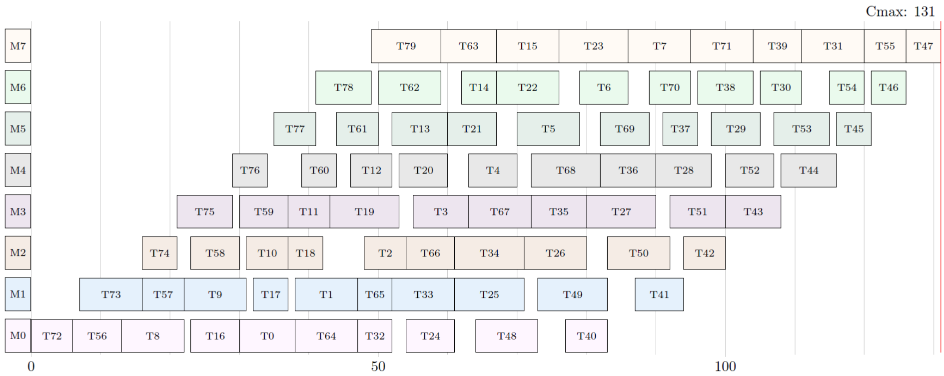
- Flowshop type problem
- Jobs consist of series of tasks processed on machines in same order
- Precedence between tasks of same job
- Disjunctive machines
- Tasks require operator during first part of run
    - Overall cumulative manpower limit

Conceptual Model (MiniZinc)

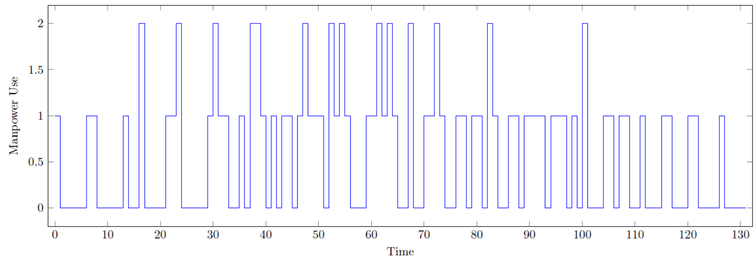
```
% variables
  array[T] of var 0..ub:start;
  var lb..ub:cmax;

% constraints
  constraint forall (i in T)
             (cmax >= start[i]+duration[i]);
  constraint forall (p in P)
             (start[prec[p,2]] >= start[prec[p,1]]+
                                  duration[prec[p,1]]);
  constraint forall(s in S)
      (cumulative([start[i]|i in T where stage[i]=s],
                  [duration[i]|i in T where stage[i] = s],
                  [1|i in T where stage[i] = s],1));
  constraint
      cumulative([start[i]|i in T],
                 [manpowerDuration[i]|i in T],
                 [1|i in T],manpower);

solve minimize cmax;
```

- Data definition not shown for brevity

Example Solution (10 Jobs/8 Machines)

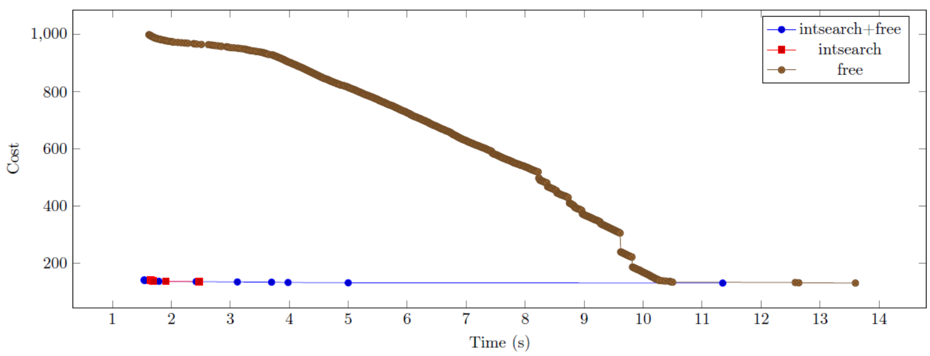


- Typical visualization: Gantt Chart
- Machine view
- Other views available



- Typical visualization: Resource Profile
- Show operator use over time

Comparison of Three Search Methods

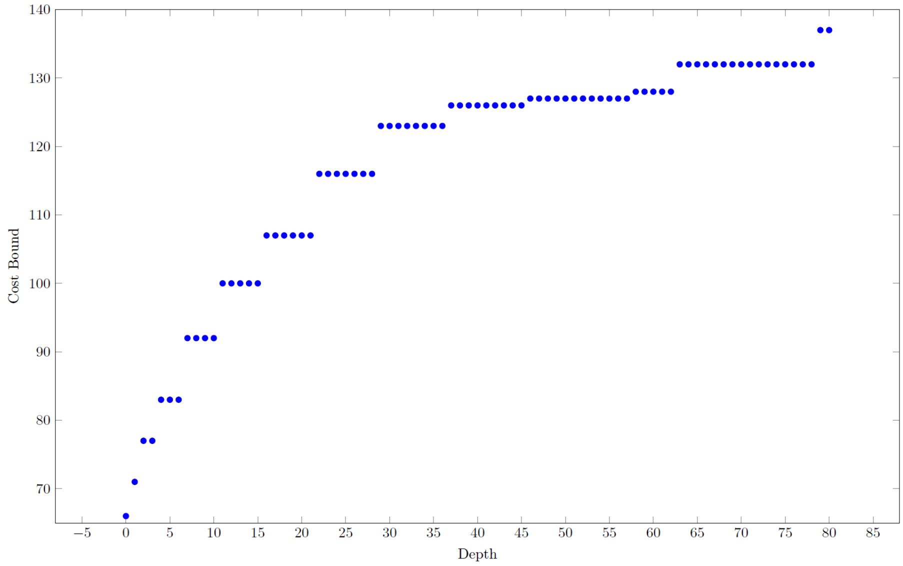


- *intsearch* finds goods solution initially, but gets stuck improving further
- *free* search starts with very poor cost, but continues to improve, reaches and proves optimum
- free search with ::intsearch annotation starts with good solution, but also finds and proves optimum (alternates search steps between free and intsearch)

Cost Estimate Visualization

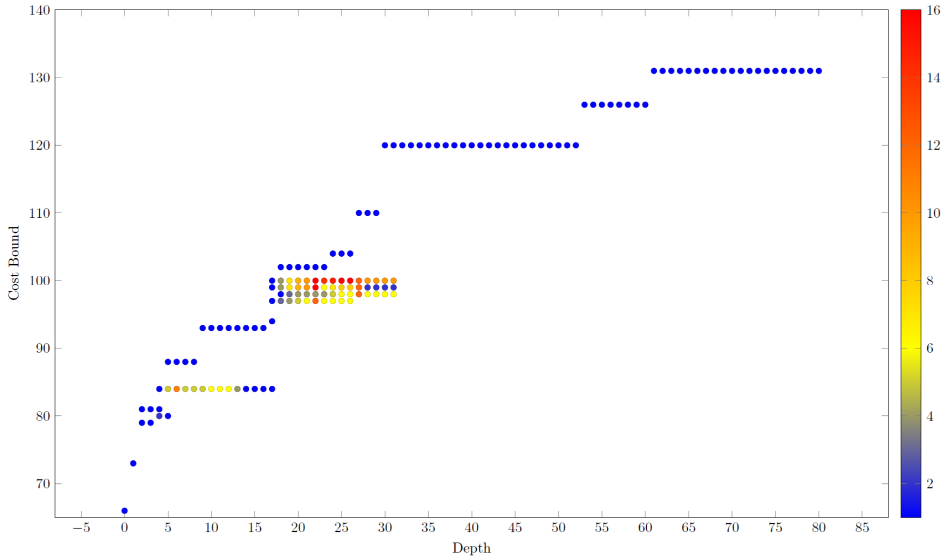
- See the lower bound of cost function over the depth of search
- On the left, initial bound due to propagation
- On the right, actual cost found
- How quickly does the lower bound rise?

Cost Estimates for Finding First Solution (SICStus)



- Initial bound of 66 due to longest job
- No backtracking for finding first solution

Finding Optimal Solution (Upper Bound 132)



- Color indicates number of nodes explored at this level
- Backtracking mainly between depth 20 and 30

Difficult Proof of Optimality

- Do we really care, if we have the optimal solution?
- We can use lower bound to stop search without further exploration
- Requires really good lower bound, or smallish problem size
- Often enough if we are "close enough" to lower bound

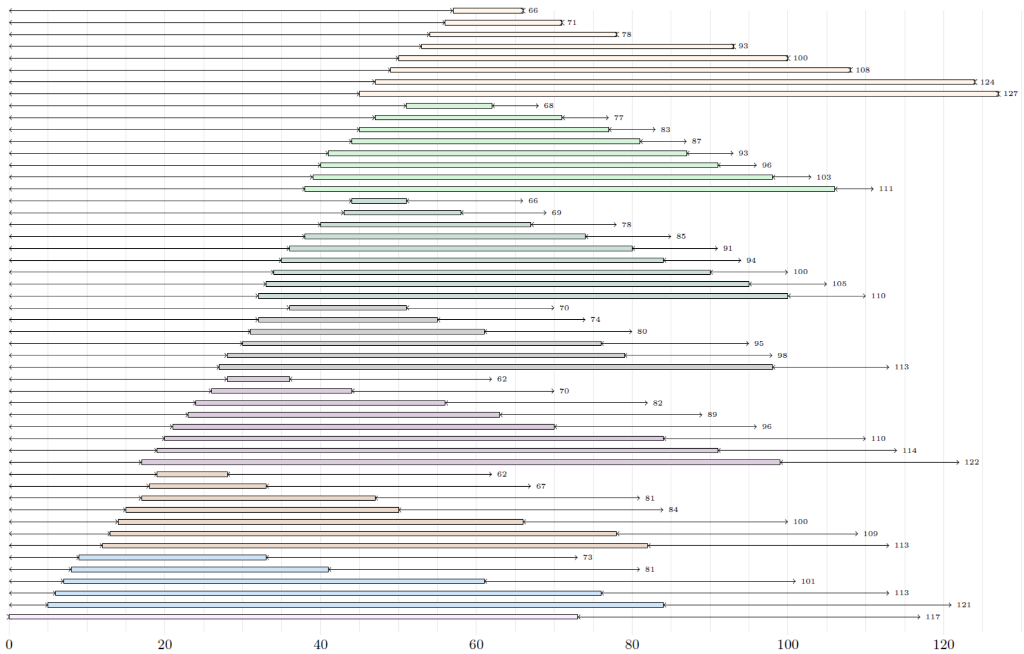
Understanding Lower Bounds

- The lower bound on the objective obtained by propagation often is very weak
- Many specialized, problem specific bounds in literature
- Used to stop search if lower bound is reached (no search for optimality proof required)
- Interest of converting lower bounds into constraints
- Example: Flowshop subset bound

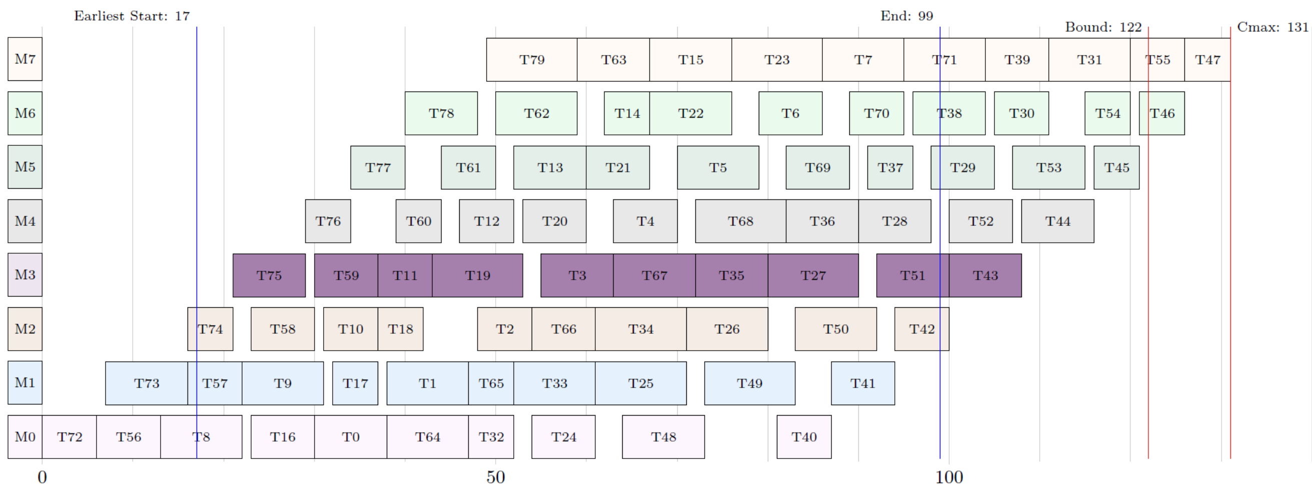
Flow Shop Stage/Subset Bound

- Initial bound on cost of flowshop is sum of durations in longest job
  - This works well if there are few jobs
  - It is meaningless if there are many jobs
- Stage bound
  - Consider time to process one stage
  - Waiting time before first task of stage begins operating
  - Workload of all tasks of that stage
  - Time to finish later stages after last task of this stage ends
  - Bound is sum of these three elements
- Subset bound
  - Also works for any subset of tasks of the same stage

Subset Bounds for Example Problem



Lower Bound and Actual Solution (Stage 3)

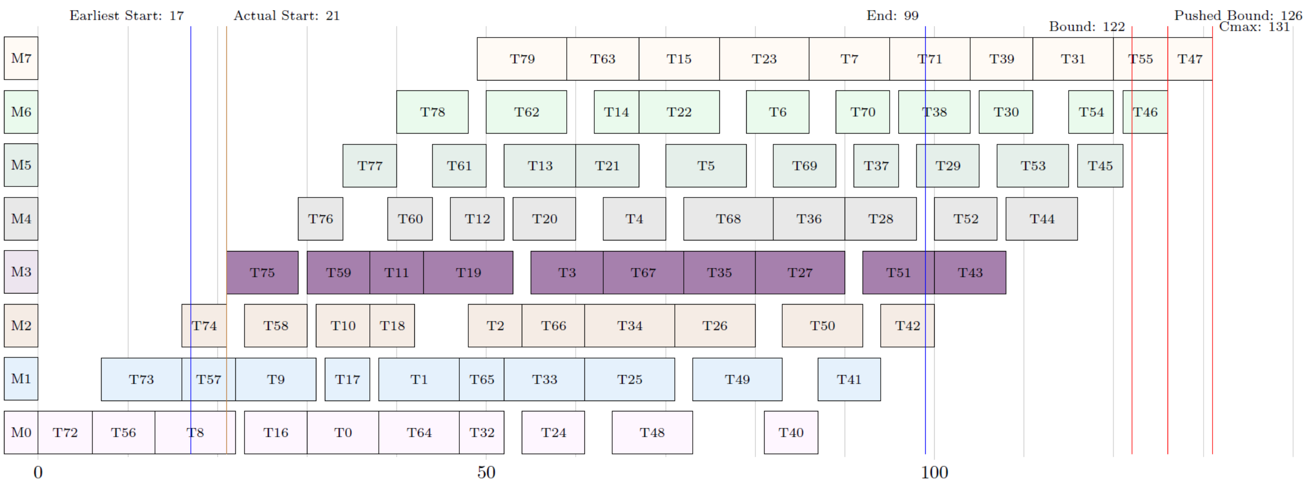


- Stage 3 does not begin at earliest start time, also has gaps in utilization

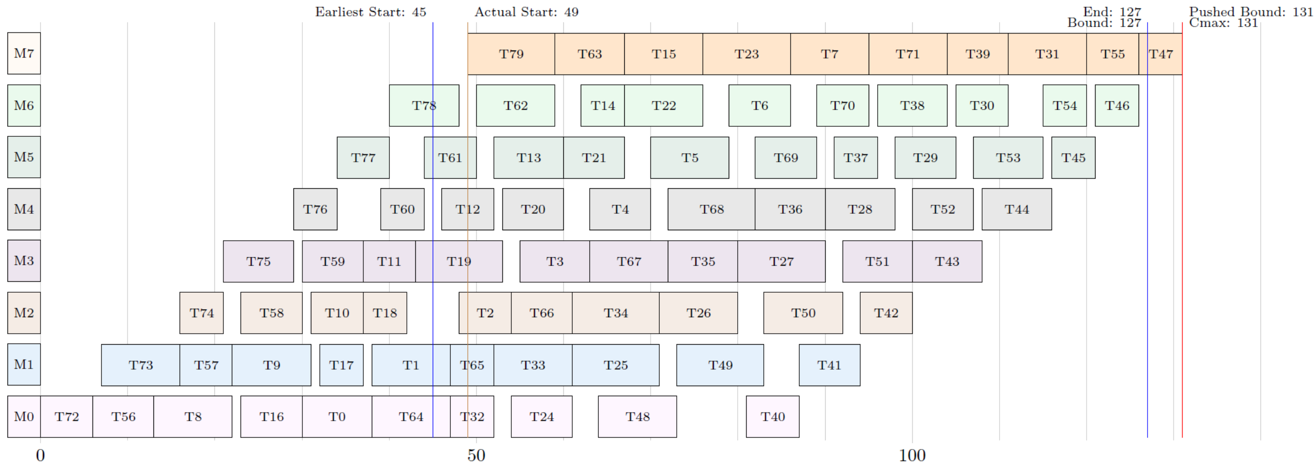
Converting Bound to Constraint

- Bound assumes that tasks start at earliest possible time
- We can see in solution that this is not the case
- At start of first task of this stage, the workload and the time to end are still correct
- Replace earliest start possible with minimum start time of the tasks
- We know that the makespan must be greater than this
- This is a constraint we can add to model
- (We can keep track of this as an invariant)

Pushing the Bound Example (Stage 3)

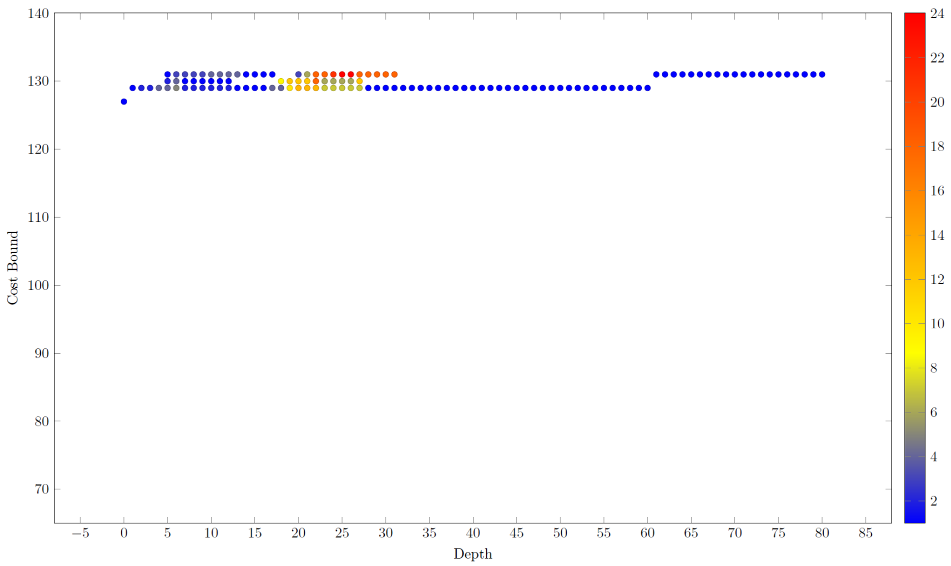


Strongest Constraint for Stage 7



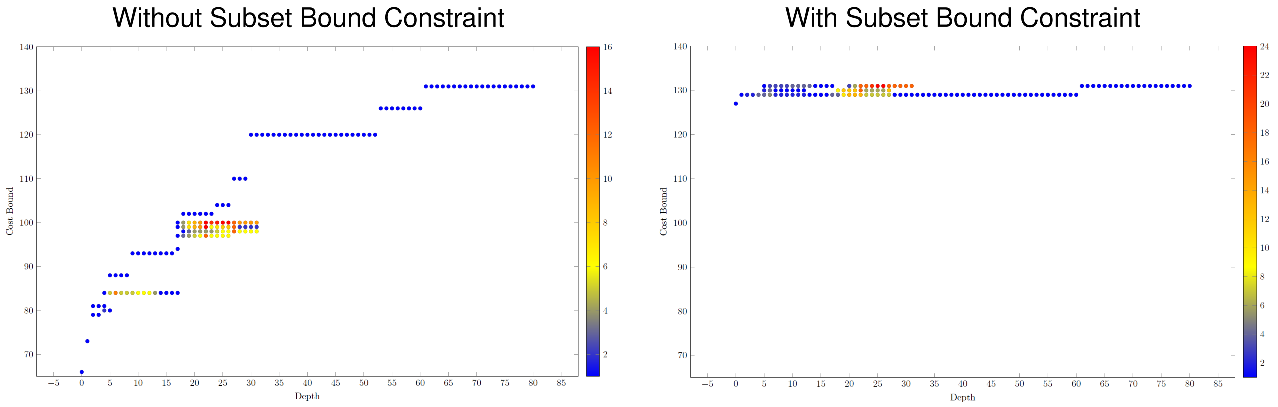
- When assigning T79 as first task of stage 7, we know that cost must be at least 131
- To improve, we must start stage 7 before time 49

Updated Cost Estimate in Search for Optimal Solution



- Cost bound jumped from lower bound 127 to 129 after first task fixed

Comparison of Cost Estimates



- Does not reduce amount of search as much as expected
- This type of view is also useful to compare heuristic variable orderings

Research Question

- How do we plot search depth in a clause learning solver?
- Actual depth of search tree varies
- Variables in search not directly linked to user defined variables

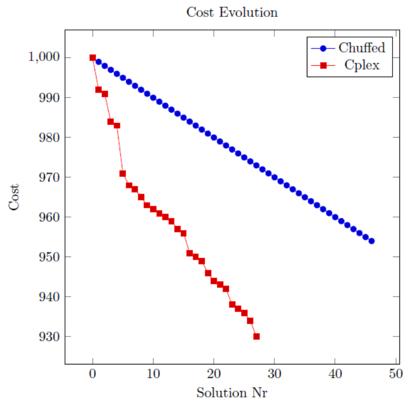
The model has been working for some time, but suddenly it stops working, or violates a constraint

- Often due to changes in base data (resources/process)
- Data entry mistake (duration too long, resource use too high)
- Missing data (null values replaced by defaults)
- Missing preferences (no default process path given for new product)
- Change in real world not reflected in input data
  - New resources
  - Upgraded machines
  - Process changes

The user can easily improve upon the best solution found

- This often happens with vehicle routing problems
- Unable to explore the search space completely
- Visualization gives an immediate picture of solution quality
- Manual changes correspond to local search moves
- One solution is to do a local search post-solving
- Careful: sometimes it only looks like an improvement
  - Especially if distance is based on actual road travel time
  - ...and visualization uses straight line links
- Having a nice visualization can become a constraint

The solver says it is optimal, but there is a better solution



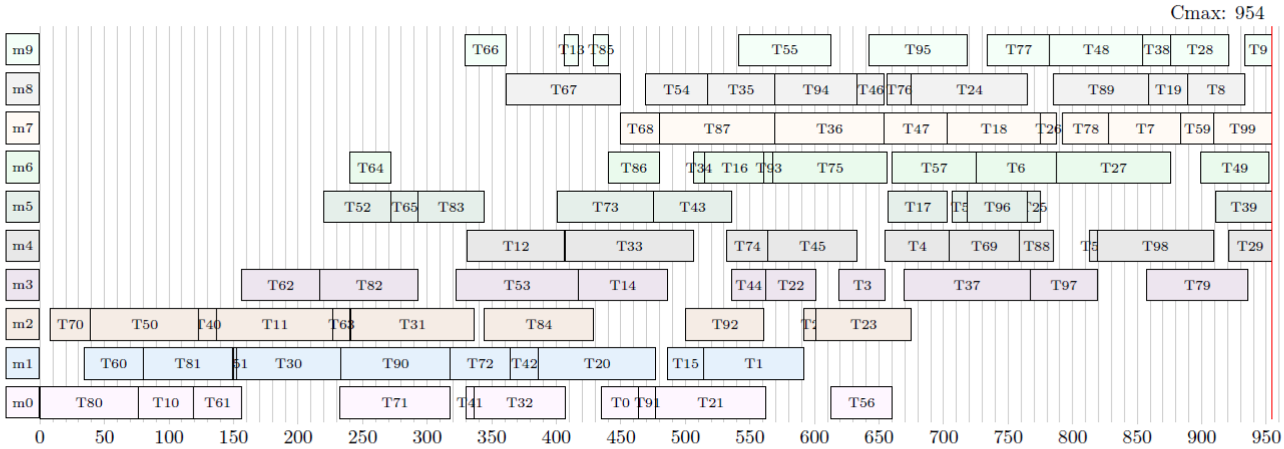
- Example MT10 with Chuffed
  - Classical scheduling benchmark [Carlier, Pinson 1989]
  - We know that optimal solution is 930
  - Compare Cplex/Chuffed solutions
  - Chuffed claims its best solution is optimal
  - But Cplex finds a better optimal solution
- This is really tough to detect for a new problem
  - Solution satisfies the constraints
  - Requires a priori knowledge of optimal cost
  - Or, two solvers that both find an optimal solution, and disagree on cost
  - (Optimal solutions with same cost can be quite different)

Job-Shop Scheduling (MiniZinc Program)

```
include "globals.mzn";
int:nrJobs;
int:nrRes;
set of int: J=1..nrJobs;
set of int: R=1..nrRes;
array[J,R] of int:taskUse;
array[J,R] of int:taskDuration;
include "mt10.dzn";
int:ub =1000;

array[J,R] of var 0..ub:start;
var 0..ub:objective;
constraint forall(j in J)
   (objective >= start[j,nrRes]+taskDuration[j,nrRes]);
 constraint forall(j in J, r in 1..nrRes-1)
  (start[j,r+1] >= start[j,r]+taskDuration[j,r]);
  constraint forall (r in R)
    (cumulative([start[j,k]|j in J, k in R where taskUse[j,k]+1=r],
                              [taskDuration[j,k]|j in J, k in R where taskUse[j,k]+1=r],
                              [1|j in J, k in R where taskUse[j,k]+1=r],1)
    );

solve minimize objective;
```

"Optimal" Solution found with Chuffed (Cost 954)

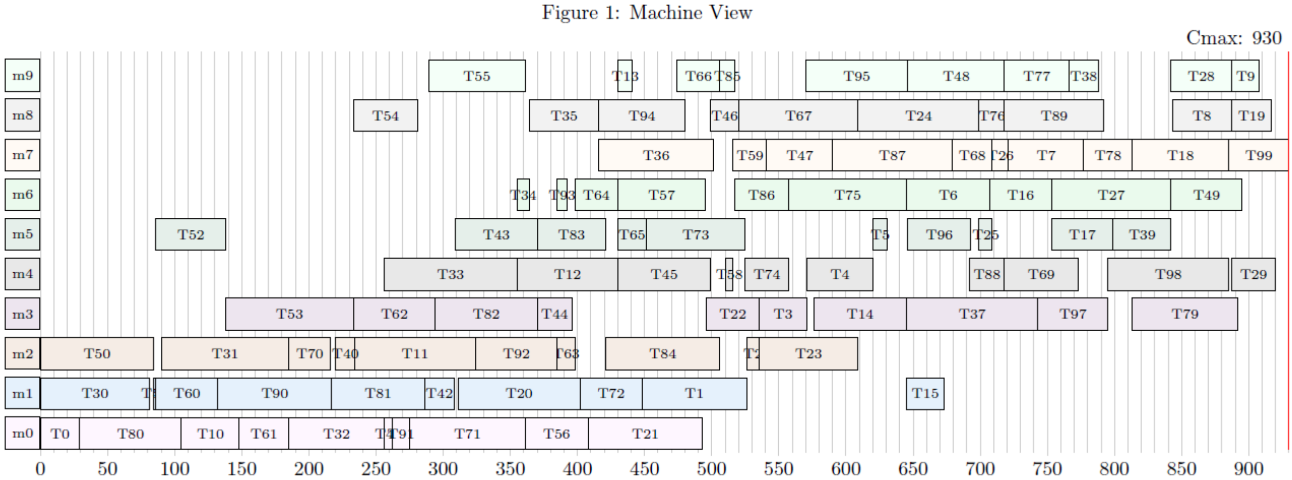
Figure 5: Machine View (Chuffed Solution)



- No reason to suspect that something is wrong
- Different upper bounds lead to different optimal values?

Real Optimum found with Cplex (Cost 930)

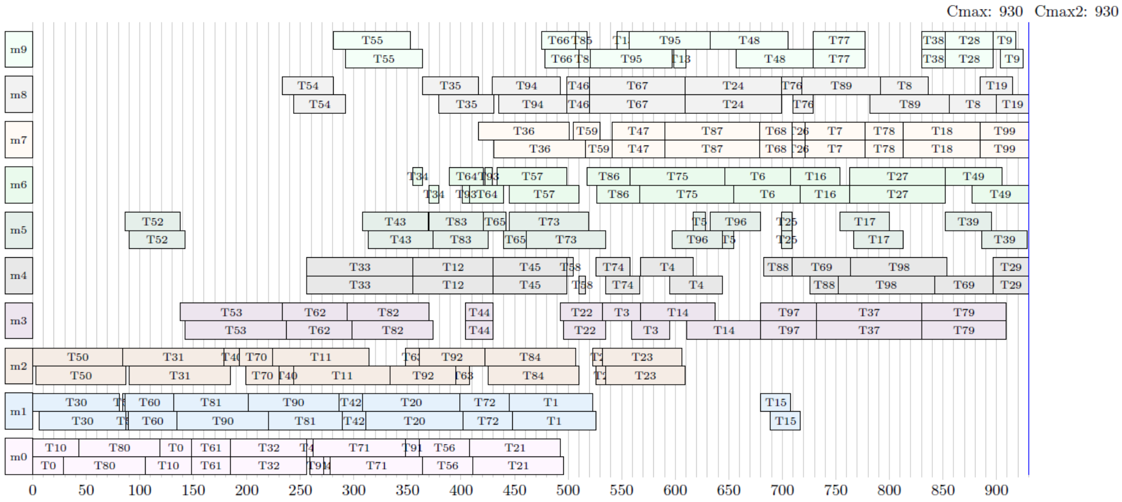
Figure 1: Machine View



- Normally, we would suspect the user program
- Identical MiniZinc program and data, only backend solver changed

Comparing Two Optimal Solutions (Chuffed/Cplex)

Figure 7: Comparison Machine View

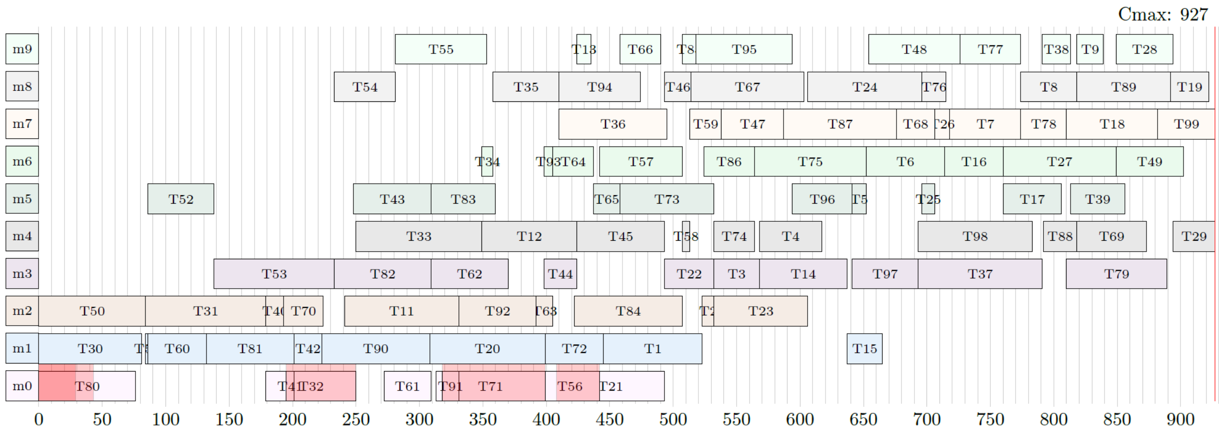


- Many tasks have different start times in the two solutions
- Order of some jobs on machines changed

The solver says it has found a solution better than the known optimum

- Known benchmark results are not just useful to check performance
- Lower bounds can be used if optimal value not known
  - If you find solution better than lower bound, then either solution or bound (or both) are wrong
  - Only works if lower bound not used to prune search
- Double check before announcing sensational result

Improved Solution for MT10?



- Highlights overlap of tasks on Machine m0
- Caused by mismatch of resource ids in data (0-9) and in program (1-10)
- No resource constraint for Machine m0

MiniZinc Program with Problem

```
include "globals.mzn";

int:nrJobs;
int:nrRes;

set of int: J=1..nrJobs;
set of int: R=1..nrRes;

array[J,R] of int:taskUse;
array[J,R] of int:taskDuration;
int:ub =1000;

array[J,R] of var 0..ub:start;
var 0..ub:objective;

constraint forall(j in J)
   (objective >= start[j,nrRes]+taskDuration[j,nrRes]);
 constraint forall(j in J, r in 1..nrRes-1)
  (start[j,r+1] >= start[j,r]+taskDuration[j,r]);
  constraint forall (r in R)
    (cumulative([start[j,k]|j in J, k in R where taskUse[j,k]=r],
                            [taskDuration[j,k]|j in J, k in R where taskUse[j,k]=r],
                            [1|j in J, k in R where taskUse[j,k]=r],1)

solve minimize objective;
```

User Rejects Valid Solution

- Hardest cases, as the model is working as intended
- You may have the wrong objective
- You may only be exploring a small part of the search space
- The user is wrong

Counterfactual Explanations

- The argument by the user is:
  - Instead of your current solution you should be doing *this and that*
  - These changes would produce a better solution
- Step 1: Can you really do *this and that*? Check if there are solutions which do these changes.
- Step 2: Does this really make an improvement with your current objective?
- Step 3: Can you modify the objective function to accept this as a better solution?
- Step 4: Can you modify the search so that you find this modified solution?
- Step 5: Can you modify the model so that this becomes a feasible solution?

Counterfactual Explanations in AI

- Hot topic in "Explainable AI" [Stepin et al., 2021, Verma et al., 2020]
- First papers applying this to CP [Korikov and Beck, 2021] [Korikov et al., 2021]
  - Restricted to changing the weights of a cost function, not the constraints themselves
- Still a question of "What is a satisfactory explanation?"
  - Differentiate model-centric and user-centric view
  - A small change for the user may be a big change in the model and vice versa

Show Flexibility in Solution

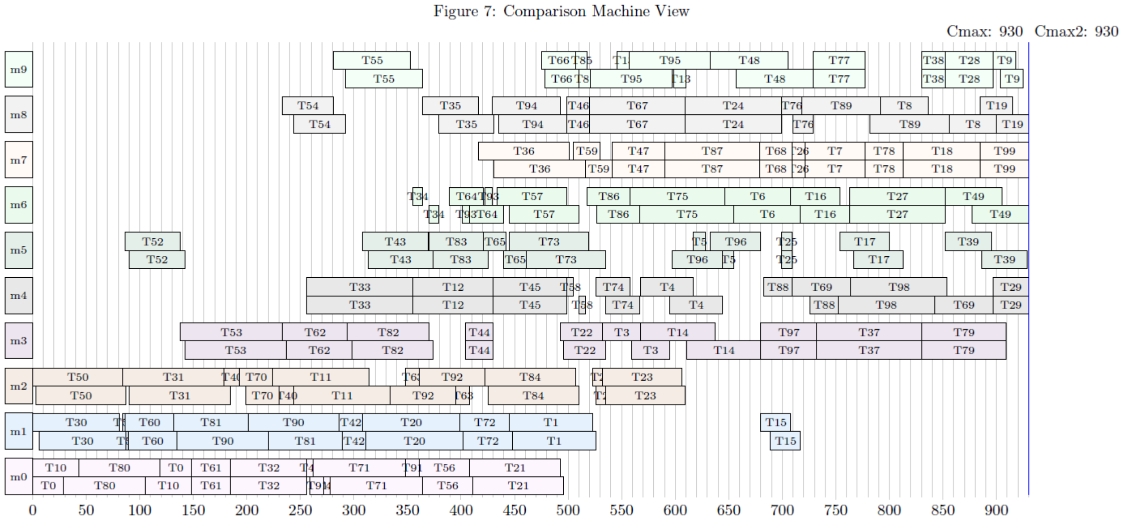
- Instead of showing single solution, show what is possible
- Compare two solutions on one visualization
- In a given solution, some variables can be assigned differently without affecting other variables
  - Easy to allow user to shift tasks within that range
- More flexibility by replacing resource constraints by inequalities
  - Keeps relative order of tasks
  - Can be pre-computed in polynomial time (interactive use may be possible)
- To find overall earliest start/latest end requires multiple solver runs (expensive, not interactive)

Job-Shop Scheduling (MiniZinc Program)

```
include "globals.mzn";
int:nrJobs;
int:nrRes;
set of int: J=1..nrJobs;
set of int: R=1..nrRes;
array[J,R] of int:taskUse;
array[J,R] of int:taskDuration;
include "mt10.dzn";
int:ub =1000;

array[J,R] of var 0..ub:start;
var 0..ub:objective;
constraint forall(j in J)
   (objective >= start[j,nrRes]+taskDuration[j,nrRes]);
 constraint forall(j in J, r in 1..nrRes-1)
  (start[j,r+1] >= start[j,r]+taskDuration[j,r]);
  constraint forall (r in R)
    (cumulative([start[j,k]|j in J, k in R where taskUse[j,k]+1=r],
                               [taskDuration[j,k]|j in J, k in R where taskUse[j,k]+1=r],
                               [1|j in J, k in R where taskUse[j,k]+1=r],1)
    );

solve minimize objective;
```

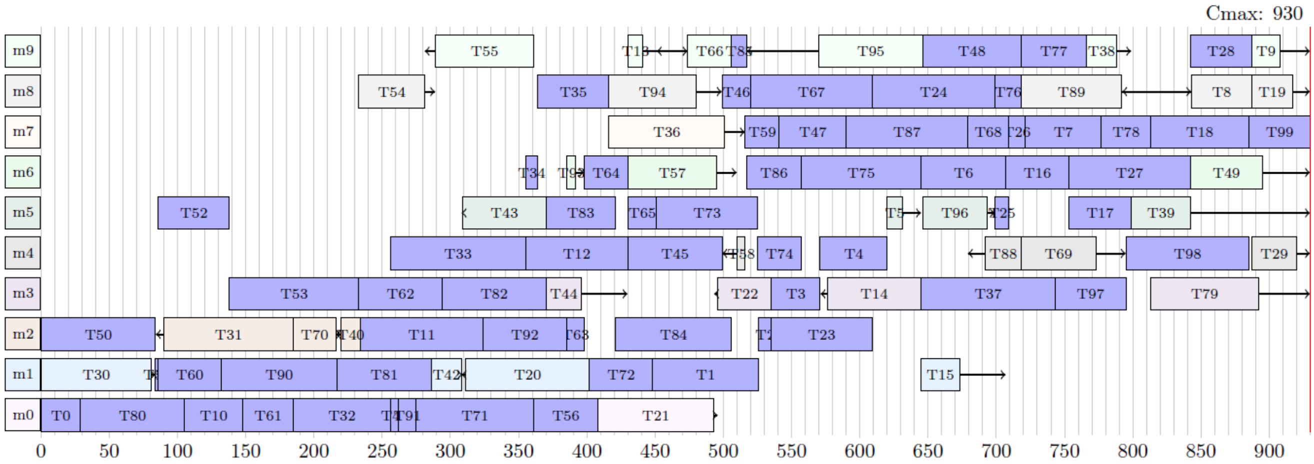

Comparing Two Optimal Solutions (Chuffed/Cplex)

Figure 7: Comparison Machine View



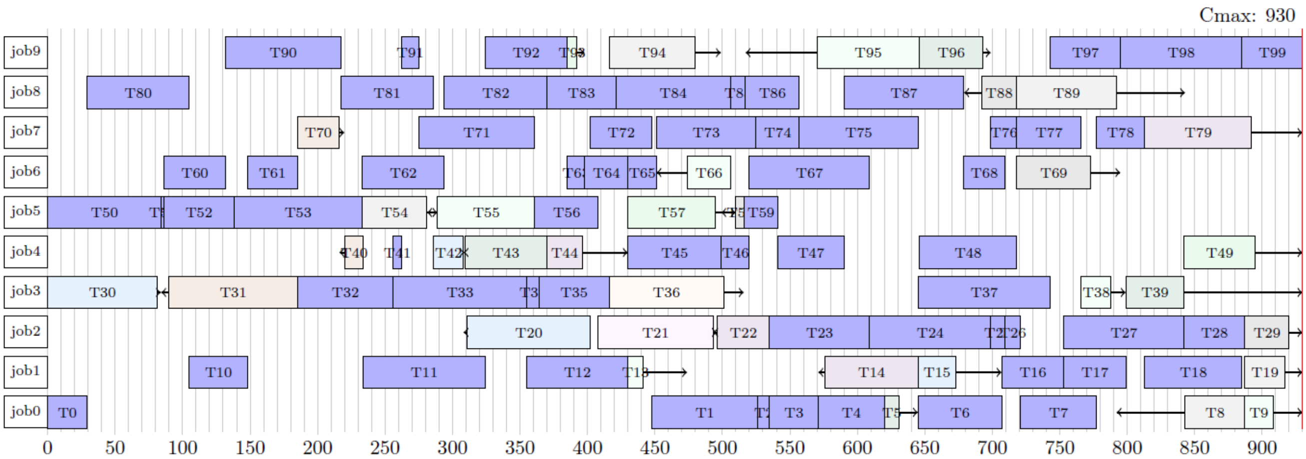
- Many tasks have different start times in the two solutions
- Order of some jobs on machines changed

Single Task Flexibility (Machine View)



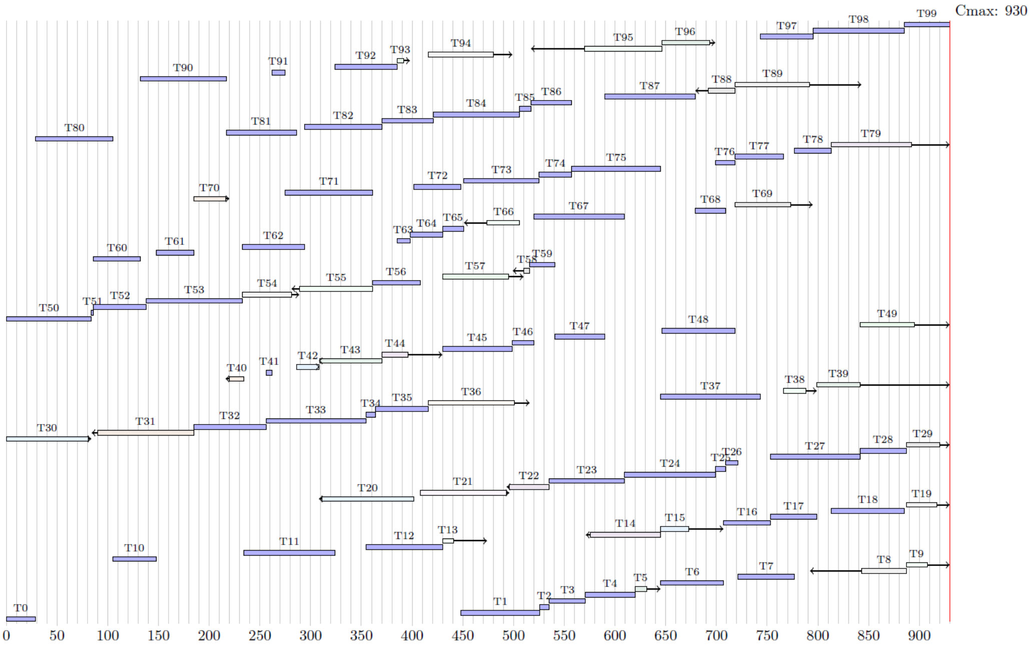
- Show how much a task can move without changing any other task
- Blue tasks are not movable in current solution
- More complex when also maintaining other constraints

Single Task Flexibility (Job View)



- Some tasks could move further by pushing other tasks (indicated in red)
- This requires running a constraint program to find bounds
- Keep task order on machines, keep objective value

Single Task Flexibility (Task View)



- Bounds best viewed in task view

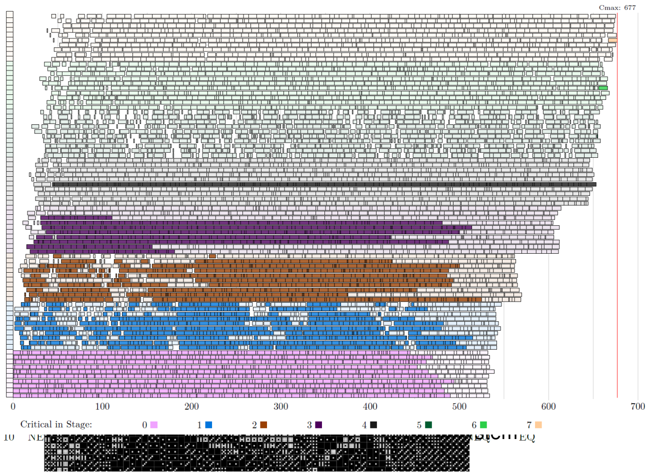
What have we discussed?

an

- How to use visualization in modelling problems
- How this integrates into development process
- What can be done with different tools
- Overview of literature in this area

What did we leave out?

- Visualizations to understand and improve a solver
- Tools for Constraint Acquisition
- Visualization for Local Search and Constraint Based Local Search [Dooms et al., 2009]
- How to test effectiveness of visualization
- Visualizations for specific applications
- Visualization as opt-art [Bosch, 2006, Cambazard et al., 2011]
- Scalability issues



Are the Examples Available?

- Much is available in the MiniZinc IDE
- Assistant based material not yet released
- Will become available as open-source
- Slide set available at

References I

Bosch, R. (2006).
Opt art.
In Beck, J. C. and Smith, B. M., editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Third International Conference, CPAIOR 2006, Cork, Ireland, May 31 - June 2, 2006, Proceedings*, volume 3990 of *Lecture Notes in Computer Science*, page 1. Springer.

Cambazard, H., Horan, J., O'Mahony, E., and O'Sullivan, B. (2011).
Domino portrait generation: a fast and scalable approach.
Ann. Oper. Res., 184(1):79–95.

Dooms, G., Hentenryck, P. V., and Michel, L. (2009).
Model-driven visualizations of constraint-based local search.
Constraints An Int. J., 14(3):294–324.

References II

Downing, N., Feydy, T., and Stuckey, P. J. (2012).
Explaining alldifferent.
In Reynolds, M. and Thomas, B. H., editors, *Thirty-Fifth Australasian Computer Science Conference, ACSC 2012, Melbourne, Australia, January 2012*, volume 122 of *CRPIT*, pages 115–124. Australian Computer Society.

Fages, F., Soliman, S., and Coolen, R. (2004).
CLPGUI: A generic graphical user interface for constraint logic programming.
Constraints An Int. J., 9(4):241–262.

References III

Gervet, C. (1998).
Large scale combinatorial optimization: A methodological viewpoint.
In Freuder, E. C. and Wallace, R. J., editors, *Constraint Programming and Large Scale Discrete Optimization, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, September 14-17, 1998*, volume 57 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 151–175. DIMACS/AMS.

Gupta, S. D., Genc, B., and O'Sullivan, B. (2021).
Explanation in constraint satisfaction: A survey.
In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4400–4407. ijcai.org.

References IV

Howell, I., Choueiry, B. Y., and Yu, H. (2020).
Visualizations to summarize search behavior.
In Simonis, H., editor, *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings*, volume 12333 of *Lecture Notes in Computer Science*, pages 392–409. Springer.

Howell, I., Woodward, R. J., Choueiry, B. Y., and Bessiere, C. (2018).
Solving sudoku with consistency: A visual and interactive approach.
In Lang, J., editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 5829–5831. ijcai.org.

References V

Korikov, A. and Beck, C. (2021).
Counterfactual explanations via inverse constraint programming.
In *Proceedings of CP 2021*.

Korikov, A., Shleyfman, A., and Beck, J. C. (2021).
Counterfactual explanations for optimization-based decisions in the context of the GDPR.
In Zhou, Z., editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4097–4103. ijcai.org.

Saket, B., Endert, A., and Demiralp, Ç. (2019).
Task-based effectiveness of basic visualizations.
IEEE Trans. Vis. Comput. Graph., 25(7):2505–2512.

References VI

Schulte, C. (1997).
Oz explorer: A visual constraint programming tool.
In Naish, L., editor, *Logic Programming, Proceedings of the Fourteenth International Conference on Logic Programming, Leuven, Belgium, July 8-11, 1997*, pages 286–300. MIT Press.

Shishmarev, M., Mears, C., Tack, G., and de la Banda, M. G. (2016).
Visual search tree profiling.
Constraints An Int. J., 21(1):77–94.

References VII

Simonis, H. and Aggoun, A. (2000).
Search-tree visualisation.
In Deransart, P., Hermenegildo, M. V., and Maluszynski, J., editors, *Analysis and Visualization Tools for Constraint Programming, Constrain Debugging (DiSCiPl project)*, volume 1870 of *Lecture Notes in Computer Science*, pages 191–208. Springer.

Simonis, H., Aggoun, A., Beldiceanu, N., and Bourreau, E. (2000).
Complex constraint abstraction: Global constraint visualisation.
In Deransart, P., Hermenegildo, M. V., and Maluszynski, J., editors, *Analysis and Visualization Tools for Constraint Programming, Constrain Debugging (DiSCiPl project)*, volume 1870 of *Lecture Notes in Computer Science*, pages 299–317. Springer.

References VIII

Simonis, H., Davern, P., Feldman, J., Mehta, D., Quesada, L., and Carlsson, M. (2010).
A generic visualization platform for CP.
In Cohen, D., editor, *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, volume 6308 of *Lecture Notes in Computer Science*, pages 460–474. Springer.

Simonis, H. and O'Sullivan, B. (2011).
Almost square packing.
In Achterberg, T. and Beck, J. C., editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems - 8th International Conference, CPAIOR 2011, Berlin, Germany, May 23-27, 2011.*

References IX

Proceedings, volume 6697 of *Lecture Notes in Computer Science*, pages 196–209. Springer.

Stepin, I., Alonso, J. M., Catalá, A., and Pereira-Fariña, M. (2021).
A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence.
IEEE Access, 9:11974–12001.

Verma, S., Dickerson, J. P., and Hines, K. (2020).
Counterfactual explanations for machine learning: A review.
CoRR, abs/2010.10596.