Improving Constraint Modelling Using Visualization

Helmut Simonis

Improving Constraint Modelling Using Visualization

Helmut Simonis

Cork Constraint Computation Centre
Computer Science Department
University College Cork
Ireland

JFPC 2010

Overview

- Visualization can play major role in CP modelling
- This is not new, but neglected
- We can do this more systematically
- Visualization is not for tool developers
- Not enough to develop tools, we need advice how to use and interpret them
- There are other issues we rarely address in CP
  - How do I pick best model without implementing all alternatives?
  - Is there such a thing as the best model?
  - Questions, examples, but no answers here!

Methodology

- Based on 3 case studies
  - Almost square packing
  - Rooms problem, sports scheduling
  - Tight production scheduling system
- Each solves a non-trivial problem
- Each shows a different aspect of the problem

Example 1

- Canonical model: Beldiceanu & Contejean, 94
- Naive search impractical
- Static variable selection, from largest to smallest item
- No value choice preference
- Complex search, phased (interleaved) assignment
- Very large search space explored

Example 1: Almost Square Packing

- Consider almost squares, rectangles $n \times (n+1)$
- Pack all items 1×2, 2×3, ..., $n \times (n+1)$
- Into smallest possible rectangle
- Items are non-overlapping

Background

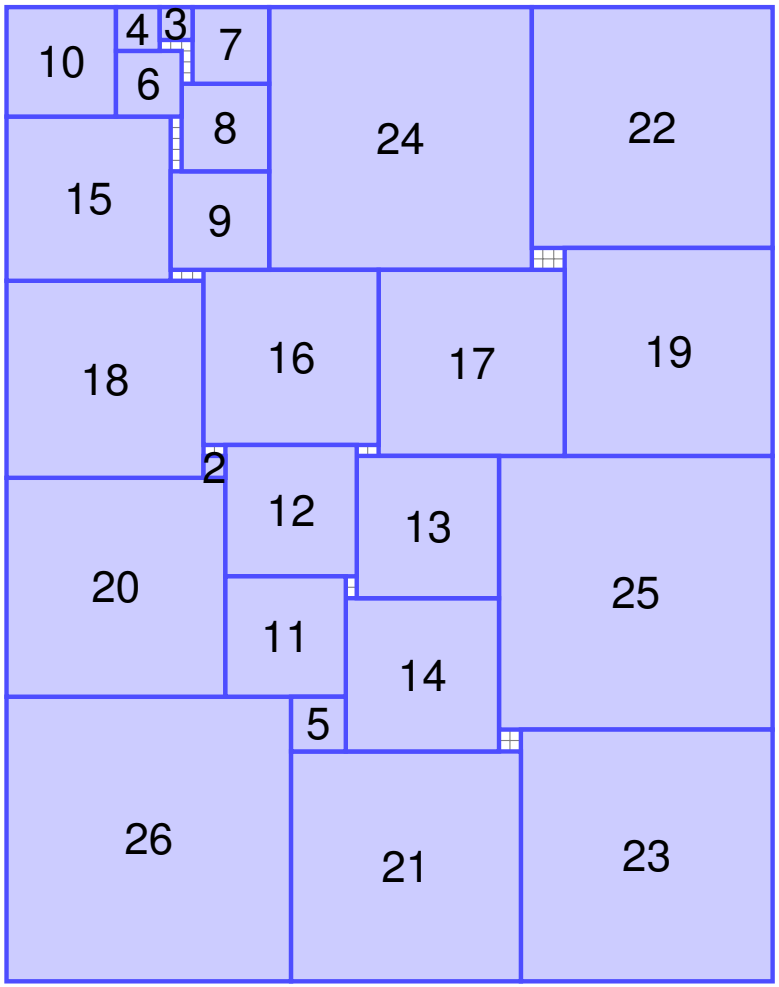
- Suggested by Prof. MacHale, Math, UCC
- Solved packing up to 13x14 by hand
- Starting point for Barry's and my interest in packing
- My objection: "This is too complicated"
  - Look at square packing
  - Known results, benchmark comparison

Simpler Problem: Square Packing

- Consider squares, rectangles $n \times n$
- Pack all items 1×1, 2×2, ..., $n \times n$
- Into smallest possible rectangle
- Items are non-overlapping

Outline

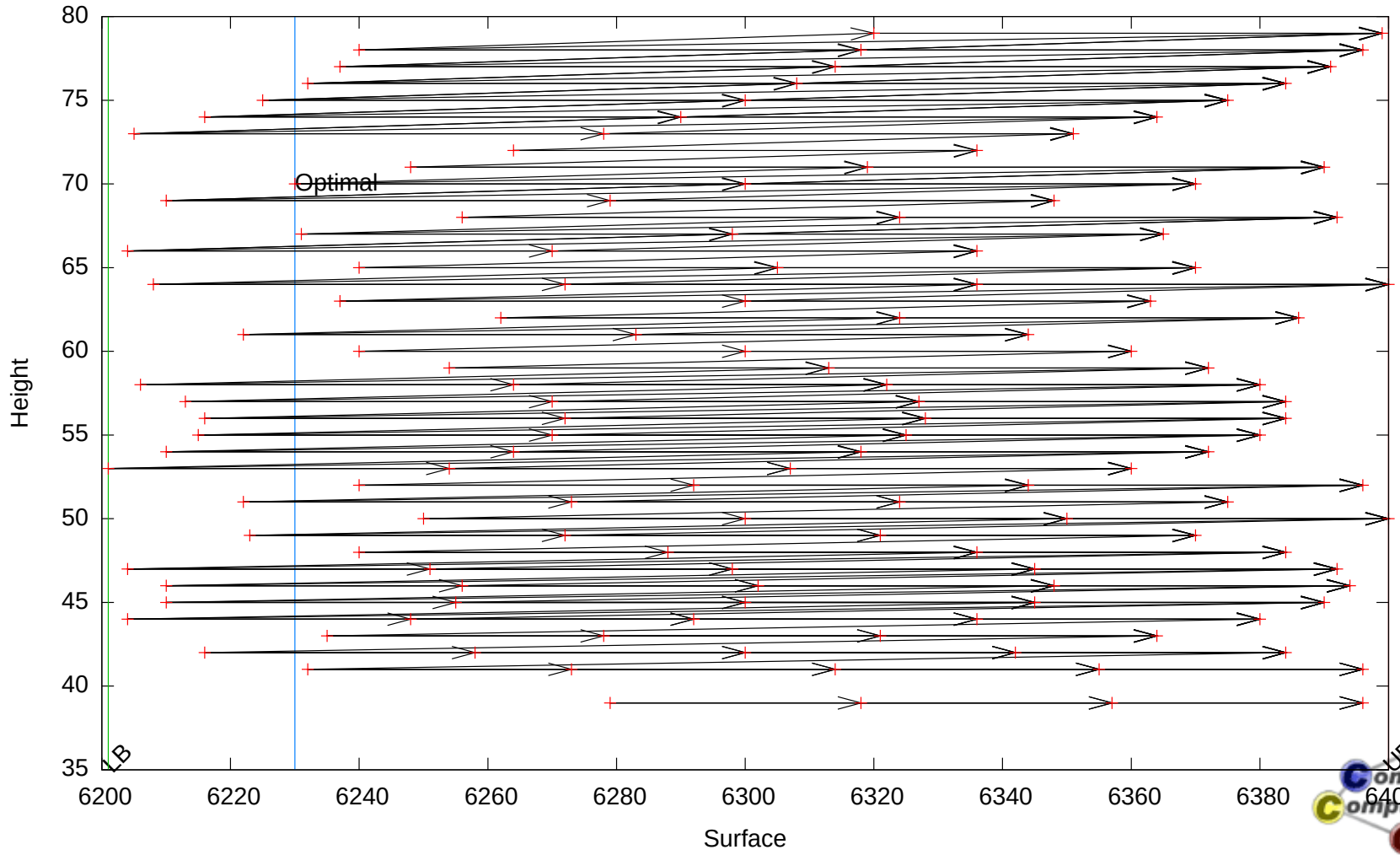
Problem (N=26)



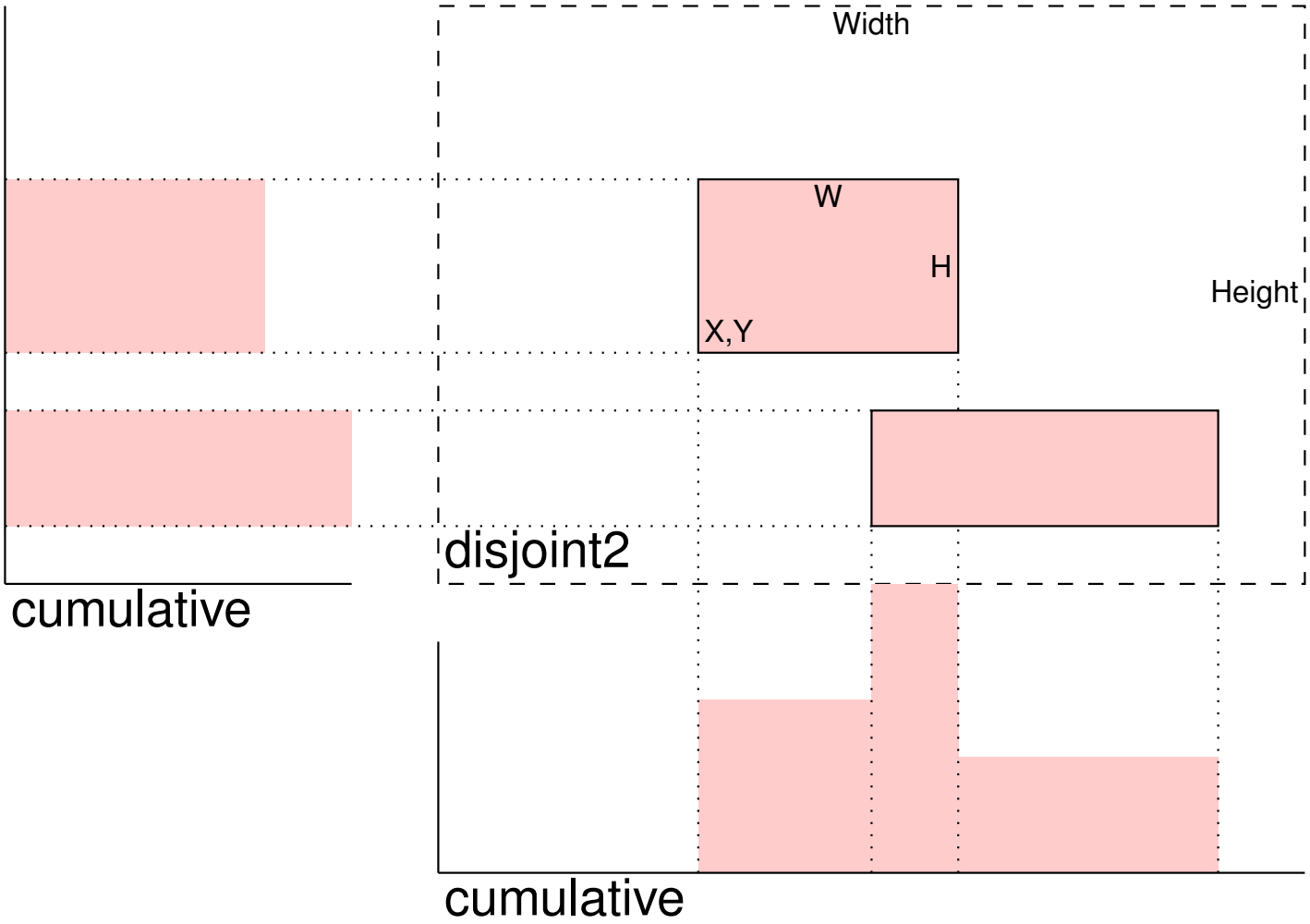
Problem Decomposition

- Search for candidate enclosing rectangle
- Area must be larger than sum of items to be placed
- Search in order of increasing area
  - and increasing “squareness”
- Check each candidate for (in)feasibility until first solution is found
- Observation: Only limited number of candidates explored

Candidates


Optimal
LB
UB
Height
Surface
80
75
70
65
60
55
50
45
40
35
6200
6220
6240
6260
6280
6300
6320
6340
6360
6380
6400

Basic Model



Outline

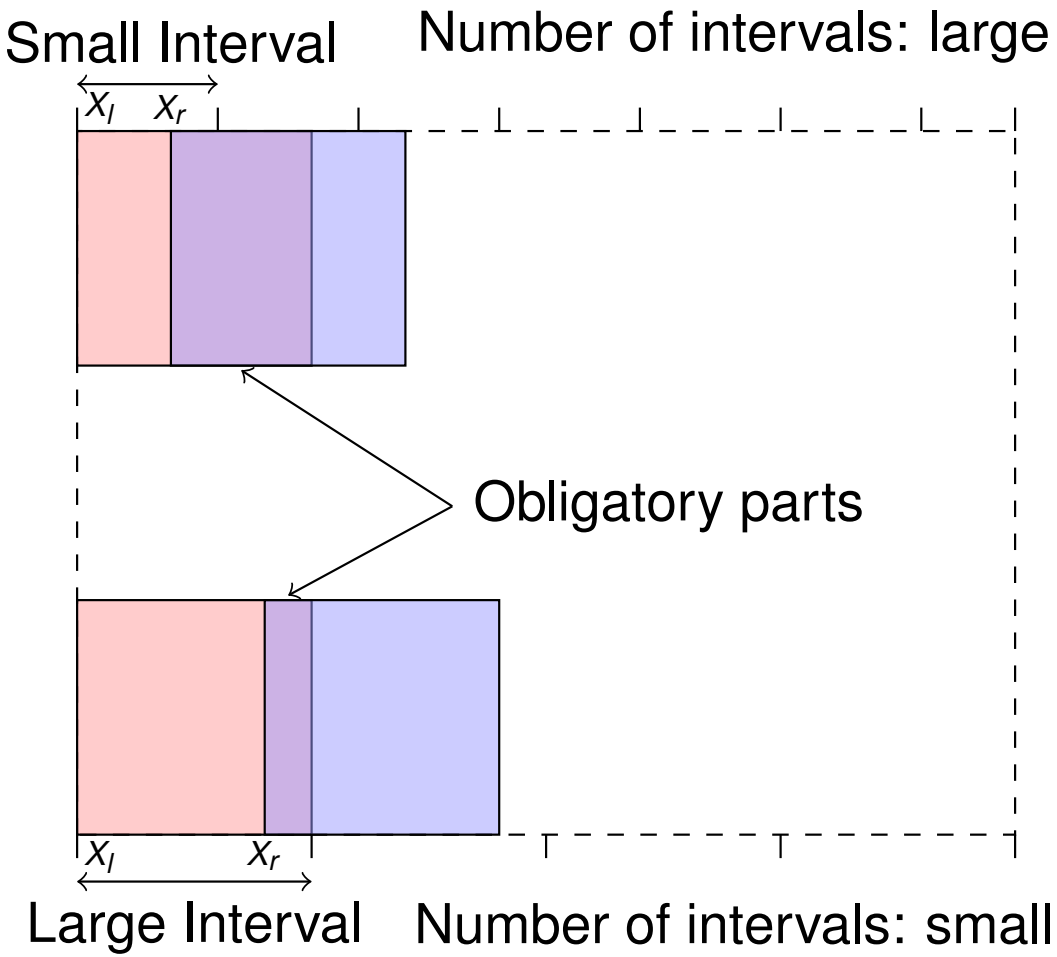
Alternatives

- naive
- x then y
- disjunctive
- semantic disjunctive
- dual
- interval
- split
- xy interval

Interval Based Strategies

- Key Idea: Fixing intervals, not values
- Fixing variables to values is too restrictive
- Select “area” in which item is placed
- Allows items to shift slightly
- Restrict domain to intervals
- Only at end fix actual values

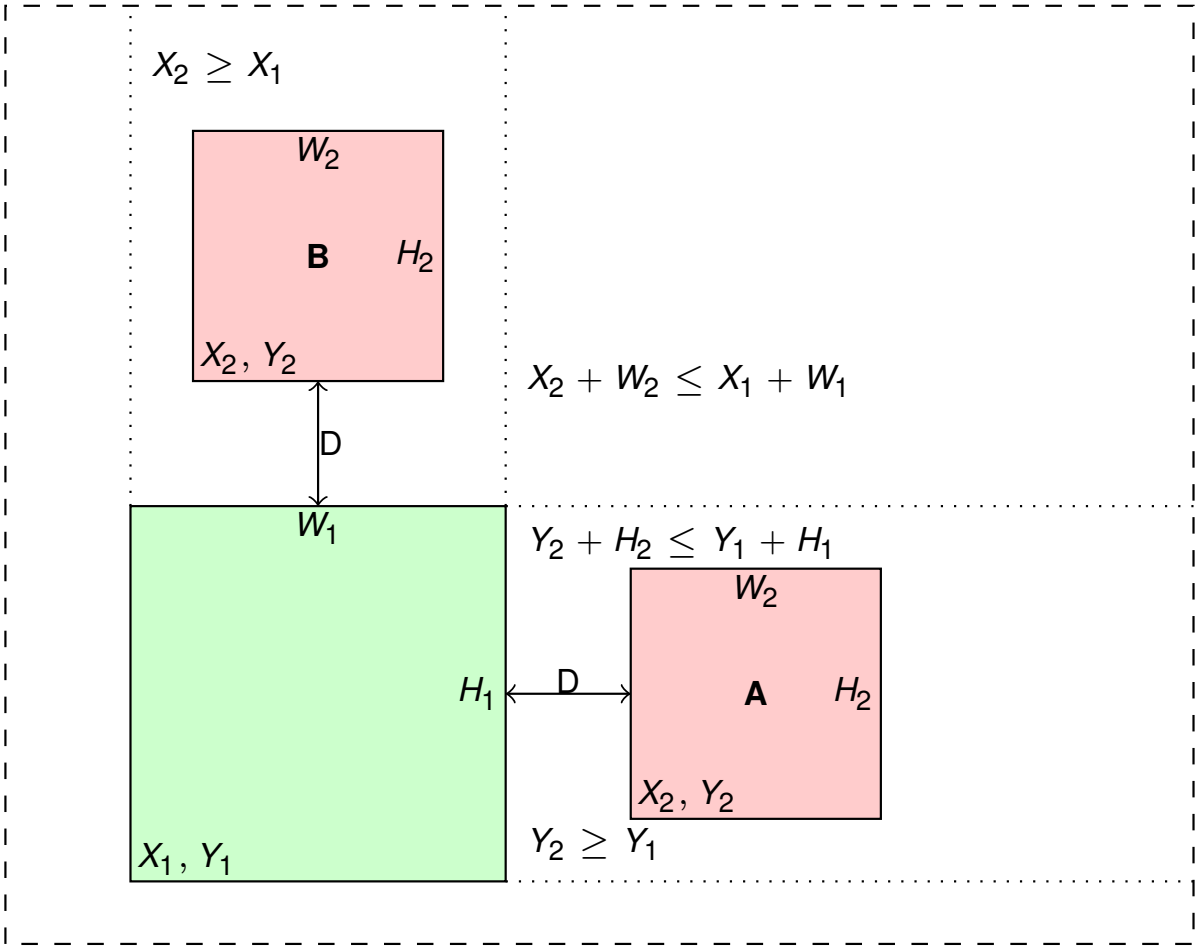
Forcing Obligatory Parts



Variants

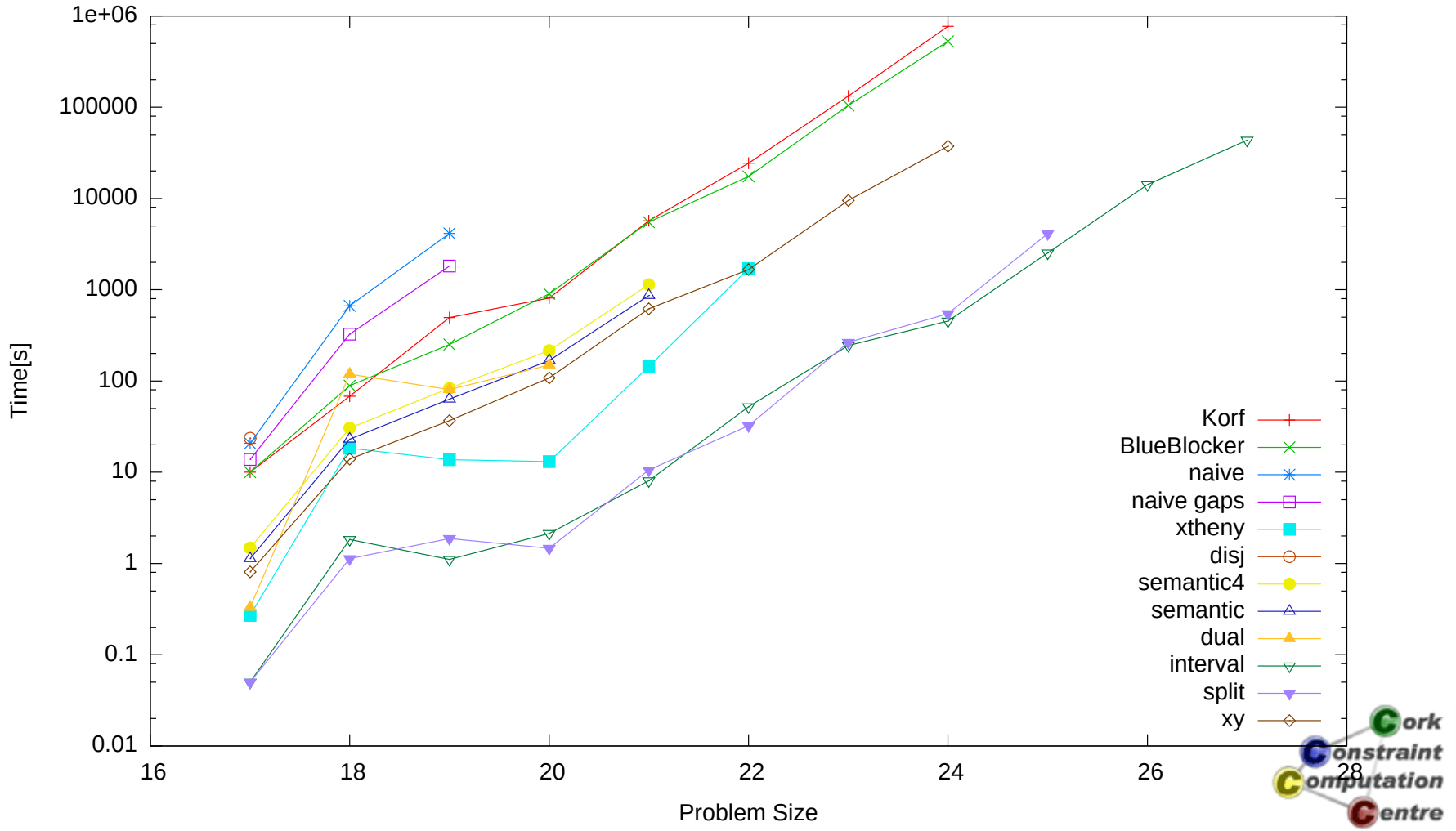
- (X) Interval
  - Split all X variables into intervals
  - Then fix X values
  - Then treat all Y variables the same way
- Split
  - Split X variables into intervals
  - Split Y variables into intervals
  - Then fix values
- XY Interval
  - For each item, split X and Y variables into intervals
  - Then fix values

Model Improvement: Dominance Criterion



Outline

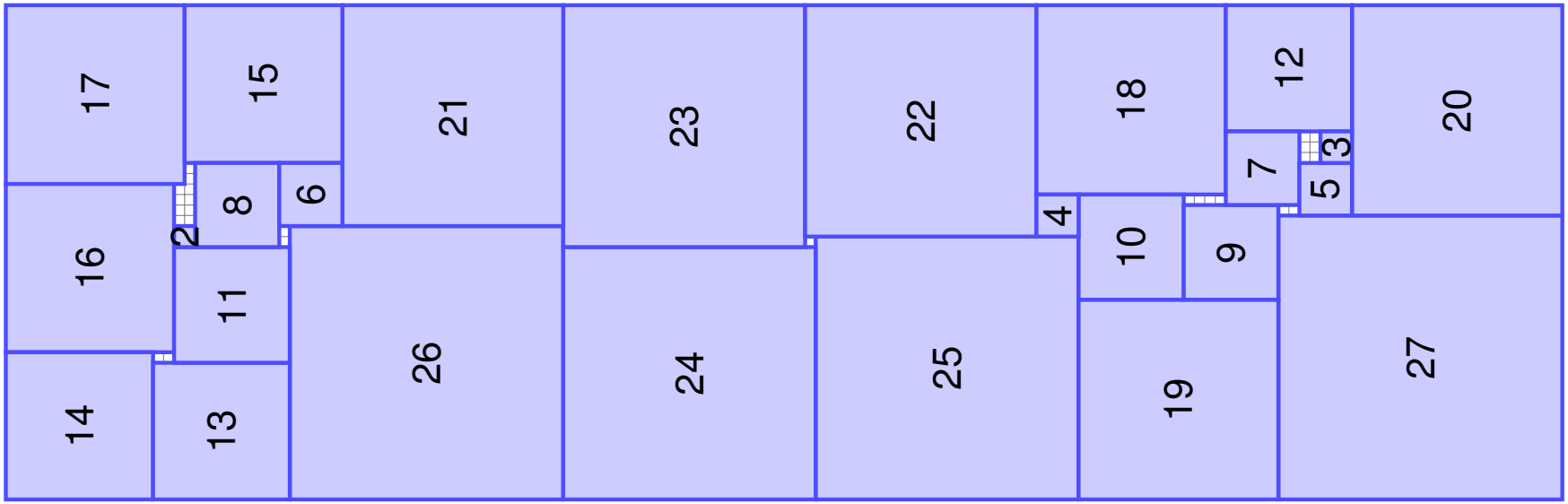
Strategies Comparison



Strategy Comparison

N	naive	xtheny	disj	semantic4	semantic	dual	interval 0.3	split 0.2	xy 0.75
15	2.92	0.09	12.12	0.55	0.45	2.63	-	0.05	-
16	10.44	0.11	98.25	1.31	1.03	0.89	-	0.05	-
17	20.75	0.27	23.57	1.48	1.13	0.33	0.05	0.05	0.81
18	667.33	18.37	-	30.53	23.05	118.58	1.83	1.13	13.94
19	4140.09	13.73	-	83.42	63.25	80.66	1.11	1.88	36.78
20	-	13.08	-	216.07	167.61	149.79	2.14	1.47	108.28
21	-	143.72	-	1138.98	865.13	-	8.09	10.59	619.45
22	-	1708.89	-	-	-	-	52.21	32.36	1668.59
23	-	-	-	-	-	-	245.07	265.54	9521.73
24	-	-	-	-	-	-	452.73	545.82	37506.20
25	-	-	-	-	-	-	2533.64	4127.41	-
26	-	-	-	-	-	-	14158.15	-	-
27	-	-	-	-	-	-	43529.87	-	-

Optimal Solution (N=27, CP 2008)



- Even better results by R. Korf in IJCAI 2009

Outline

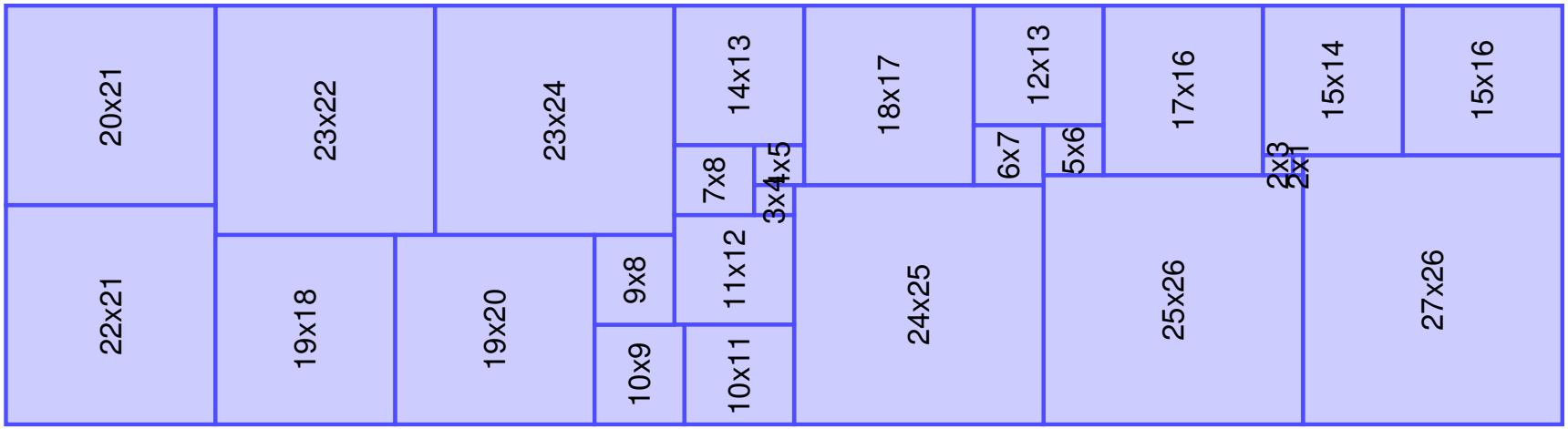
Back to the Future

- Apply lessons learned to almost square packing
- Added degree of freedom, rotation of items
- Weak impact, length only changes by one
- But 2^n additional choices

Expectations: What is feasible?

- We did find optimal solution for square packing up to 27
- Show of hands: Which problem size can we handle for almost squares?
- With 2^n additional choices

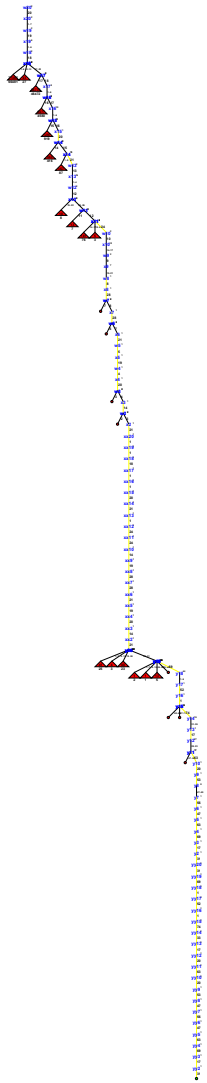
Almost Square, Optimal Solution (N=26)



Results

N	Surface	K	Width	Height	Area	Loss	Back	Time
3	20	1	4	5	20	0.00	2	00:00
4	40	1	4	10	40	0.00	2	00:00
5	70	1	5	14	70	0.00	2	00:00
6	112	3	6	19	114	1.79	5	00:00
7	168	3	12	14	168	0.00	6	00:00
8	240	4	15	16	240	0.00	5	00:00
9	330	6	14	24	336	1.82	20	00:00
10	440	6	17	26	442	0.45	199	00:00
11	572	3	22	26	572	0.00	55	00:00
12	728	8	21	35	735	0.96	340	00:00
13	910	3	26	35	910	0.00	1218	00:00
14	1120	4	28	40	1120	0.00	1695	00:00
15	1360	4	34	40	1360	0.00	8169	00:00
16	1632	4	32	51	1632	0.00	5362	00:00
17	1938	3	34	57	1938	0.00	124959	00:07
18	2280	4	30	76	2280	0.00	6711	00:00
19	2660	4	35	76	2660	0.00	15887	00:01
20	3080	4	35	88	3080	0.00	188708	00:15
21	3542	5	39	91	3549	0.20	7336903	09:31
22	4048	3	44	92	4048	0.00	20073166	26:19
23	4600	3	40	115	4600	0.00	7543034	12:37
24	5200	3	40	130	5200	0.00	11368115	18:23
25	5850	5	45	130	5850	0.00	15147701	28:44
26	6552	5	42	156	6552	0.00	35672212	01:04:14

Visualization: Search Tree (N=20)



Problem with Tree View

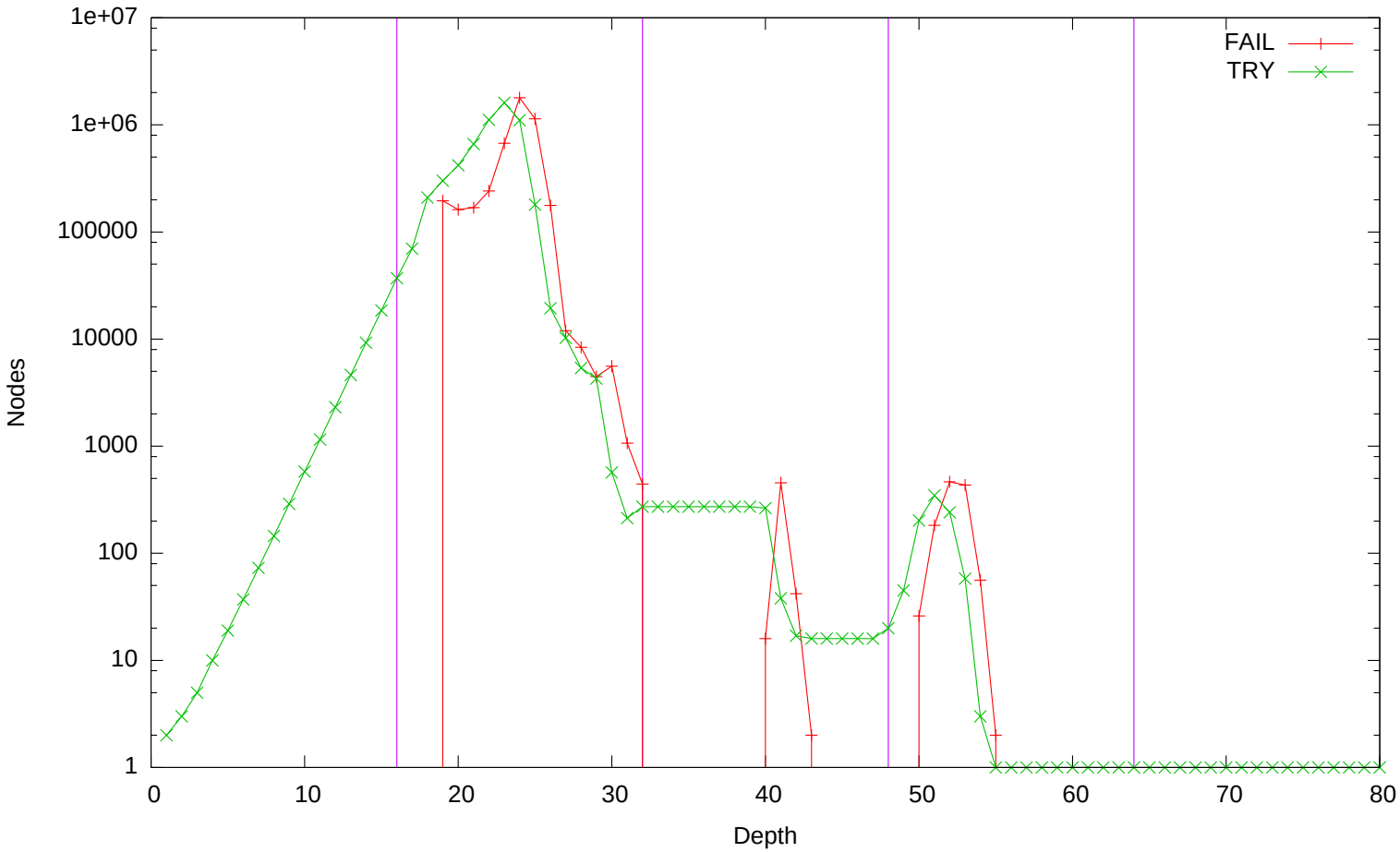
- Showing complete tree is clearly infeasible
- We only show path to solution
- But most time is spent in non-solution parts of tree

Search Choices

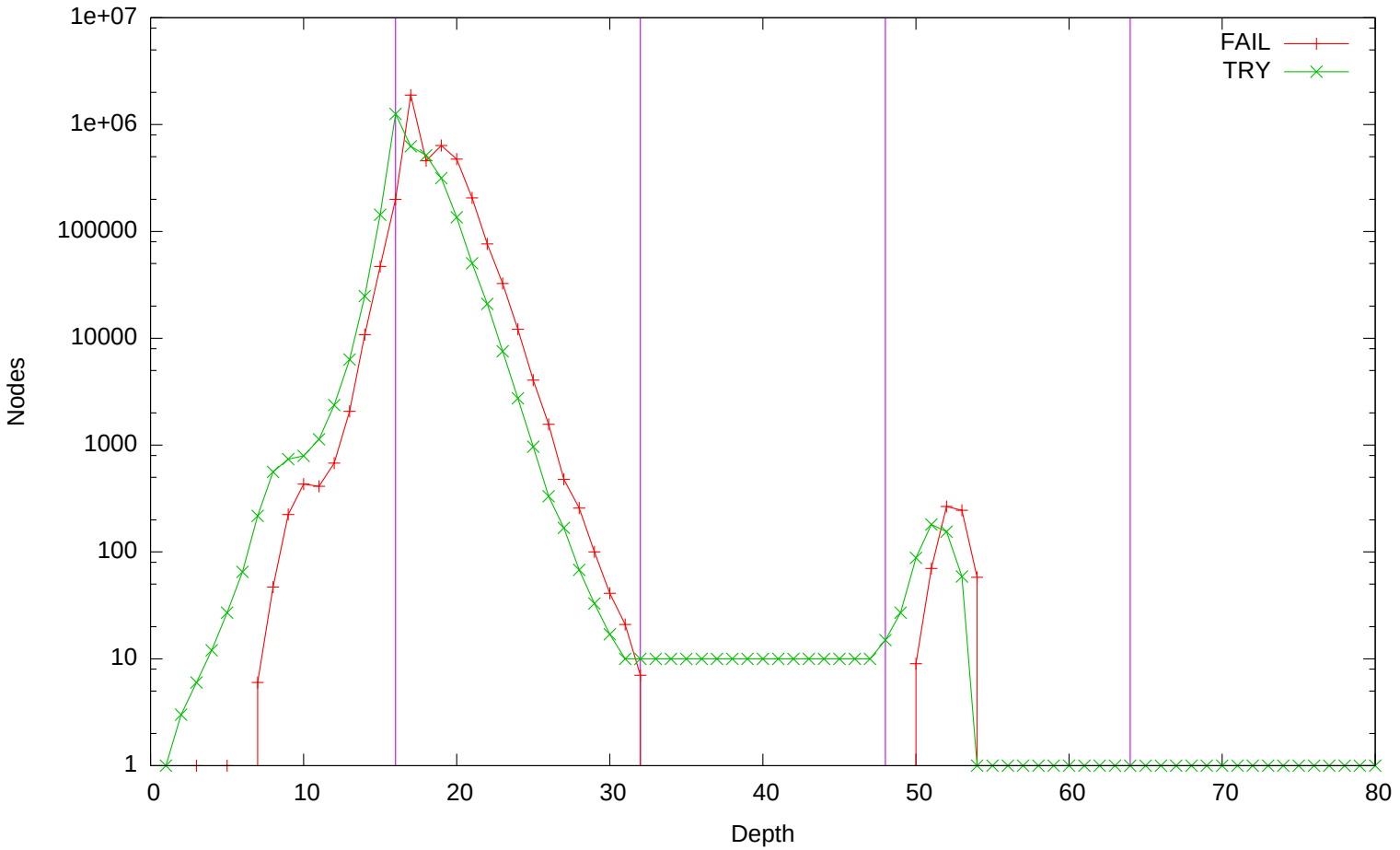
- Based on best method for square packing
- Assign X intervals, fix X values, assign Y intervals, fix Y values
- When to fix orientation?

Eager	Before assigning X intervals
Lazy	After assigning X intervals
Interleaved	Mixed with X interval assignment

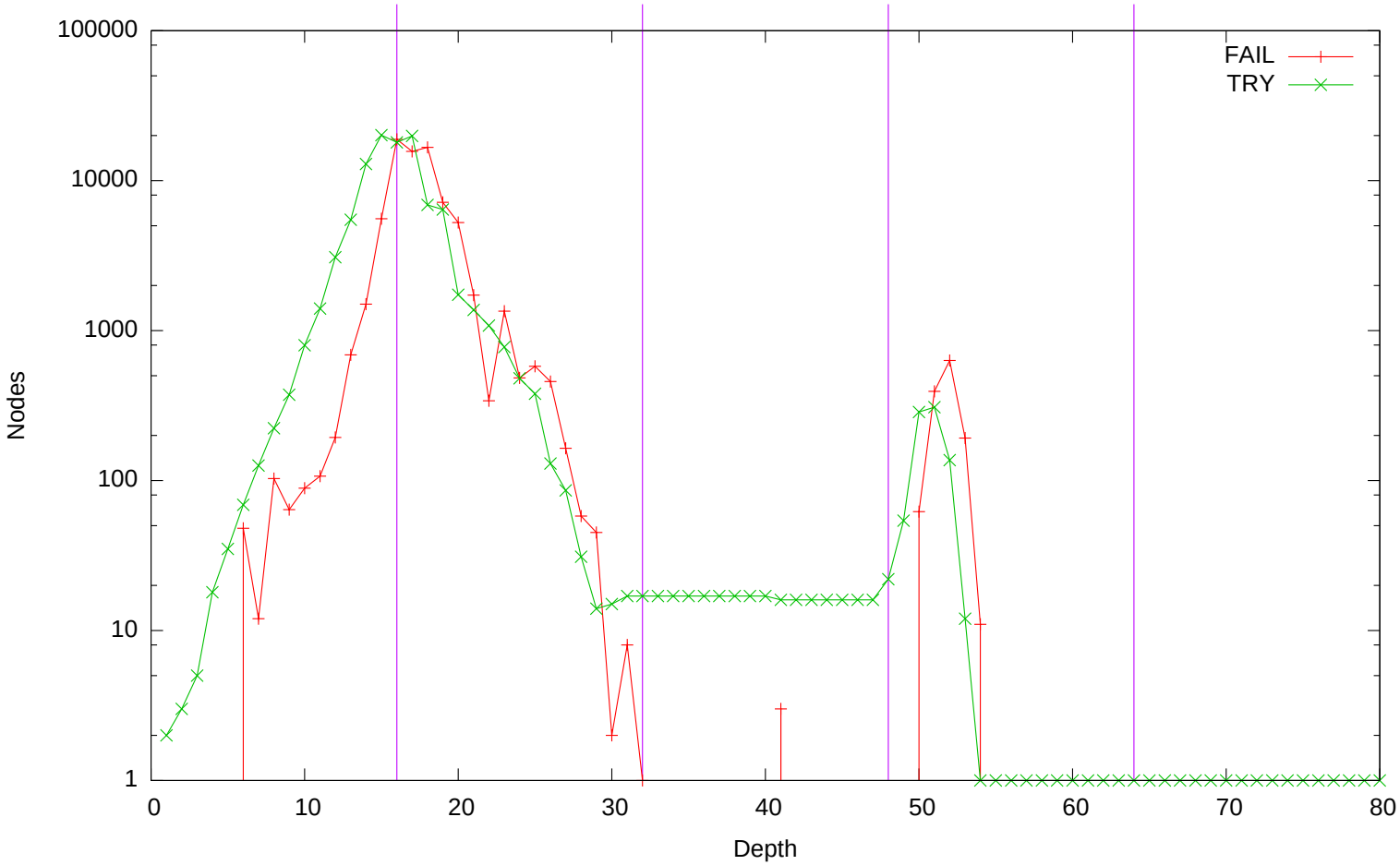
Eager Orientation (N=17)



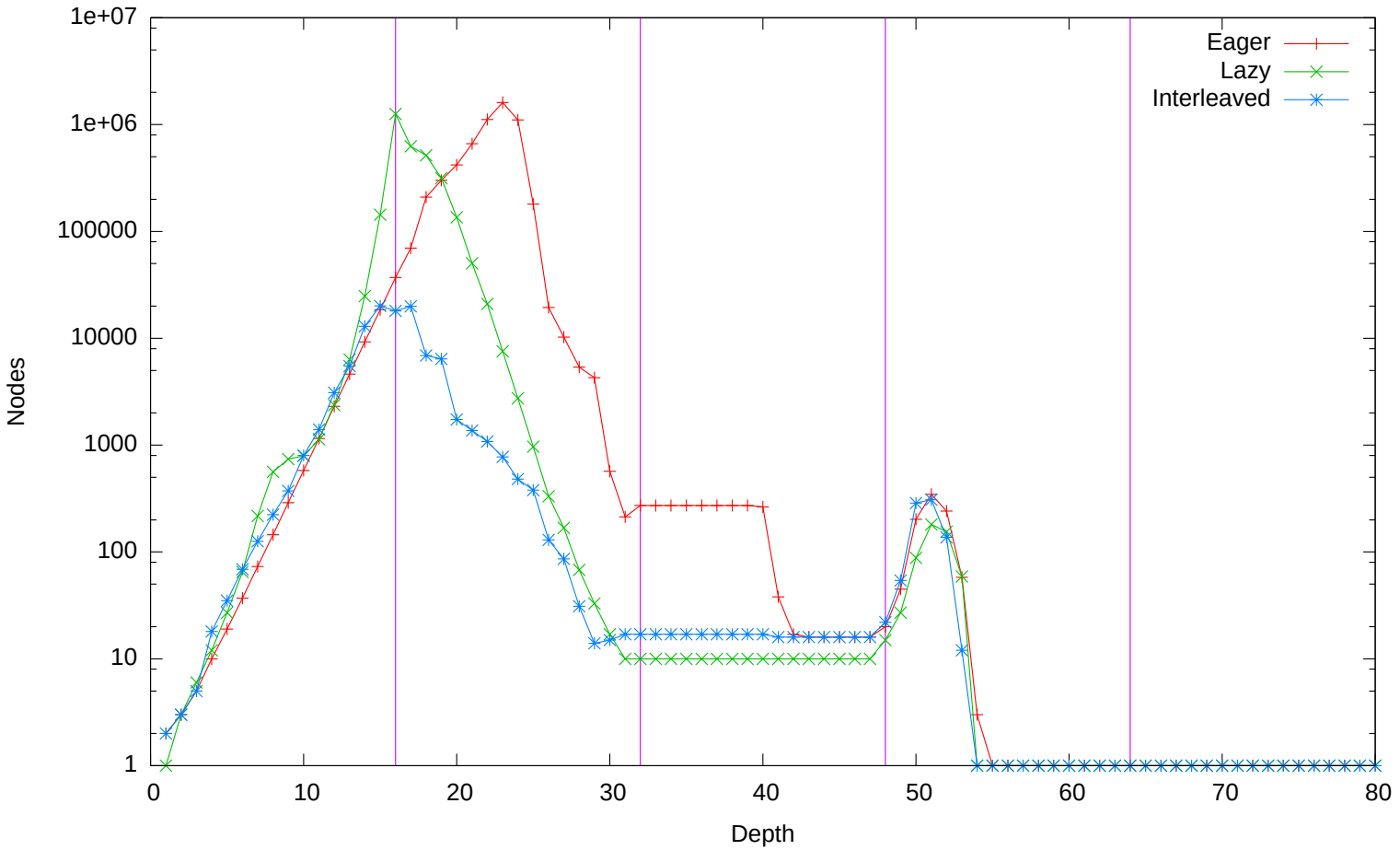
Lazy Orientation (N=17)



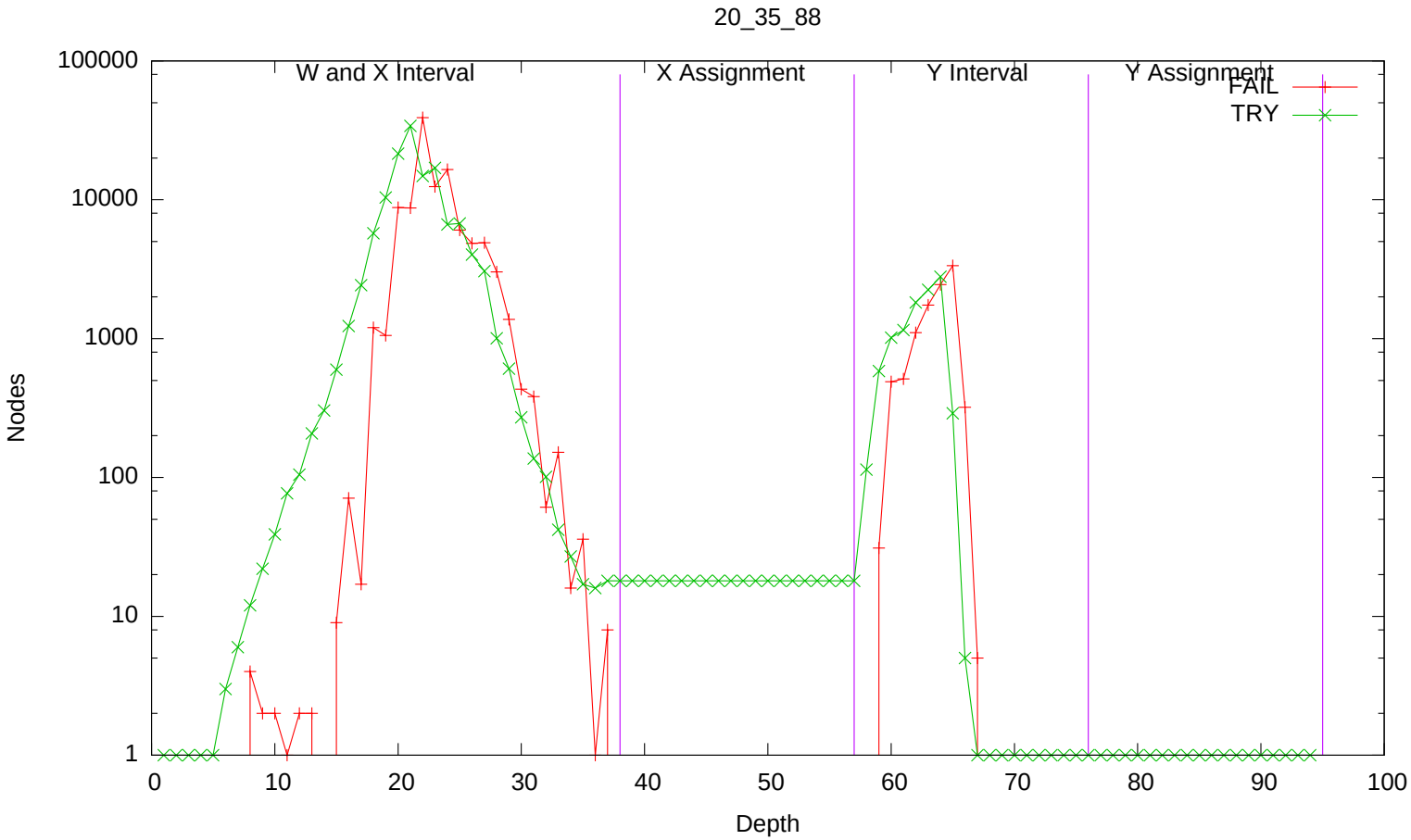
Interleaved Orientation (N=17)



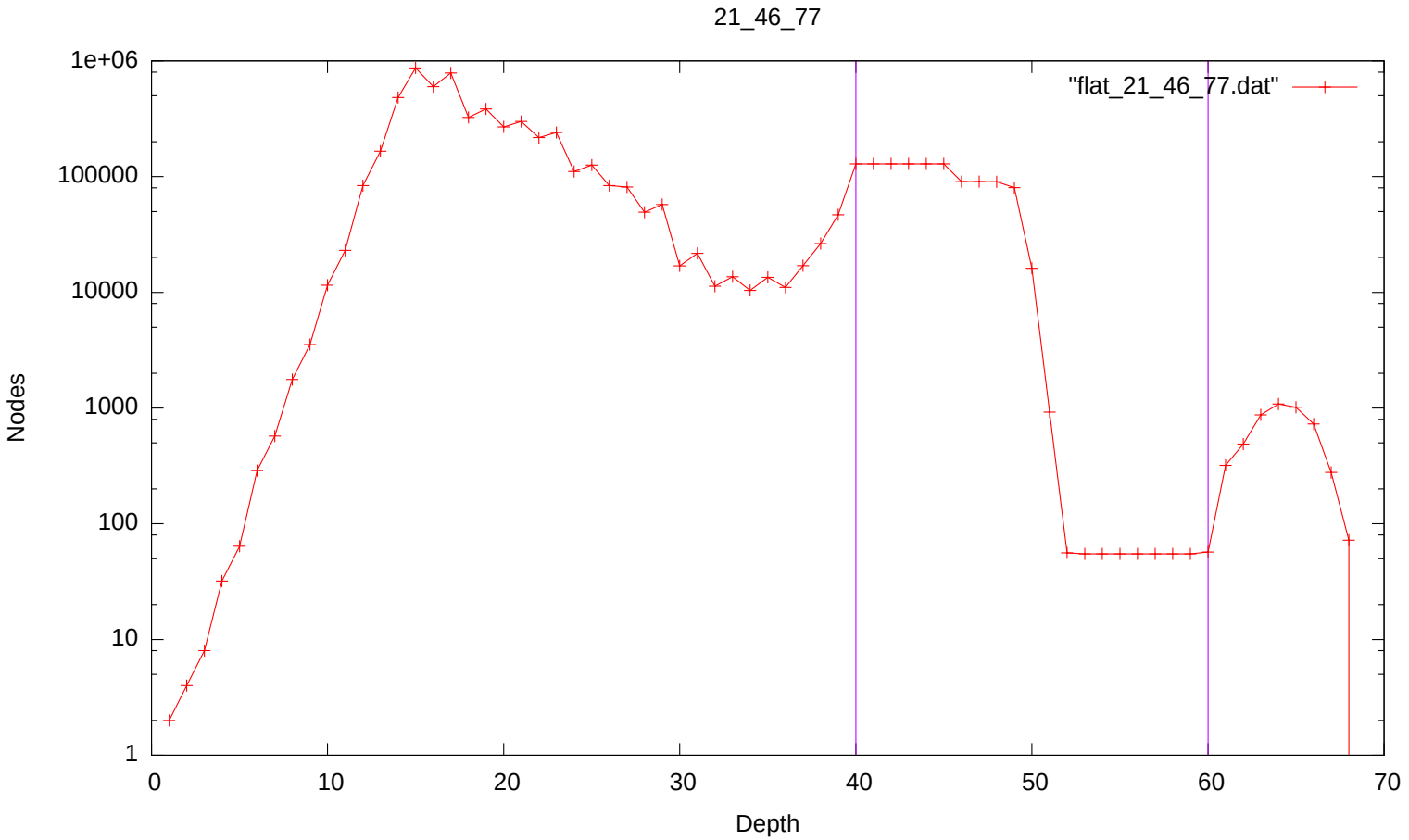
Comparison (N=17)



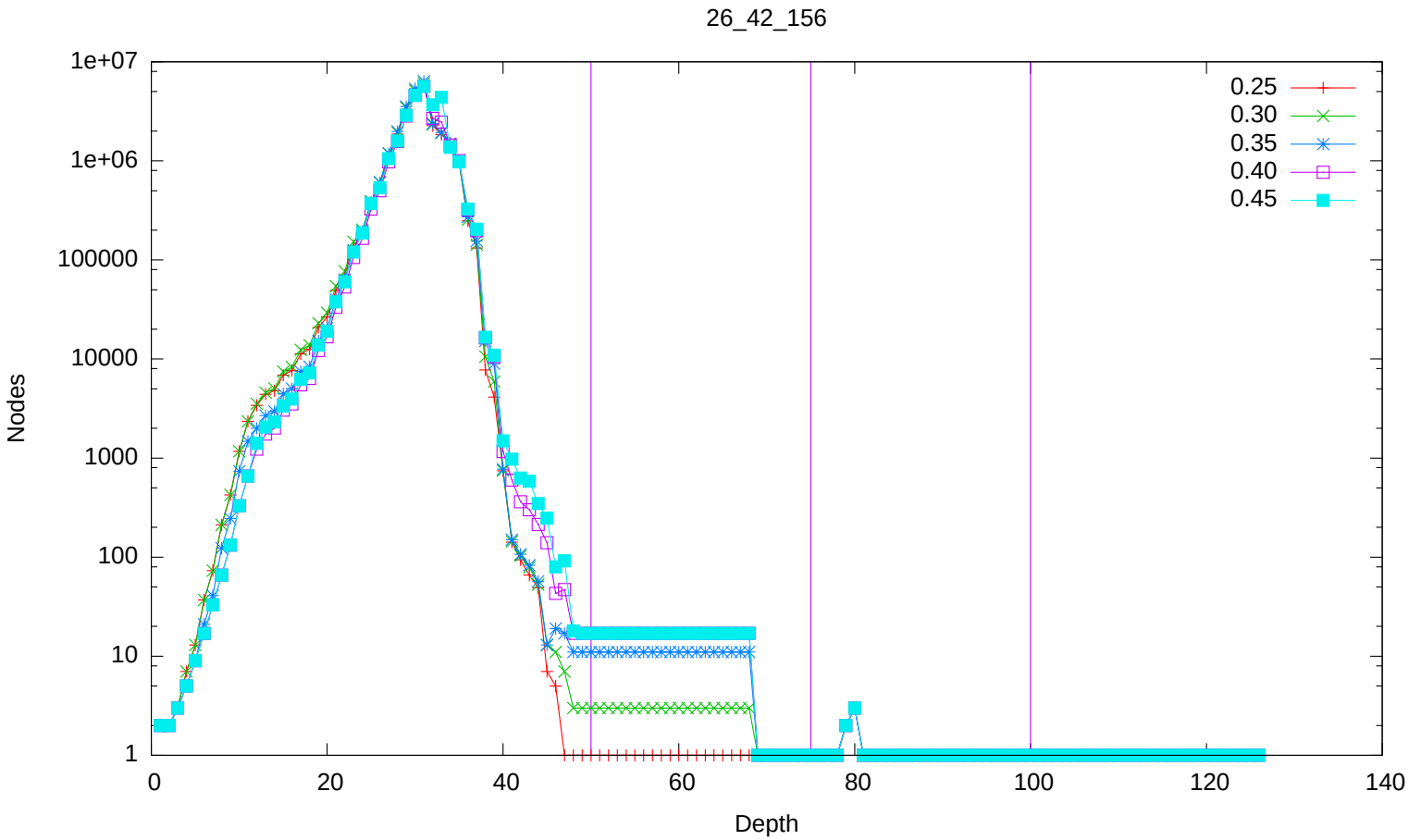
Node Distribution (N=20)



Problem with Slack (N=21)



Impact of Interval Size (N=26)



Lessons Learned, Future Work

- Little hope for further improvement of search for perfect problems
- Potential for problems with slack, ignore smaller items
- Need better reasoning
  - Dominance criterion for multiple areas at the same time
  - Dominance criterion for partially fixed items

Problem 2: Sports Scheduling

- A Nero Wolfe Mystery: Too Many Models
- How do we choose a good model
- No search at all

Outline

Sports Scheduling

Tournament Planning

We plan a tournament with 8 teams, where every team plays every other team exactly once. The tournament is played on 7 days, each team playing on each day. The games are scheduled in 7 venues, and each team should play in each venue exactly once.
As part of the TV arrangements, some preassignments are done: We may either fix the game between two particular teams to a fixed day and venue, or only state that some team must play on a particular day at a given venue. The objective is to complete the schedule, so that all constraints are satisfied.

Example

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

Solution

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		6, 8		1, 2	5, 7		3, 4
Day 2	2, 3	1, 5			4, 8	6, 7	
Day 3	1, 7	2, 4	3, 8				5, 6
Day 4			4, 7		2, 6	3, 5	1, 8
Day 5	5, 8			3, 6		1, 4	2, 7
Day 6		3, 7	1, 6	4, 5		2, 8	
Day 7	4, 6		2, 5	7, 8	1, 3		

A More Abstract Formulation

Rooms Puzzle, (Thomas G. Room, 1955)

Place numbers 1 to 8 in cells so that each row and each column has each number exactly once, each cell contains either no numbers or two numbers (which must be different from each other), and each combination of two different numbers appears in exactly one cell.

A More Abstract Formulation

Rooms Puzzle, (Thomas G. Room, 1955)

Place numbers 1 to 8 in cells so that each row and each column has each number exactly once, each cell contains either no numbers or two numbers (which must be different from each other), and each combination of two different numbers appears in exactly one cell.

Puzzle presented by R. Finkel

Outline

How to come up with a model

- What are the variables/what are their values?
- How can we express the constraints?
- Do we have these constraints in our system?
- Does this do good propagation?
- Backtrack to earlier step as required

Requirements

1. There are 8 teams, seven days and seven locations
2. Each team plays each other team exactly once
3. Each team plays 7 games (redundant)
4. Each team plays in each location exactly once
5. Each team plays on each day exactly once
6. A game consists of two (different) teams
7. There are four games on each day (redundant)
8. There are four games at each location (redundant)
9. In any location there is atmost one game at a time

Idea 1

- Matrix *Day* $\times$ *Game* (7×4)
- Each cell contains two variables, denoting teams
- Easy to say that team plays once on each day, `alldifferent`
- Columns don't have significance
- Model does not mention location, how to add this?
- How to express that each team plays each other once?

Idea 2, Change problem structure

- Matrix of *Day* $\times$ *Location* (7×7)
- Each cell contains two variables, each denoting a team
- How do we avoid symmetry inside cell?
- Need special value (0) to denote that there is no game
- In one cell, either both or none of the variables are 0
- Easy to say that each row and column contains each team exactly once
- Except for value 0, can not use `alldifferent`
- Link between two variables in cell to state that game needs two different teams
- How to express that each (ordered) pair occurs exactly once?

Idea 3, Add location variables

- Model as in Idea 1, matrix *Day* $\times$ *Game*
- Each cell contains two variables for teams and one for location
- Easy to state that games on one day are in different locations
- How to express condition that each team plays in each location once?
- Also, how to express that each team plays each other exactly once?

Idea 4, Use variables for pairs

- Matrix *Day* $\times$ *Location*
- Each cell contains one variable ranging over (sorted) pairs of teams, and special value 0 (no game)
- Each pair value occurs once, except for 0
  - Special constraint `alldifferent0`
  - Or use `gcc`
- How to state that each team plays once per day?
- How to state that each team plays in each location?

Idea 5: If all else fails, use binary variables

- Binary variable stating that team *i* plays in location *j* at day *k*
- Three dimensional matrix
- Each team plays once on each day
- Each team plays once in each location
- Each game has two (different) teams, needs auxiliary variable
- Each pair of team meets once, needs auxiliary variables

Idea 6: An even bigger binary model

- Use four dimensions
- Team *i* meets team *j* in location *k* on day *l*
- 3136 = 8*8*7*7 variables
- Constraints all linear
- Why use finite domain constraints?

Idea 7: A different mapping

- Each team plays each other exactly once, one variable for each combination (8*7/2=28 variables)
- Decide when and where this game is played, values range over combinations of days and locations (7*7=49 values)
- All variables must be different (no two games at same time and location)
- Each team plays 7 games, by construction
- How to express that each team plays once per day?
- How to express that each team plays in each location once?

Expand Idea 7 into Full Model

Numbering Values

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

Four games on each day

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

- Day 1 corresponds to values 1..7
- Four variables can take these values
- Day 2 corresponds to values 8..14, etc
- One constraint per day
- Exactly four of all variables take their value in the set ...
- Seven such constraints

Four games at each location

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

- City 1 corresponds to values
  - 1, 8, 15, 22, 29, 36, 43
- Four variables can take these values
- City 2 corresponds to values
  - 2, 9, 16, 23, 30, 37, 44
- One constraint per location
- Exactly four of all variables take their value in the set ...
- Seven such constraints over 28 variables each

Teams plays once on a day (at a location)

- Select those variables which correspond to Team *i*
- Exactly one of those variables takes its value in the set 1..7
- Same for all other days
- Same for all other teams
- 56 Constraints over 7 variables each
- Similar for teams and locations, another 56 constraints

Are we there yet?

- 28 variables with 49 possible values
- 1 alldifferent
- 7 exactly constraints over all variables (Days)
- 7 exactly constraints over all variables (Locations)
- 56 exactly constraints over 7 variables each (Days)
- 56 exactly constraints over 7 variables each (Locations)

Idea 8: Mapping games to days and locations

- For each game to be played, we have two variables
  - One ranges over the days
  - The other over the locations
- Easy to state that there are four games per day and location
- Easy to state that each team plays once per day and location
- How do we express that no two games are played at the same location and the same time?
  - If we had an `alldifferent` over pairs of variables...
  - Not in ECLiPSe

We have four games on each day

- Each row value is taken four times amongst the variables
- `gcc([gcc(4,4,1),...,gcc(4,4,7)],Rows)`
- Similar for columns:
- `gcc([gcc(4,4,1),...,gcc(4,4,7)],Cols)`

Reminder: `gcc(Pattern, Variables)`

- `gcc` *global cardinality constraint*
- `Pattern` is list of terms `gcc(Low, High, Value)`
- The overall number of variables taking value `Value` is between `Low` and `High`
- Generalization of `alldifferent`
- Domain consistent version in ECLiPSe

Each team plays once per day

- For the seven variables which describe games of a team
- Each row value is taken exactly once amongst the variables
- Could use
  `gcc([gcc(1,1,1),...,gcc(1,1,7)],Vars)`
- But `alldifferent(Vars)` is more compact
- Similar for columns

How do the models differ?

Idea	Mapping
1	$D \times G \times \{f, s\} \rightarrow T$
2	$D \times L \times \{f, s\} \rightarrow T \cup \{0\}$
3	$D \times G \times \{f, s\} \rightarrow T$ $D \times G \rightarrow L$
4	$D \times L \rightarrow T \triangle T \cup \{0\}$
5	$T \times D \times L \rightarrow \{0, 1\}$
6	$T \times T \times D \times L \rightarrow \{0, 1\}$
7	$T \triangle T \rightarrow D \times L$
8	$T \triangle T \rightarrow D$ $T \triangle T \rightarrow L$

D	Days
T	Teams
L	Locations
G	Games

Comments on models

Idea	Main point
1	missing locations, first second symmetry
2	spare value, first second symmetry
3	first second symmetry
4	spare value
5	0/1, non-linear constraints
6	0/1, large matrix
7	needs exactly constraint
8	needs alldifferent on tuples

Viewpoints and Channeling

- Instead of expressing all constraints over one set of variables
- Use multiple sets of variables (*viewpoints*)
- Decide which constraint to express over which variables
- Allows more freedom on how to express problem
- Link the different variables with *channeling* constraints

In Our Case

- Combine ideas 7 and 8
- One set of variables ranging over pairs
- Another using two variables per game for day and location
- How to combine variables?
- Minimize loss of information

Projection

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

- Link pair variables to row and column variables
- Pair variable uses cell numbers 1-49 as values
- Row and column variables indicate on which day (row) and in which location (column) the game is played
- Pair value 23 = row 4, column 2
- `element` constraint to link the variables
- Two projections from $D \times L$ space onto D and L

Channeling Constraints

- This is one common type, a *projection*
- Another common type is the *inverse*
  - Link a variable $A \rightarrow B$ to another $B \rightarrow A$
  - Typically used for bijective mappings
  - Built-in `inverse/2`
- Also used: *Boolean* channeling
  - Link variables $A \rightarrow B$ and $A \times B \rightarrow \{0, 1\}$
  - Built-in `bool_channeling/3`

Outline

Selected Model

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables ($T \triangle T \rightarrow D \times L$)
  - `alldifferent` (Req 9)
- Day and Location variables ($T \triangle T \rightarrow D$), ($T \triangle T \rightarrow L$)
  - `gcc` (Req 4, 5)
  - `alldifferent` (Req 7, 8)
- Channeling Constraints
  - `element` projection from pairs onto rows and columns
- Search only on pair variables

Selected Model

Req 1: There are 8 teams, seven days and seven locations

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables ($T \triangle T \rightarrow D \times L$)
  - `alldifferent` (Req 9)
- Day and Location variables ($T \triangle T \rightarrow D$), ($T \triangle T \rightarrow L$)
  - `gcc` (Req 4, 5)
  - `alldifferent` (Req 7, 8)
- Channeling Constraints
  - `element` projection from pairs onto rows and columns
- Search only on pair variables

Selected Model

Req 2: Each team plays each other team exactly once

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables ($T \triangle T \rightarrow D \times L$)
  - `alldifferent` (Req 9)
- Day and Location variables ($T \triangle T \rightarrow D$), ($T \triangle T \rightarrow L$)
  - `gcc` (Req 4, 5)
  - `alldifferent` (Req 7, 8)
- Channeling Constraints
  - `element` projection from pairs onto rows and columns
- Search only on pair variables

Selected Model

Req 3: Each team plays 7 games

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables ($T \triangle T \rightarrow D \times L$)
  - `alldifferent` (Req 9)
- Day and Location variables ($T \triangle T \rightarrow D$), ($T \triangle T \rightarrow L$)
  - `gcc` (Req 4, 5)
  - `alldifferent` (Req 7, 8)
- Channeling Constraints
  - `element` projection from pairs onto rows and columns
- Search only on pair variables

Selected Model

Req 4: Each team plays in each location exactly once

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables ($T \triangle T \rightarrow D \times L$)
  - `alldifferent` (Req 9)
- Day and Location variables ($T \triangle T \rightarrow D$), ($T \triangle T \rightarrow L$)
  - `gcc` (Req 4, 5)
  - `alldifferent` (Req 7, 8)
- Channeling Constraints
  - `element` projection from pairs onto rows and columns
- Search only on pair variables

Selected Model

Req 5: Each team plays on each day exactly once

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables ($T \triangle T \rightarrow D \times L$)
  - `alldifferent` (Req 9)
- Day and Location variables ($T \triangle T \rightarrow D$), ($T \triangle T \rightarrow L$)
  - `gcc` (Req 4, 5)
  - `alldifferent` (Req 7, 8)
- Channeling Constraints
  - `element` projection from pairs onto rows and columns
- Search only on pair variables

Selected Model

Req 6: A game consists of two (different) teams

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables ($T \triangle T \rightarrow D \times L$)
  - `alldifferent` (Req 9)
- Day and Location variables ($T \triangle T \rightarrow D$), ($T \triangle T \rightarrow L$)
  - `gcc` (Req 4, 5)
  - `alldifferent` (Req 7, 8)
- Channeling Constraints
  - `element` projection from pairs onto rows and columns
- Search only on pair variables

Selected Model

Req 7: There are four games on each day

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables ($T \triangle T \rightarrow D \times L$)
  - `alldifferent` (Req 9)
- Day and Location variables ($T \triangle T \rightarrow D$), ($T \triangle T \rightarrow L$)
  - `gcc` (Req 4, 5)
  - `alldifferent` (Req 7, 8)
- Channeling Constraints
  - `element` projection from pairs onto rows and columns
- Search only on pair variables

Selected Model

Req 8: There are four games at each location

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables ($T \triangle T \rightarrow D \times L$)
  - `alldifferent` (Req 9)
- Day and Location variables ($T \triangle T \rightarrow D$), ($T \triangle T \rightarrow L$)
  - `gcc` (Req 4, 5)
  - `alldifferent` (Req 7, 8)
- Channeling Constraints
  - `element` projection from pairs onto rows and columns
- Search only on pair variables

Selected Model

Req 9: In any location there is atmost one game at a time

- Two sets of variables (Req 1, 2, 3, 6, by construction)
- Pair variables ($T \triangle T \rightarrow D \times L$)
  - `alldifferent` (Req 9)
- Day and Location variables ($T \triangle T \rightarrow D$), ($T \triangle T \rightarrow L$)
  - `gcc` (Req 4, 5)
  - `alldifferent` (Req 7, 8)
- Channeling Constraints
  - `element` projection from pairs onto rows and columns
- Search only on pair variables

Handling of hints (I)

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

- This value (17) can not be used by pairs not involving team 8
- One of the pairs involving team 8 must use this value (17)

Handling of hints (I)

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

- This value (17) can not be used by pairs not involving team 8
- One of the pairs involving team 8 must use this value (17)

Handling of hints (I)

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

- This value (17) can not be used by pairs not involving team 8
- One of the pairs involving team 8 must use this value (17)

Handling of hints (II)

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

- The pair involving teams 5 and 7 must take value 5, fixes variable

Handling of hints (II)

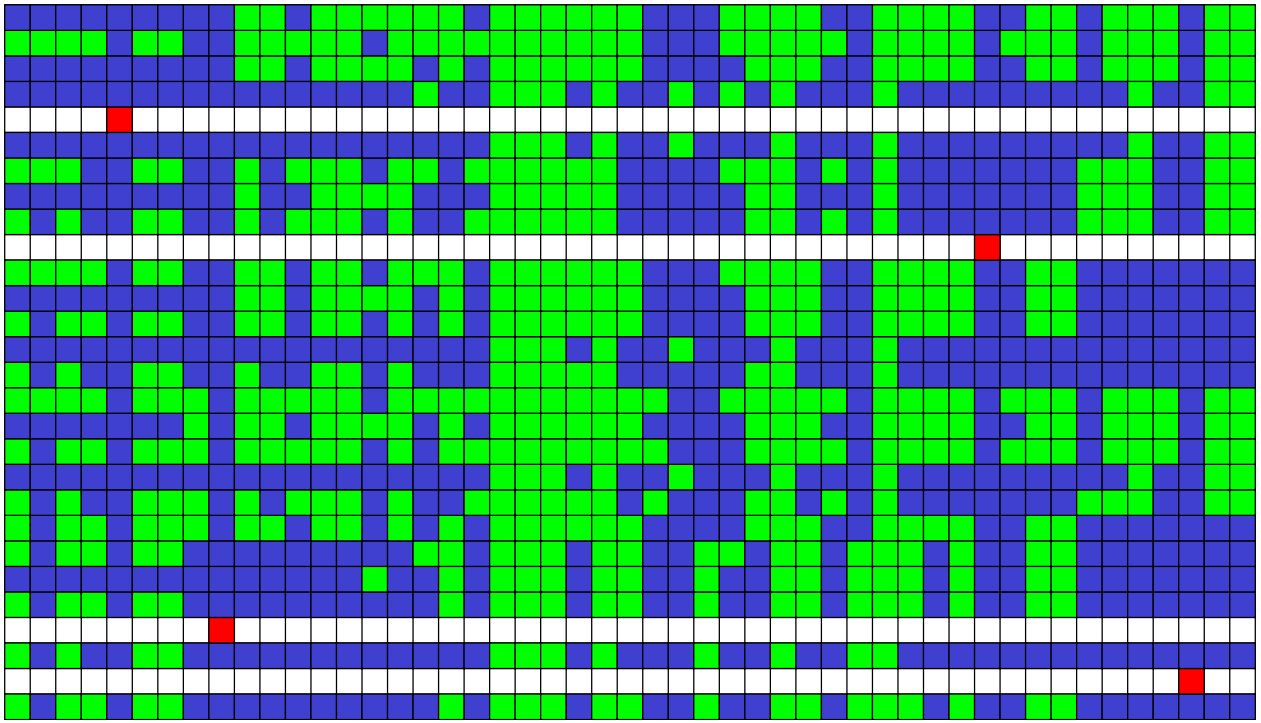
	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

- The pair involving teams 5 and 7 must take value 5, fixes variable

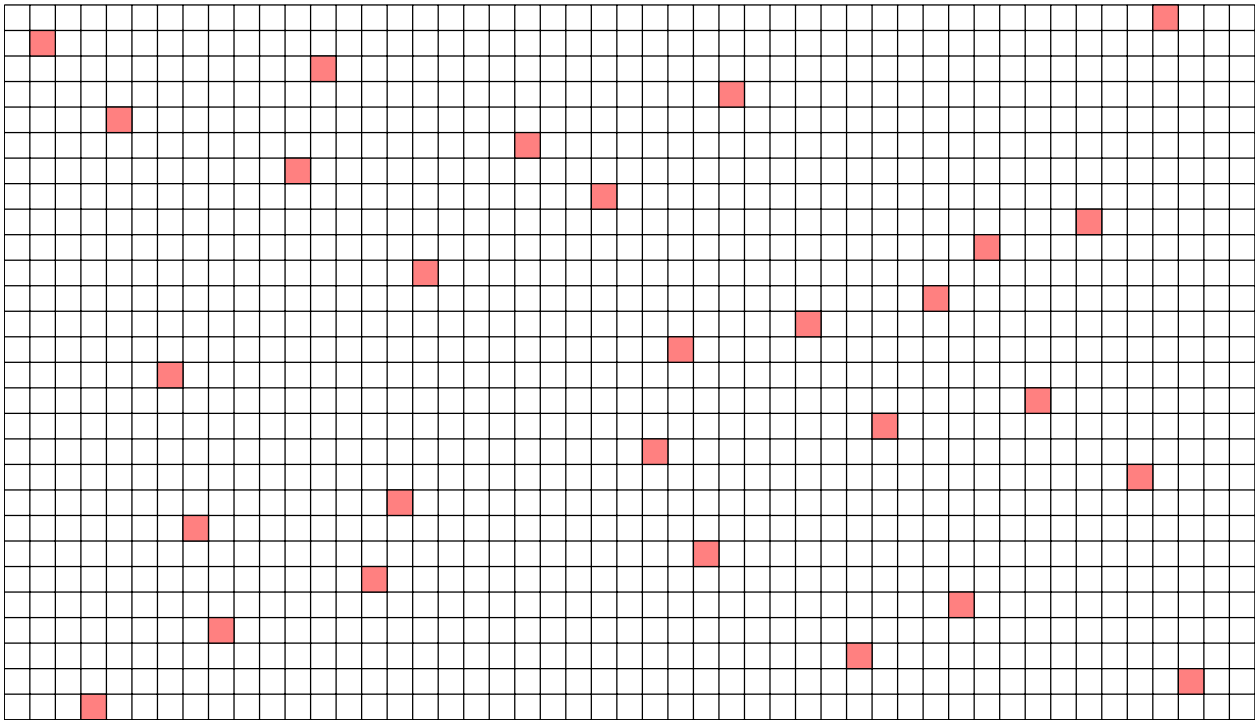
Before Search

Values

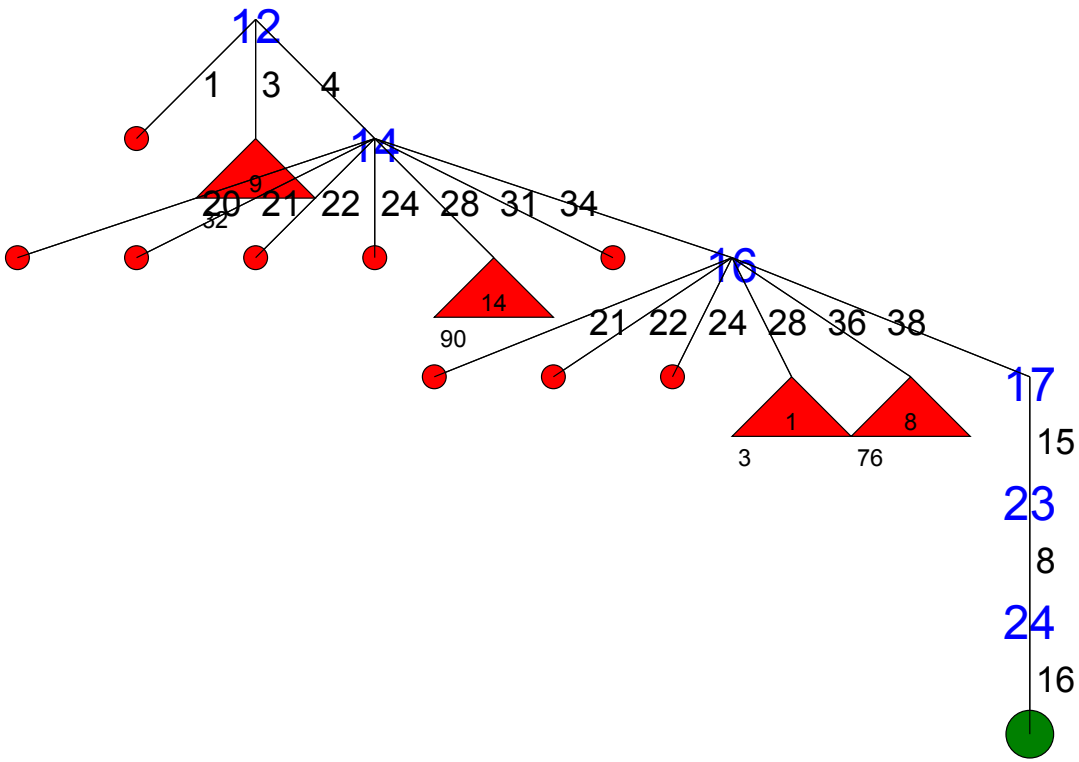
Vars



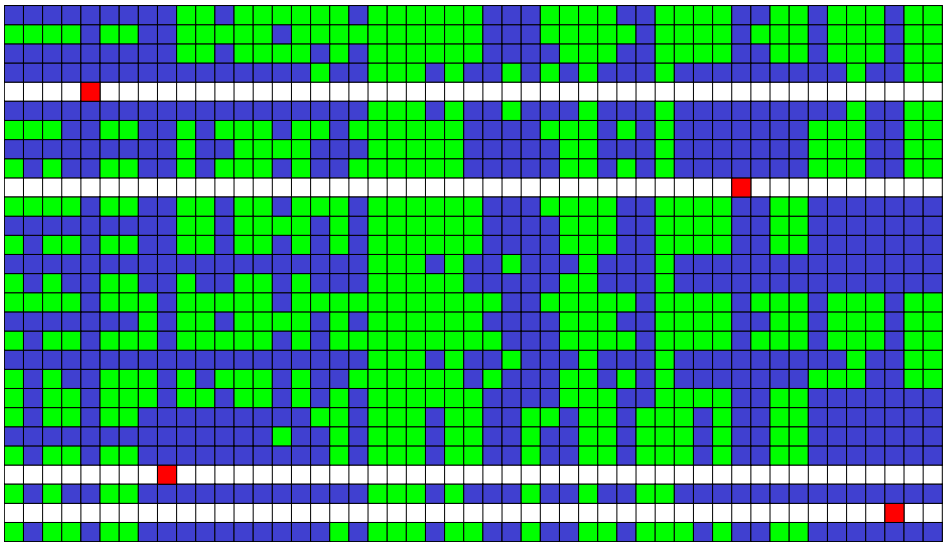
Solution



Search Tree with input order



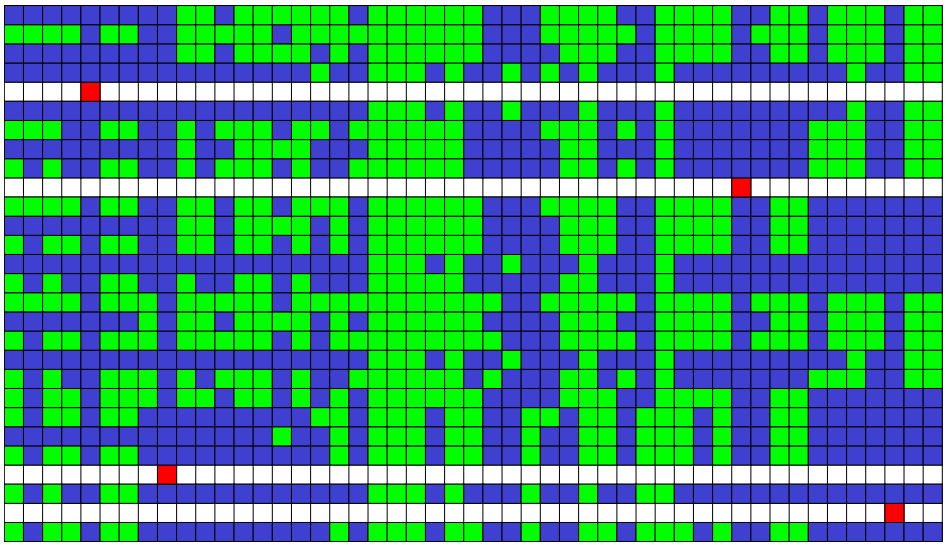
Missing Propagation



	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

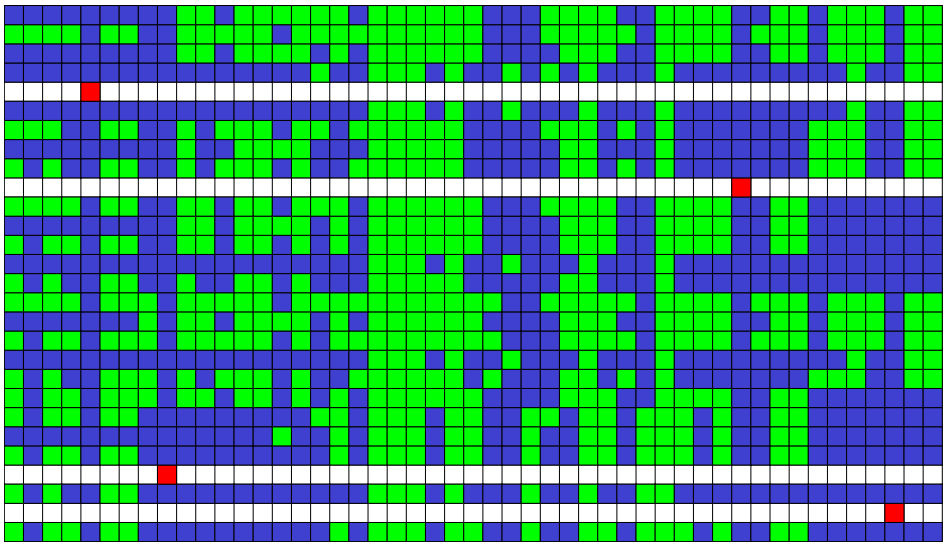
Missing Propagation



	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

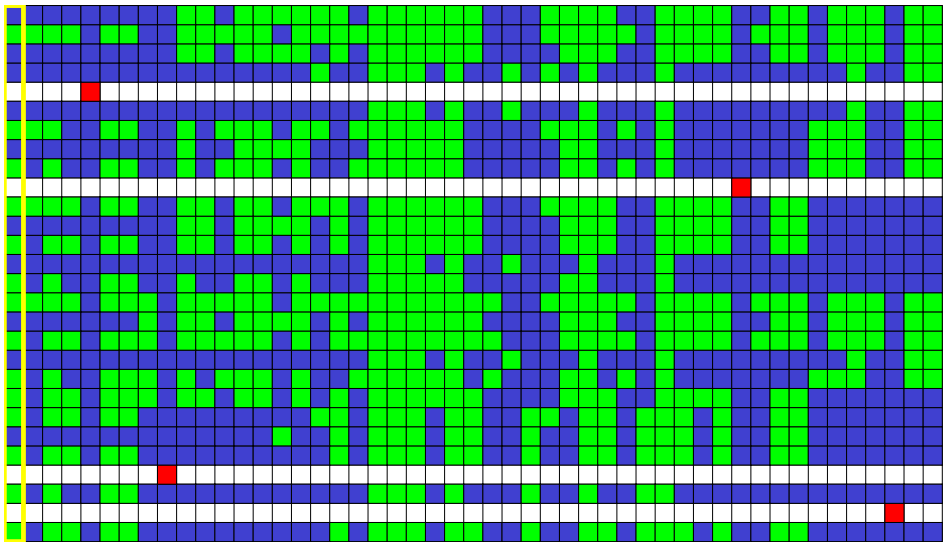
Missing Propagation



	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

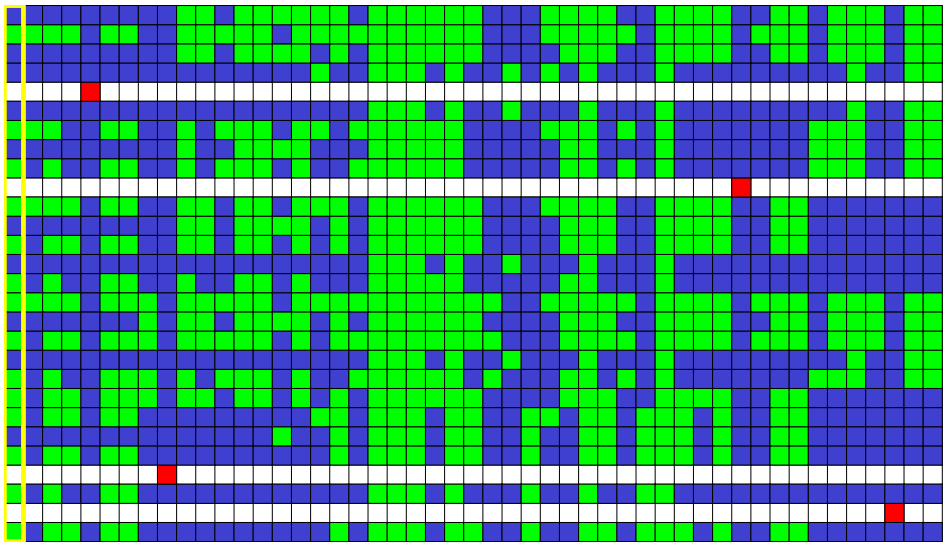
Missing Propagation



	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

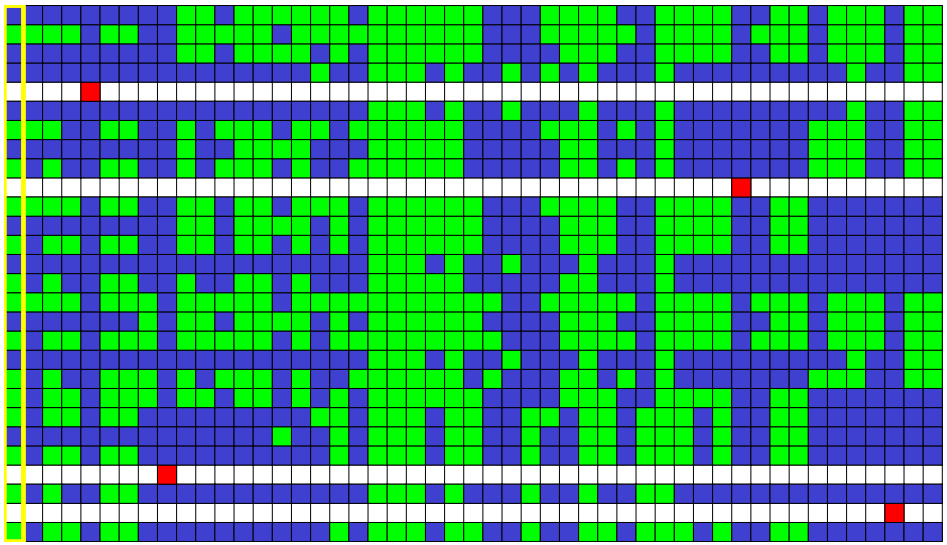
Missing Propagation



	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

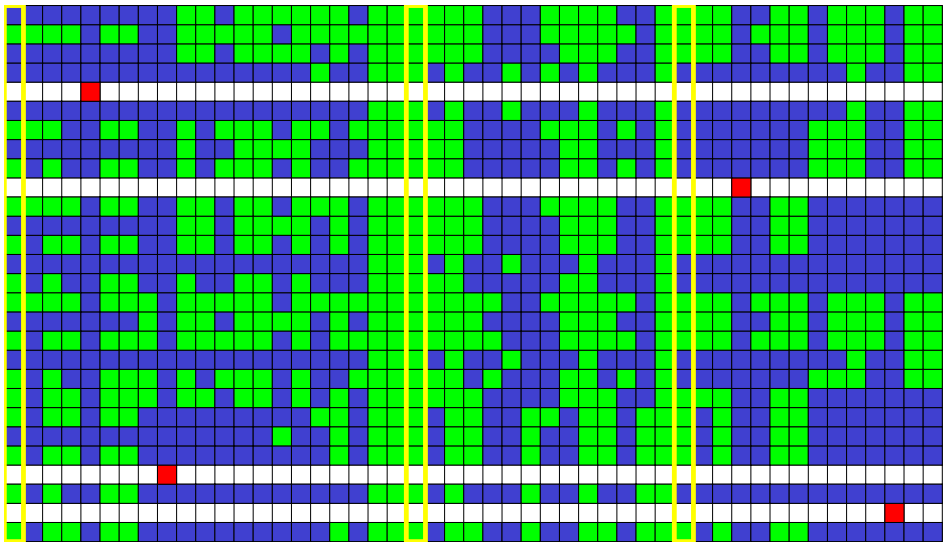
Missing Propagation



	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

Missing Propagation



	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

Why is this?

- Constraints involved:
  - `gcc` constraint on row: four variables can use values from this row
  - four `occurrence` constraints for hints: One of the variables must take this value
- No interaction between constraints, only between constraints and variables
- We do not detect that value 1 can not be used
- Eventual solution respects condition, model is correct
- We are concerned about propagation, not just correctness

Adding Redundant Constraints

- Add constraints which do more propagation, but do not affect solutions
- Lead to smaller search tree, hopefully faster solution
- Introduction requires understanding of (lack of) propagation
- Visualization is key to detect missing propagation

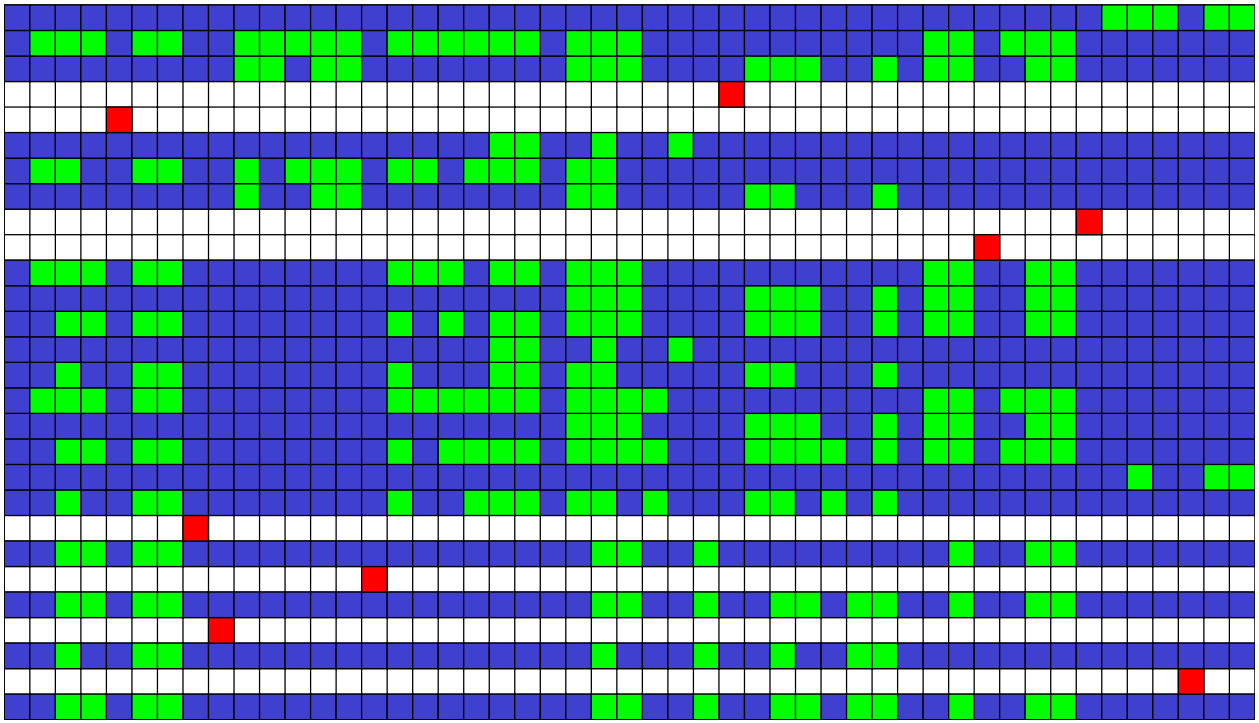
First Attempt: Adding 0/1 Viewpoint

- *Day* $\times$ *Location* matrix of 0/1 variables
- Indicates if there is a game on this day at this location
- Row/column sums: 4 games in each row/column
- Hint given for cell: Game variable is 1

Channeling Constraint

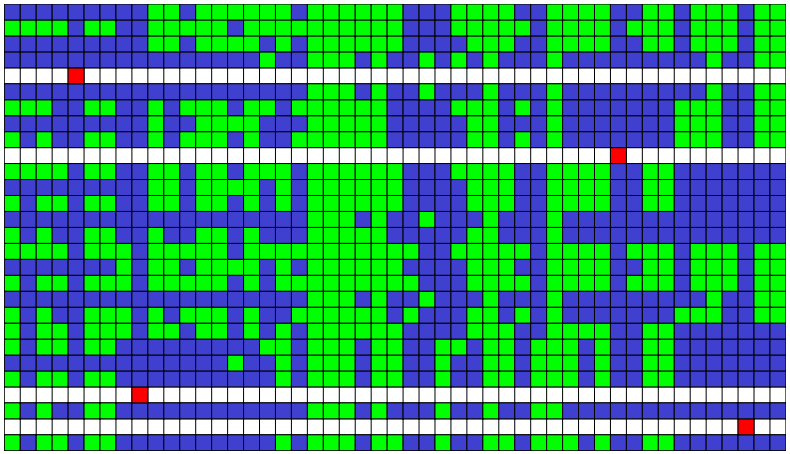
- Link pair variables P_i to 0/1 variables Y_j as *value-index*
- $(\exists i \text{ s.t. } P_i = v) \Leftrightarrow Y_v = 1$
- Propagation:
  - $P_i = v \Rightarrow Y_v = 1$
  - $Y_v = 0 \Rightarrow \forall i : \ P_i \neq v$
  - $(\forall i : \ v \notin d(P_i)) \Rightarrow Y_v = 0$
  - $Y_v = 1 \Rightarrow \text{occurrence}(V, P_1 ... P_n, N), N \geq 1$

Before Search

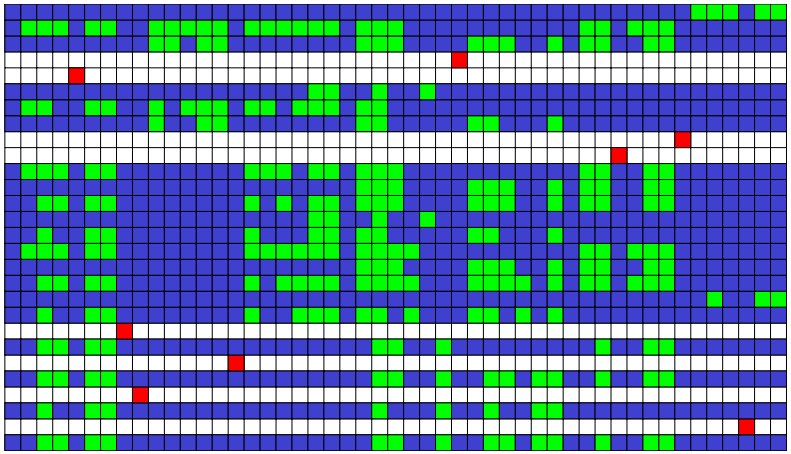


Impact of Redundant Constraints

Without

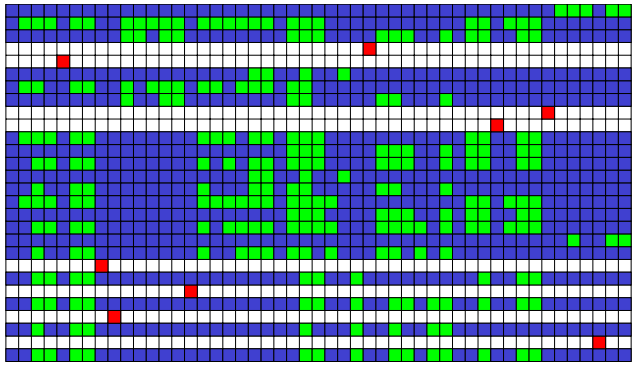


With value index channeling



Search tree with redundant constraints

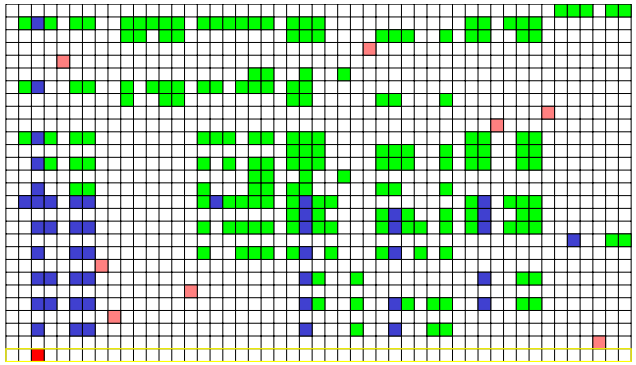
12



Skip Animation

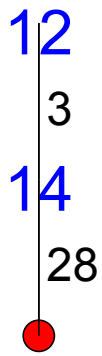
Search tree with redundant constraints

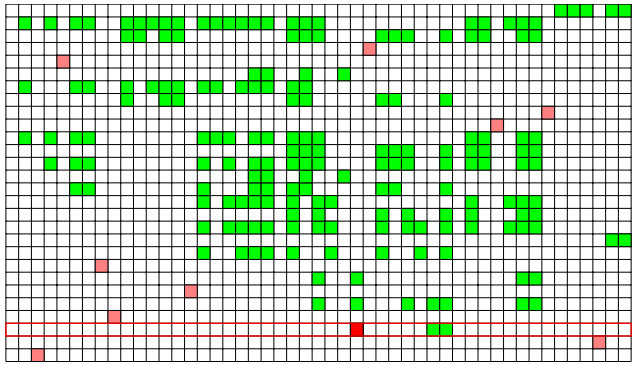
12
3
14



Back to Start Skip Animation

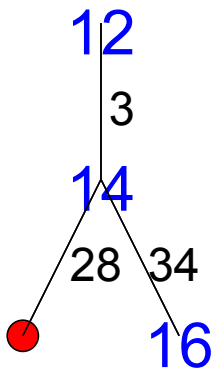
Search tree with redundant constraints

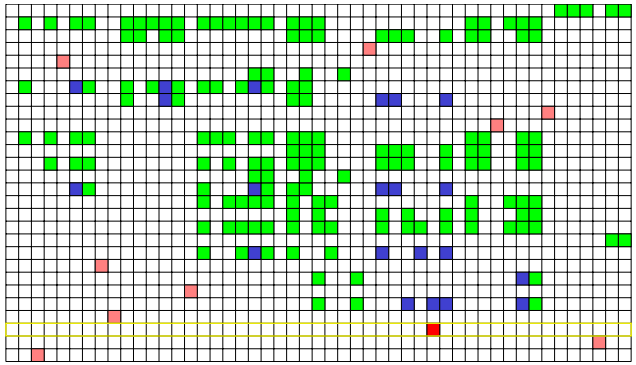




Back to Start Skip Animation

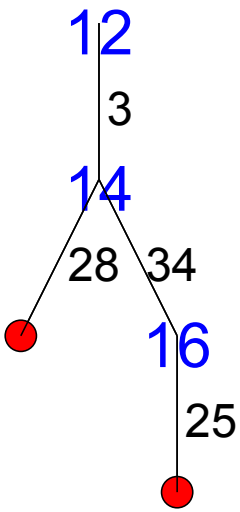
Search tree with redundant constraints

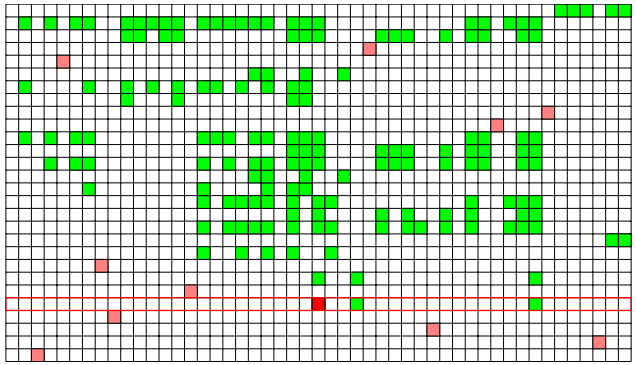




Back to Start Skip Animation

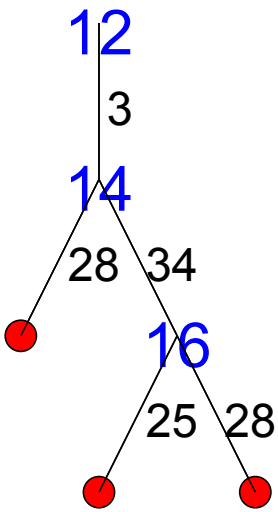
Search tree with redundant constraints

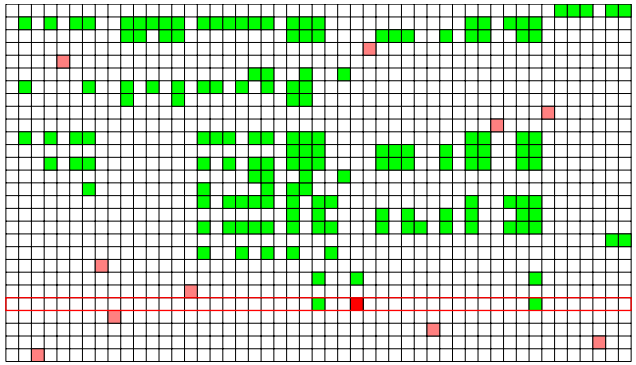




Back to Start Skip Animation

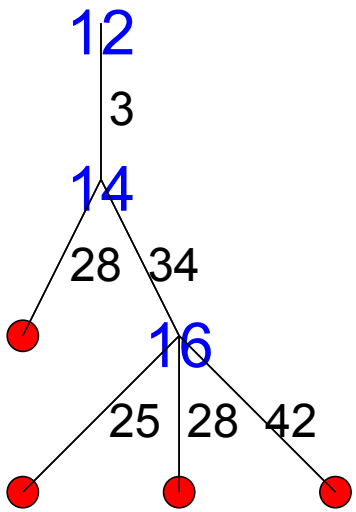
Search tree with redundant constraints

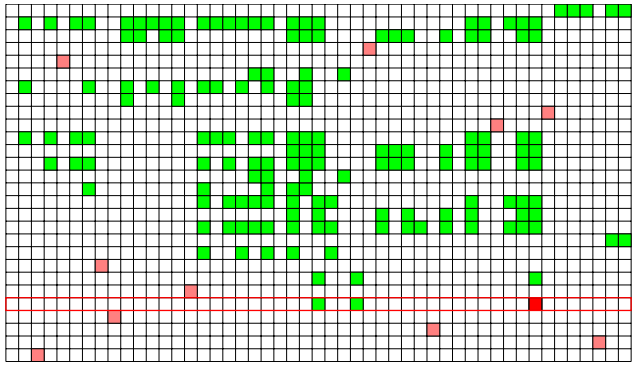




Back to Start Skip Animation

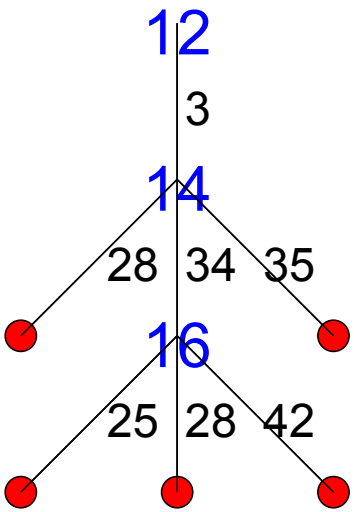
Search tree with redundant constraints

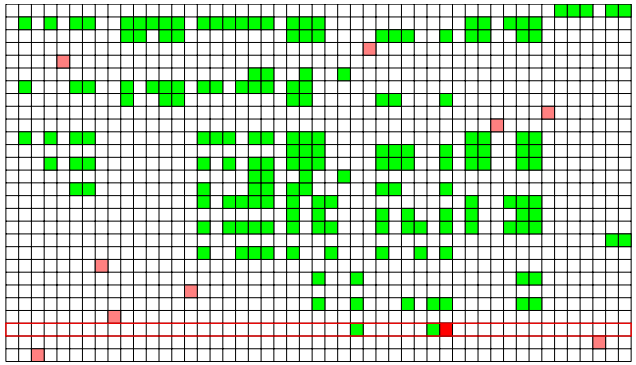




Back to Start Skip Animation

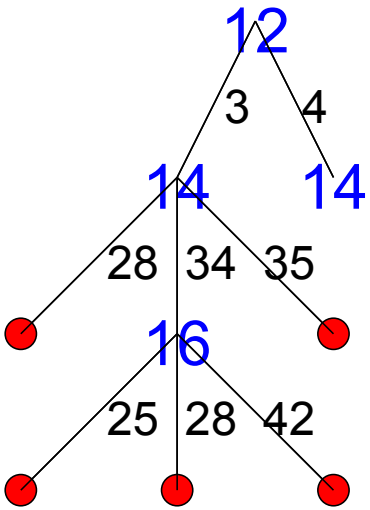
Search tree with redundant constraints

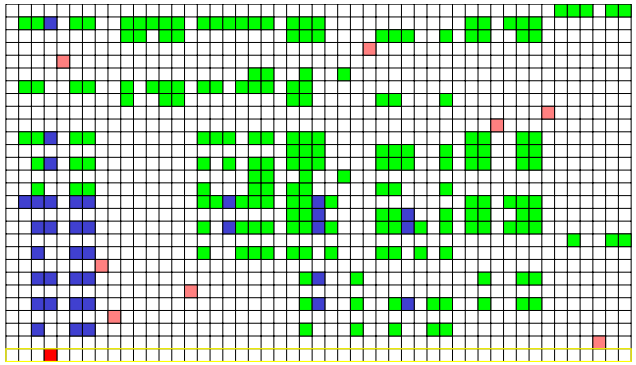




Back to Start Skip Animation

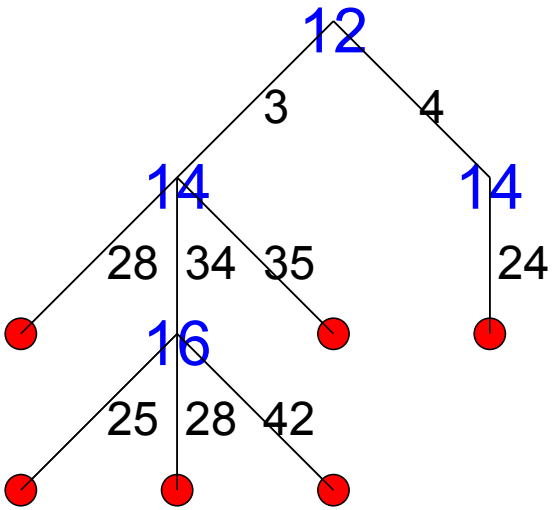
Search tree with redundant constraints

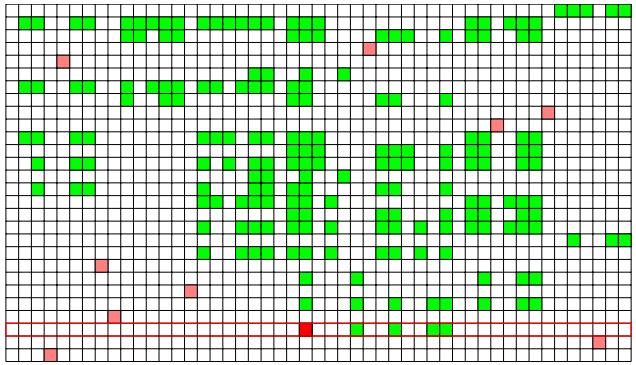




Back to Start Skip Animation

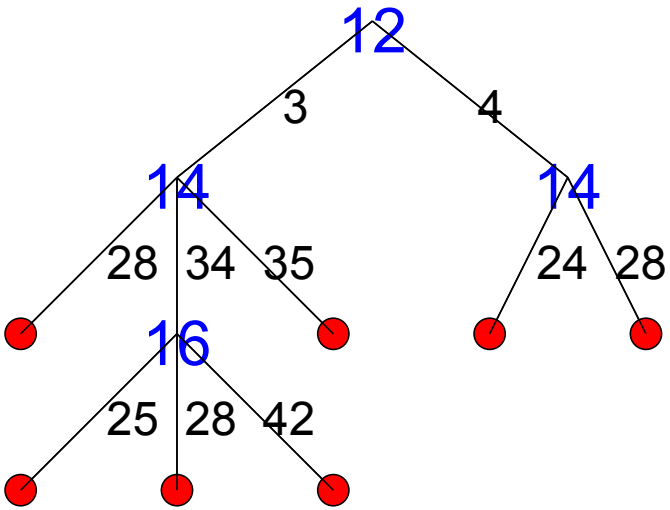
Search tree with redundant constraints

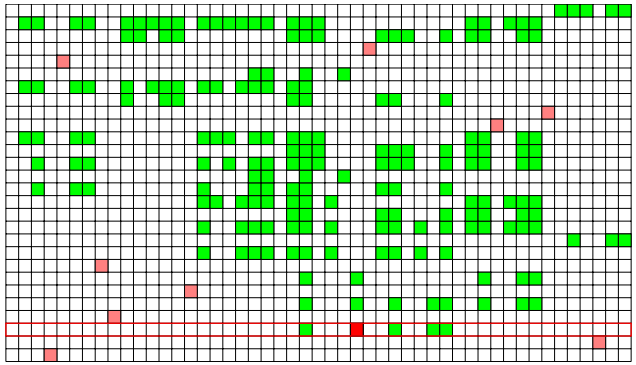




Back to Start Skip Animation

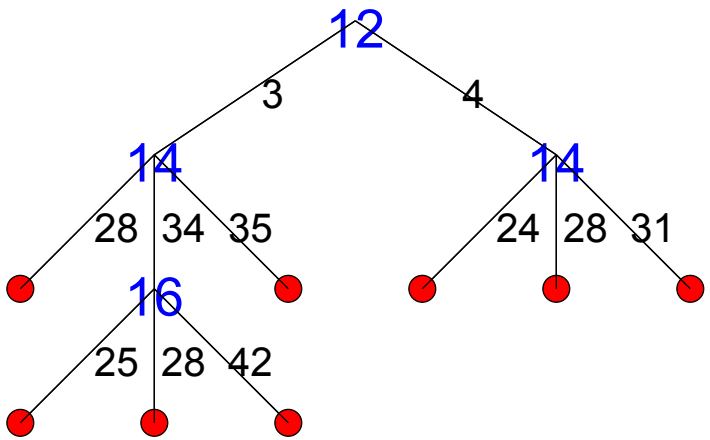
Search tree with redundant constraints

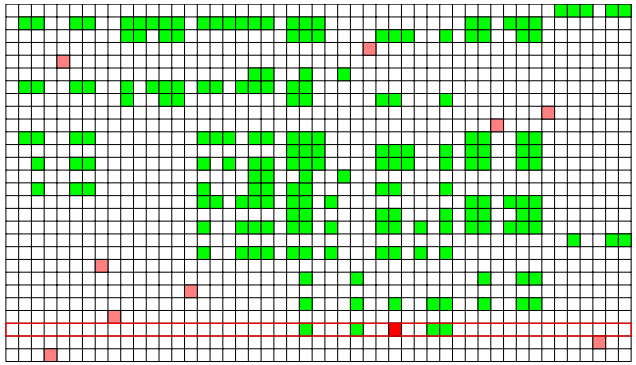




Back to Start Skip Animation

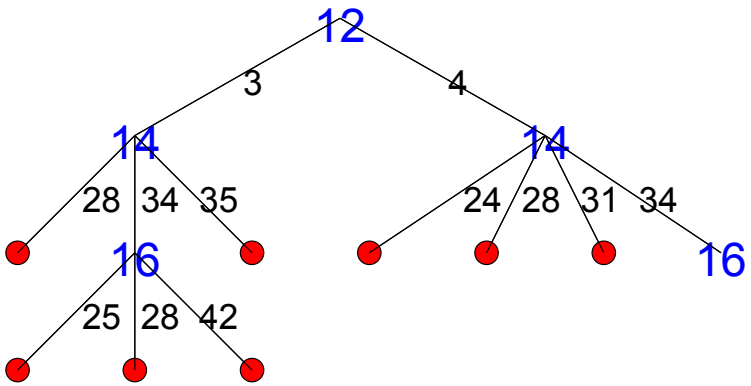
Search tree with redundant constraints

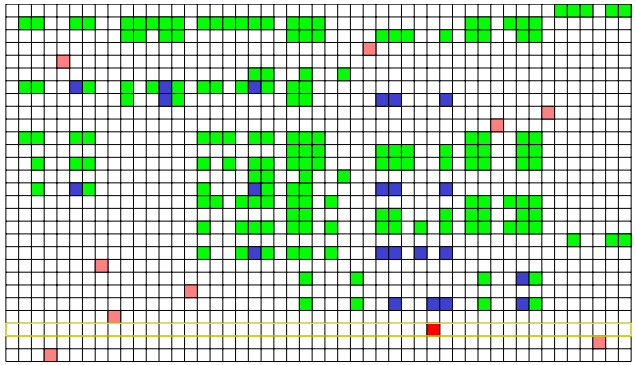




Back to Start Skip Animation

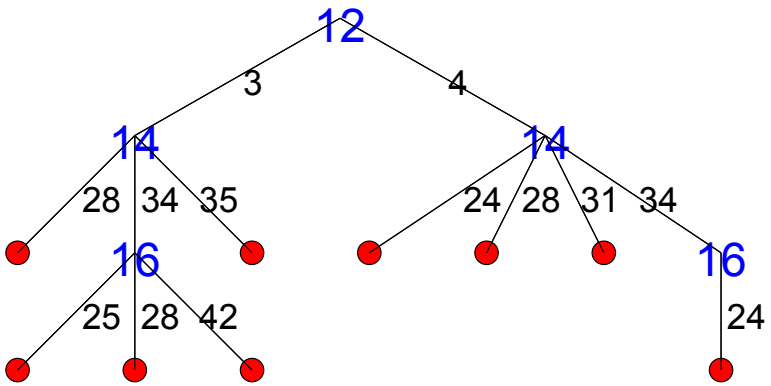
Search tree with redundant constraints

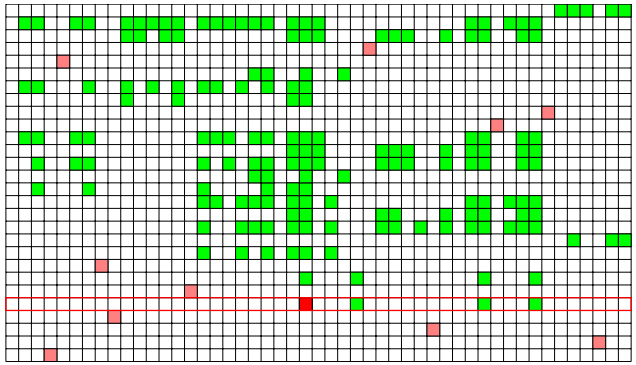




Back to Start Skip Animation

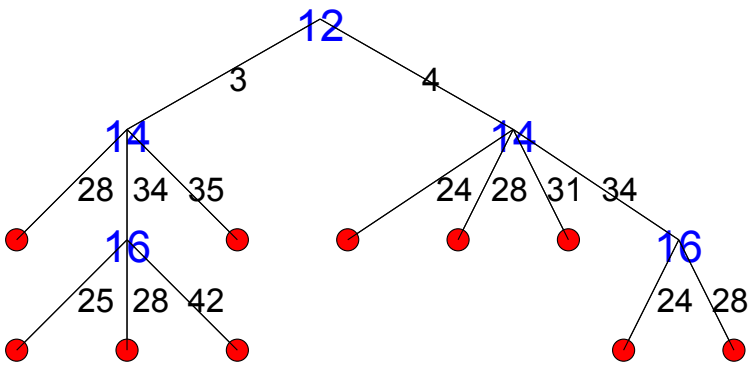
Search tree with redundant constraints

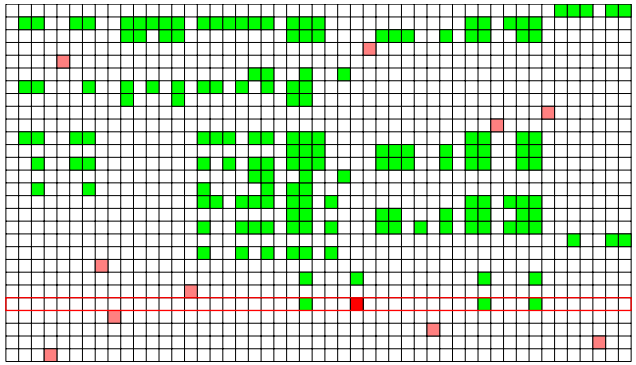




Back to Start Skip Animation

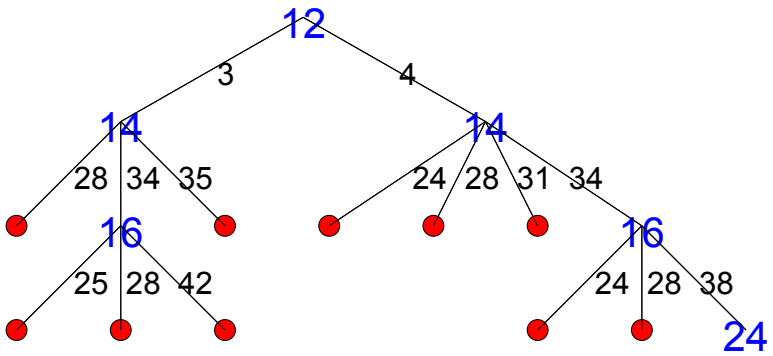
Search tree with redundant constraints

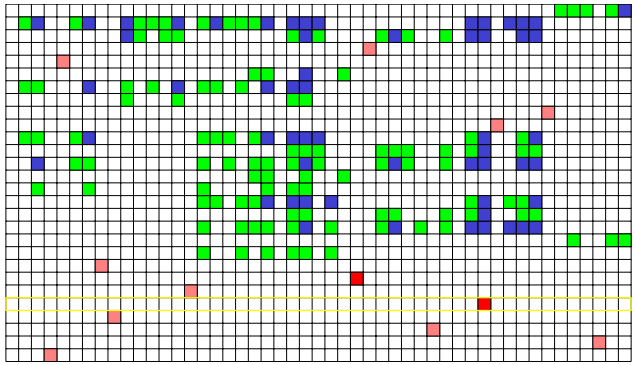




Back to Start Skip Animation

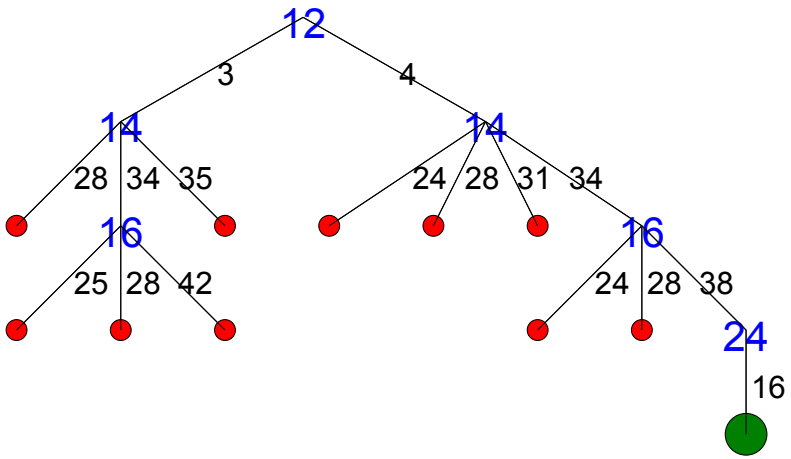
Search tree with redundant constraints

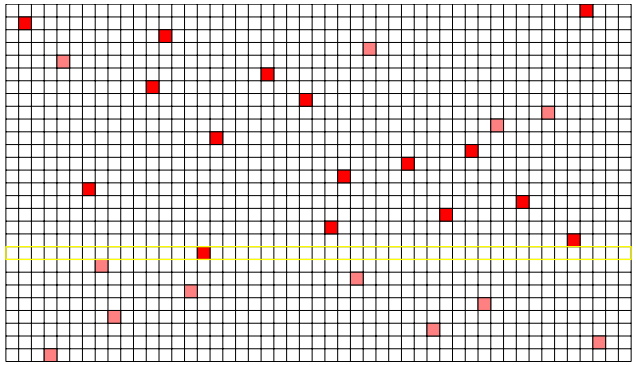




Back to Start Skip Animation

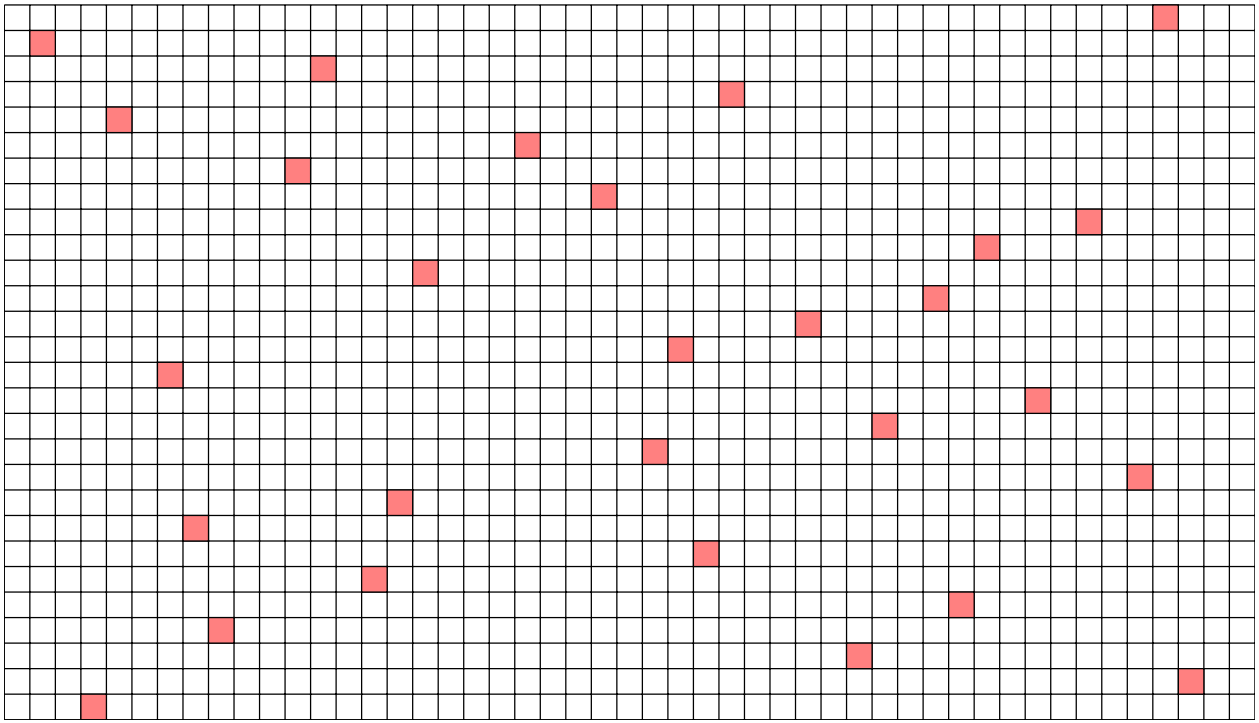
Search tree with redundant constraints



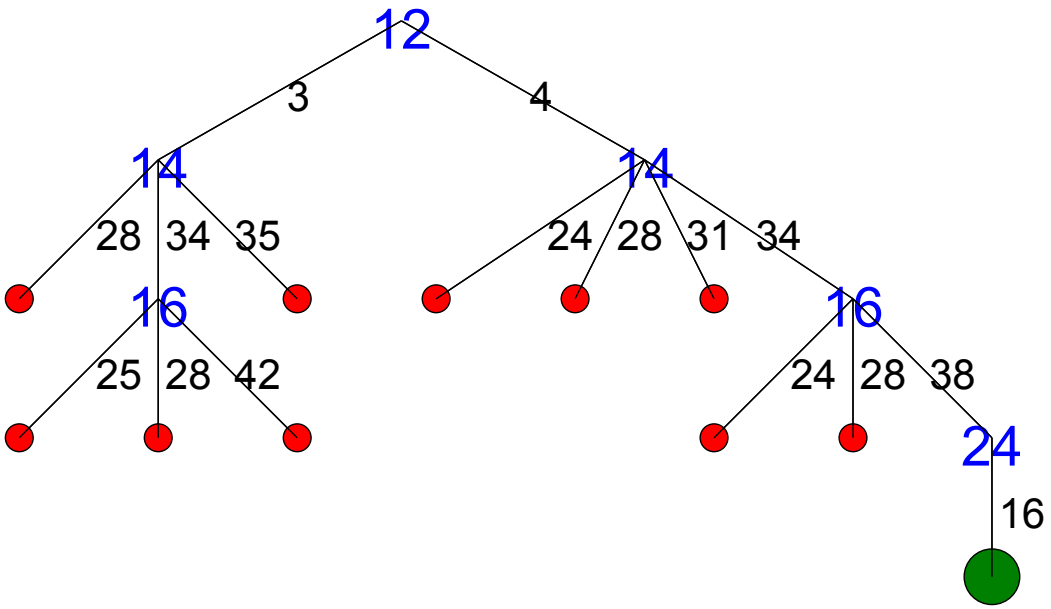


Back to Start

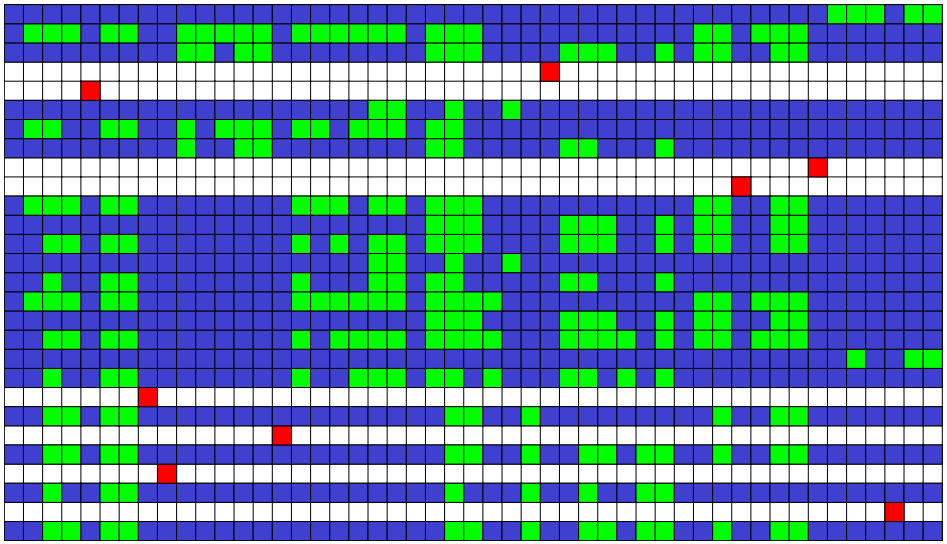
Solution



Search Tree



Still Missing Propagation

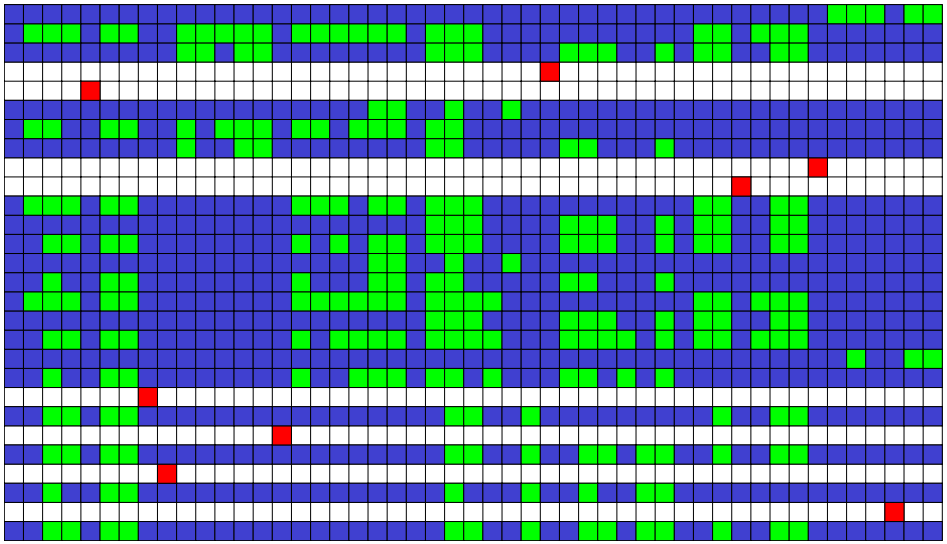


	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

Game 12 can not be played on day 1 at locations 5 or 6

Still Missing Propagation

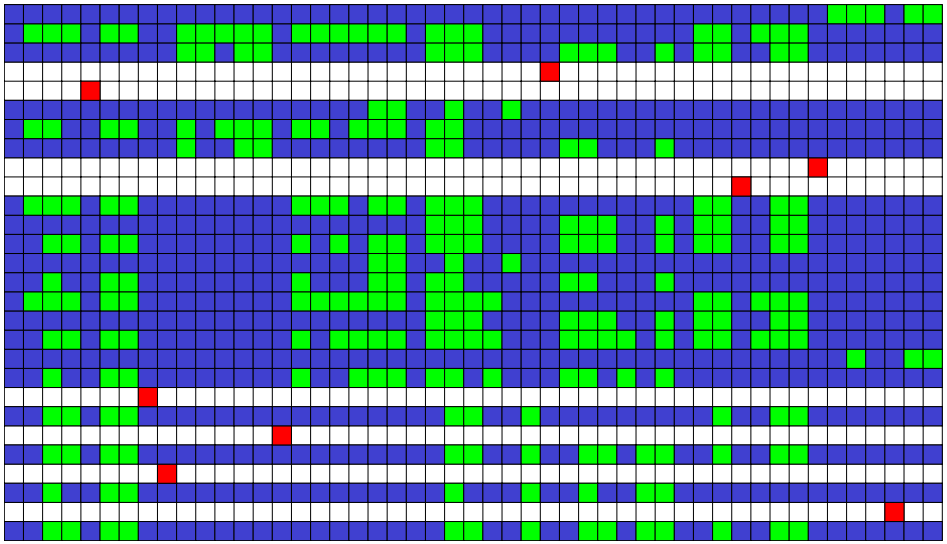


	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

Game 12 can not be played on day 1 at locations 5 or 6

Still Missing Propagation

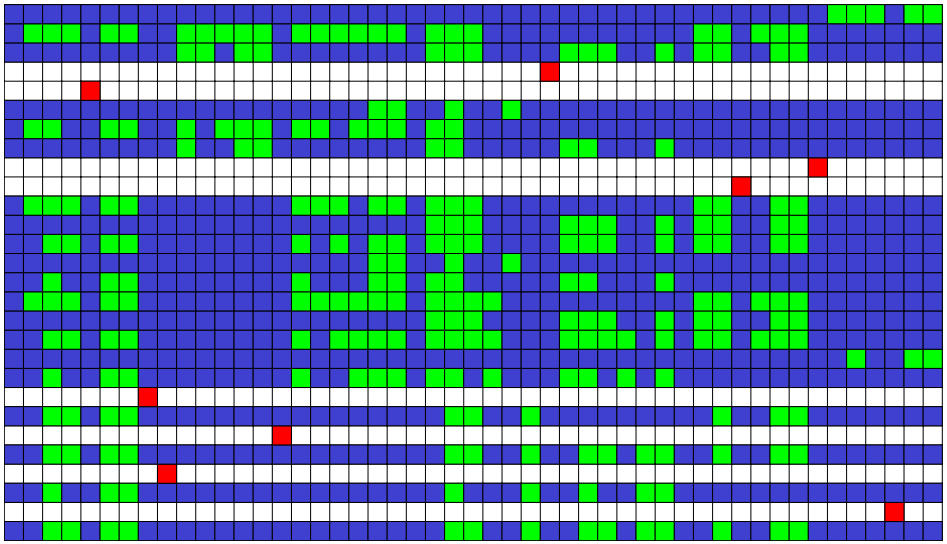


	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

Game 12 can not be played on day 1 at locations 5 or 6

Still Missing Propagation

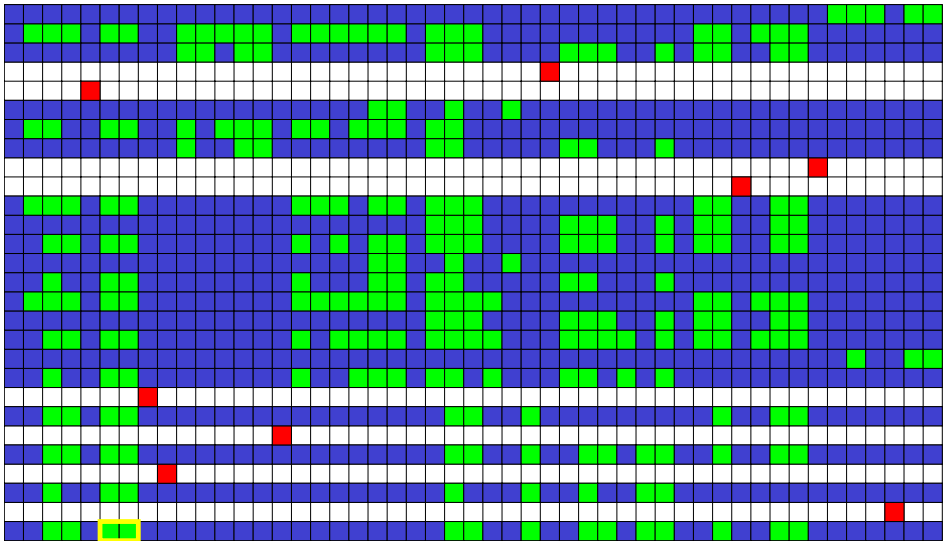


	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

Game 12 can not be played on day 1 at locations 5 or 6

Still Missing Propagation

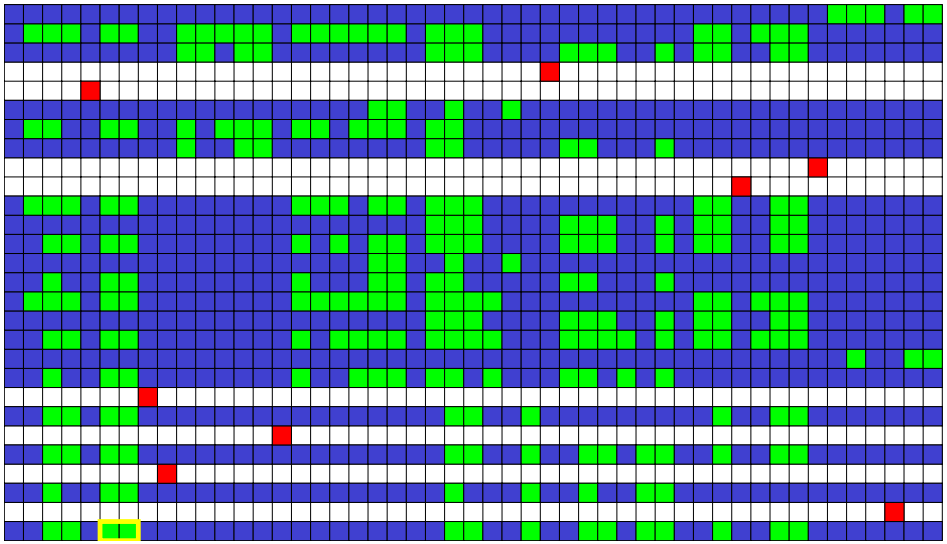


	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

Game 12 can not be played on day 1 at locations 5 or 6

Still Missing Propagation



	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1	1	2	3	4	5	6	7
Day 2	8	9	10	11	12	13	14
Day 3	15	16	17	18	19	20	21
Day 4	22	23	24	25	26	27	28
Day 5	29	30	31	32	33	34	35
Day 6	36	37	38	39	40	41	42
Day 7	43	44	45	46	47	48	49

Game 12 can not be played on day 1 at locations 5 or 6

Our model does not deal well with hints

- Preset game is ok, leads to variable assignment
- Preset team is weak, adds new constraint
- As there is no interaction of this constraint with the other constraints, there is no initial domain restriction
- Model is correct, but lazy

Second Attempt: Improving the handling of hints

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

- This value can not be used by pairs not involving team 8
- One of the pairs involving team 8 must use this value
- These values can not be used by any pair involving team 8

Second Attempt: Improving the handling of hints

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

- This value can not be used by pairs not involving team 8
- One of the pairs involving team 8 must use this value
- These values can not be used by any pair involving team 8

Second Attempt: Improving the handling of hints

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

- This value can not be used by pairs not involving team 8
- One of the pairs involving team 8 must use this value
- These values can not be used by any pair involving team 8

Second Attempt: Improving the handling of hints

	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

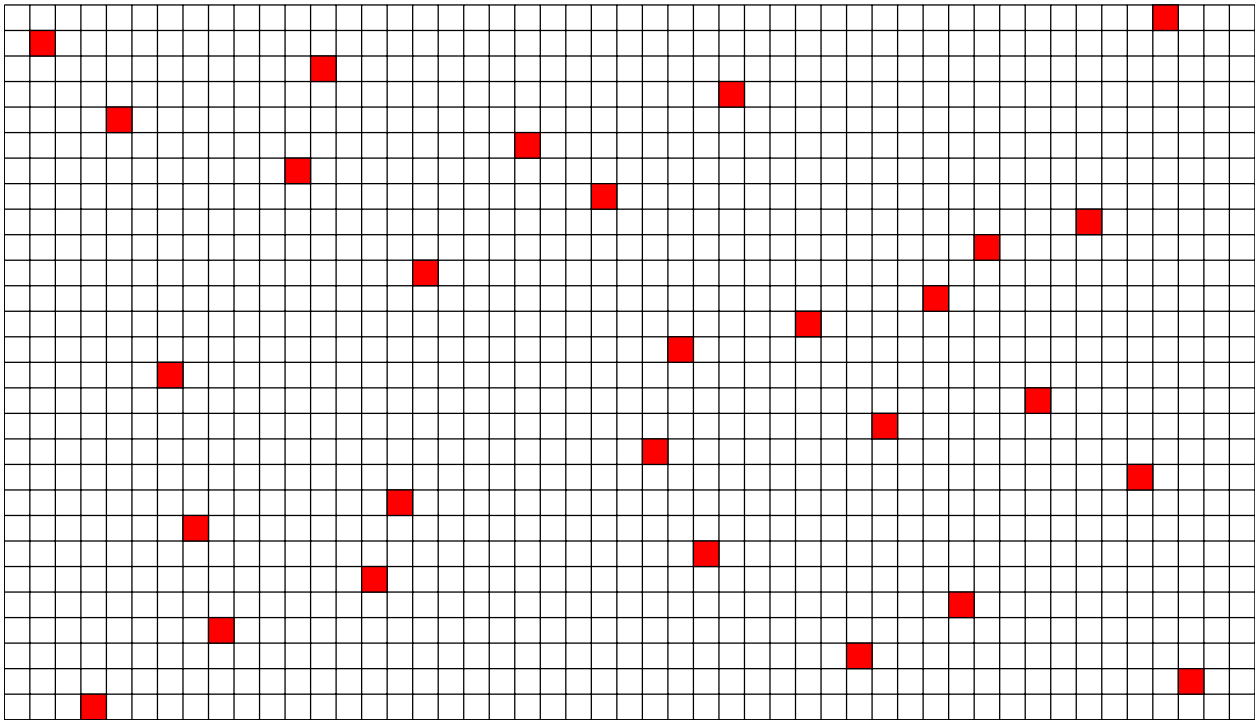
- This value can not be used by pairs not involving team 8
- One of the pairs involving team 8 must use this value
- These values can not be used by any pair involving team 8

Redundant Constraints

- Red value can not be used by pairs not involving team 8
  - disequalities
- One of the pairs involving team 8 must use red value
  - occurrences(gcc) constraint
- Yellow values can not be used by any pair involving team 8
  - disequalities

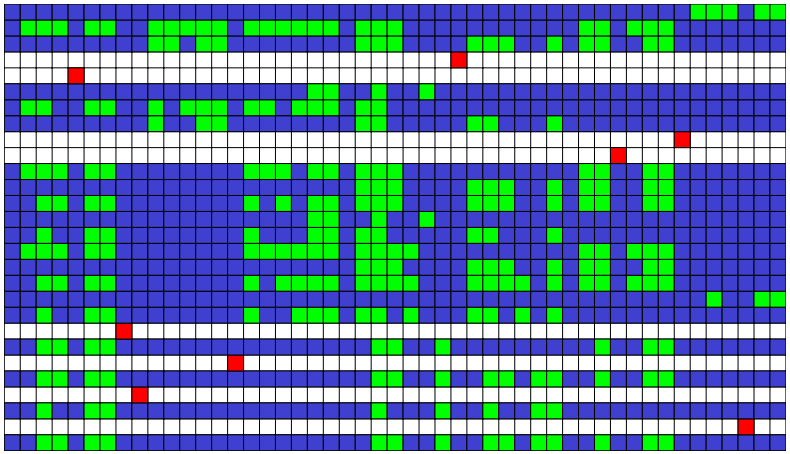
	City 1	City 2	City 3	City 4	City 5	City 6	City 7
Day 1		8			7, 5		
Day 2	2	1, 5					
Day 3	7		8				
Day 4					2	5	1
Day 5	8					1	
Day 6				5, 4			
Day 7	4				1, 3		

Before Search

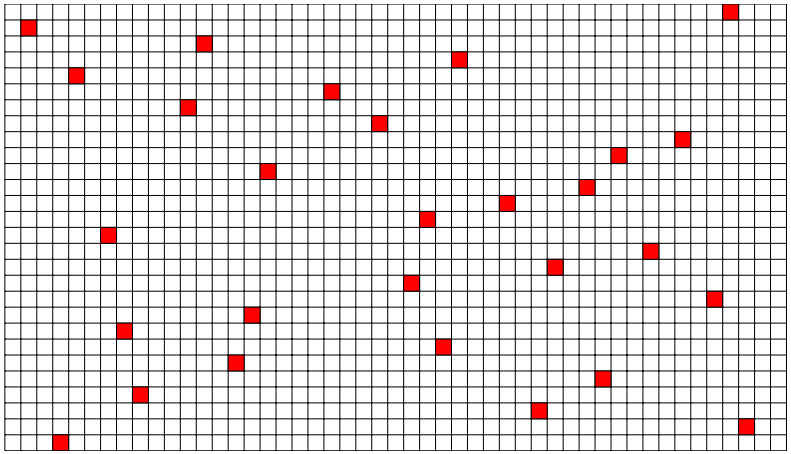


Impact of improved hint handling

With index set channeling



Improved Hints



Conclusions

- Many ways of modelling even simple problems
- Selection of “best” model difficult
  - Depends on constraints available
  - Often needs experimentation
- How do we measure if one model is “better” than another?
  - Execution time?
  - Size of search tree?
  - Scalability?
- Definition of variables is key
- Explore choices by considering mapping operators

Outline

Why CP-Viz?


Viz
YOU DON'T HAVE TO LIKE IT,
YOU JUST HAVE TO BUY IT
Latest Issue
Regulars
Features
Buy Stuff etc

Why Visualization?

- Constraint Programming = Declarative Programming ?
- If they work, they are easy to understand
- What to do if they don't work?
- Visualization shows what is happening
  - At the right level of abstraction
  - In terms a user can understand

Previous Work

- Sepia
- GRACE
- Oz-Explorer
- DISCiPl (COSYTEC, PrologIA, Madrid, INRIA,...)
- OADymPPaC (COSYTEC, ILOG, INRIA, EMN, IRISA, LIFO)

Problems

- Tools either very system specific (nearly all existing systems)
- or too generic (OADymPPaC)
- Required: Light weight, generic visualization tool

Features

- Detailed or compact tree view
- Views of state after each choice
- Specialized visualizers for different global constraints
- State, path, (tree) and evolution views
- Invariant checking
- No view of internals of propagation
- No view of constraint network

Background

- Initially developed for ECLiPSe
- Funded by Cisco through gift grant
- ECLiPSe ELearning Course
- Intended reuse for multiple systems

How do we understand behavior?

- Mental model
- Formal analysis
- Debugging
- Tracing
- Life visualization
- Post-mortem analysis

How do we understand behavior?

- Mental model
- Formal analysis
- Debugging
- Tracing
- Life visualization
- Post-mortem analysis

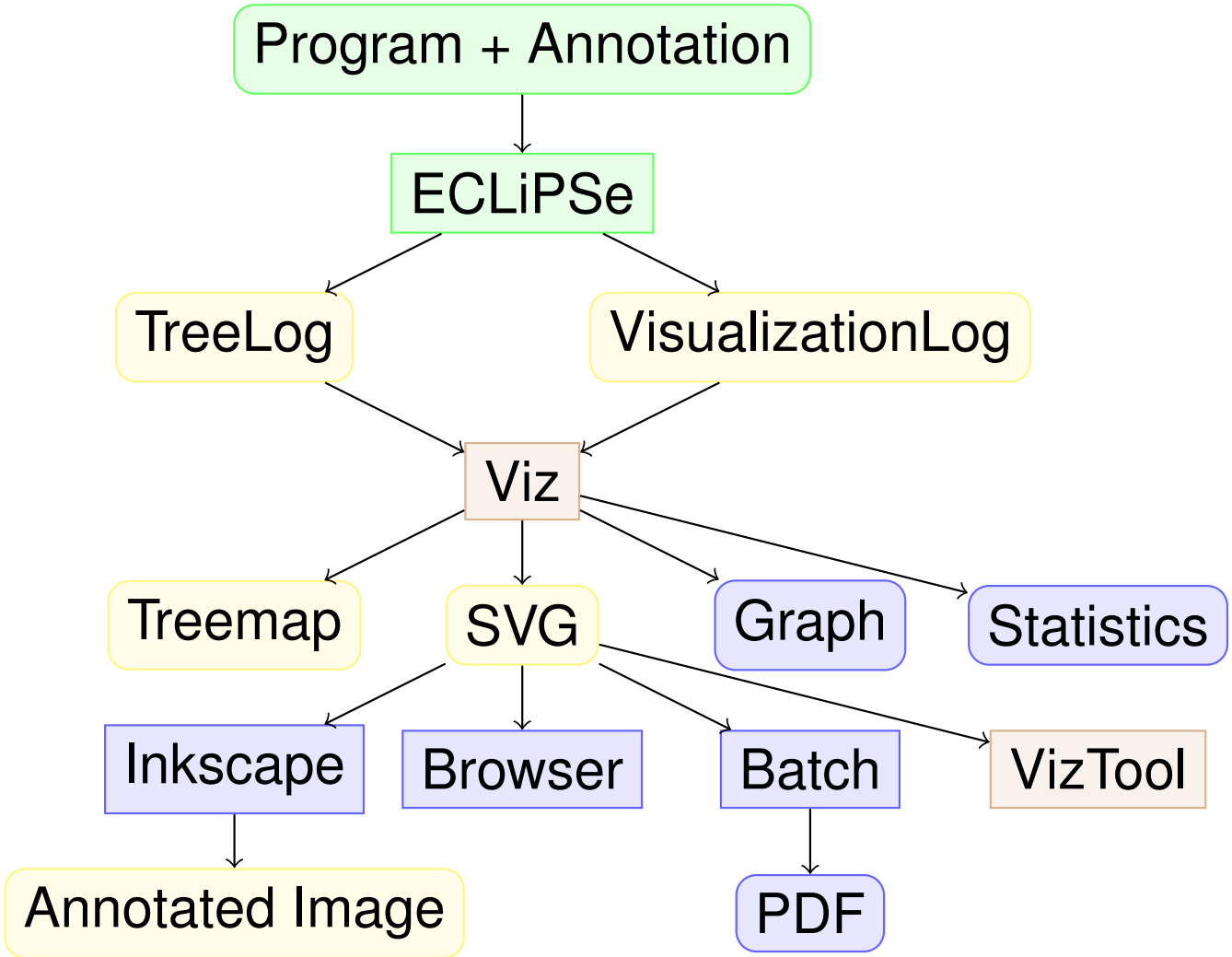
Why Visualize?

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

Conceptual Model

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

Architecture



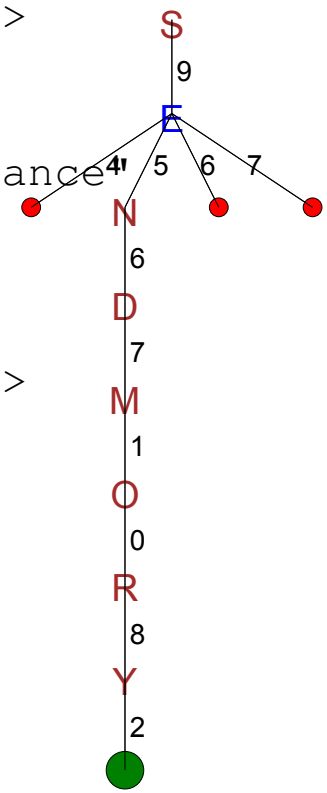
Outline

Log File Example

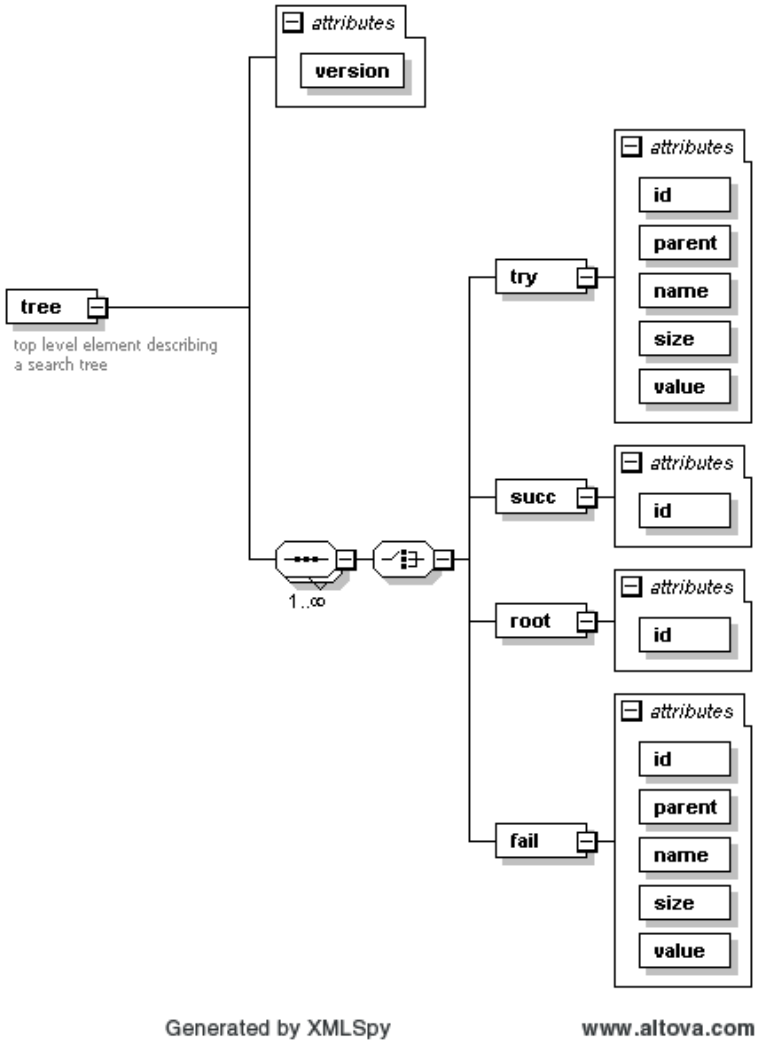
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!- Helmut Simonis (University College Cork) ->
<tree version="1.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="tree.xsd">
  <root id="0"/>
  <try id="1" parent="0" name="S" size="1" value="9"/>
  <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>
```

Log File Example

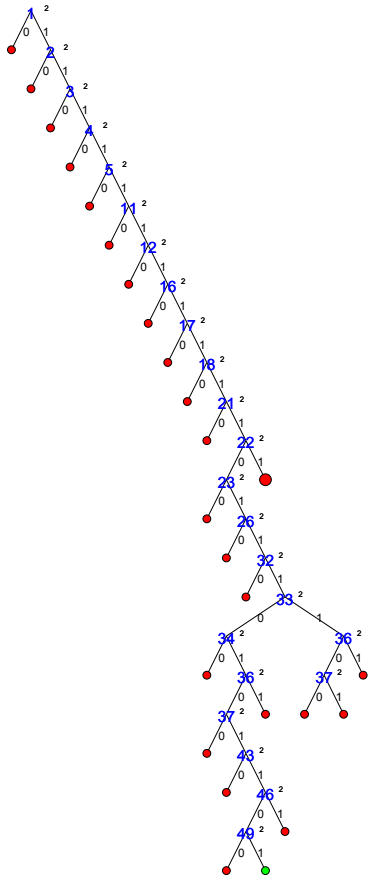
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!- Helmut Simonis (University College Cork) ->
<tree version="1.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="tree.xsd">
  <root id="0"/>
  <try id="1" parent="0" name="S" size="1" value="9"/>
  <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>
```



Search Tree Schema



Example Search Tree Output

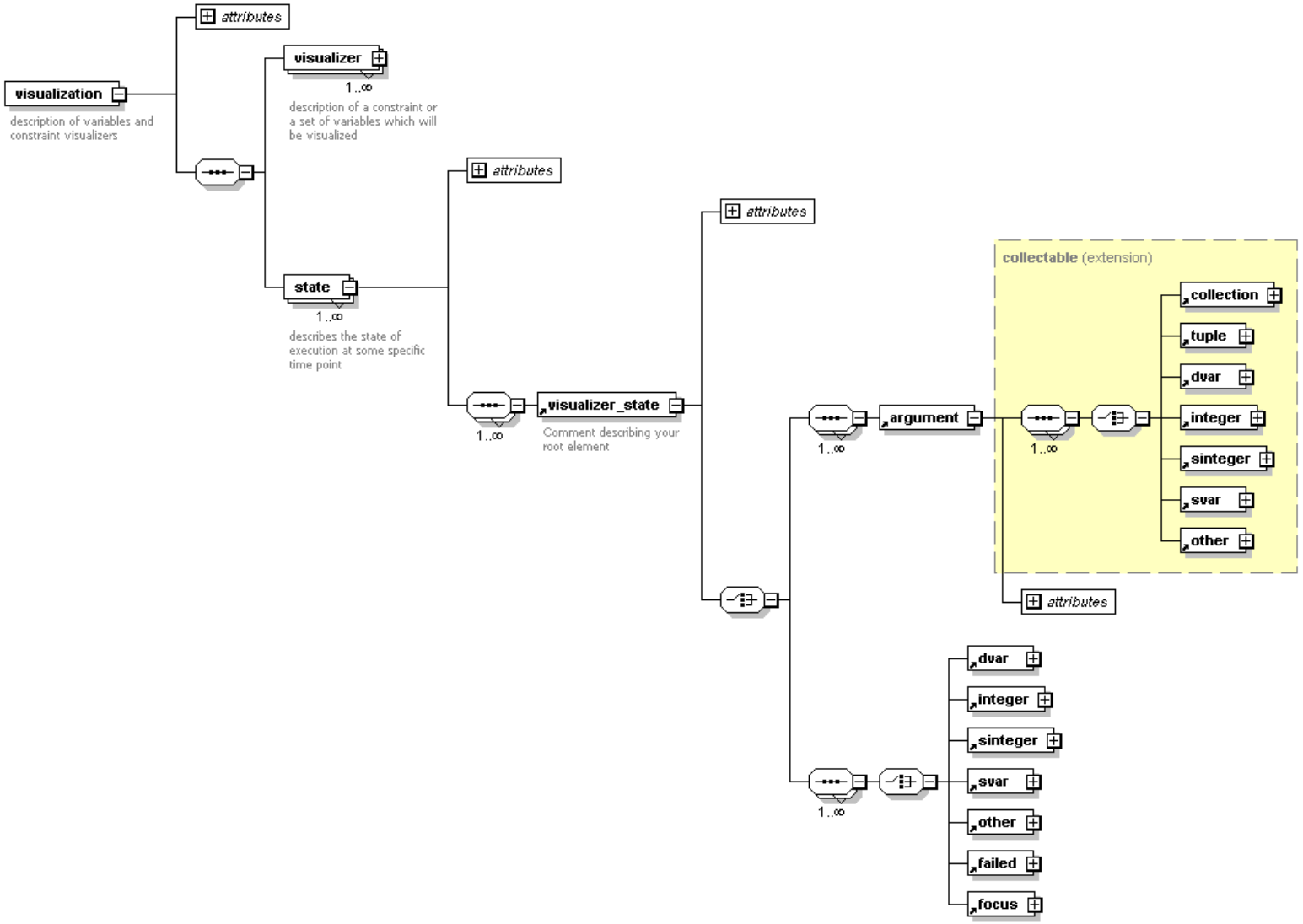


Outline

Example Log for Cumulative Constraint

```
</visualizer_state>
<visualizer_state id="6" >
<argument index="tasks" >
<tuple index="1" >
<dvar index="start" domain="1 .. 8" />
<integer index="dur" value="1" />
<integer index="res" value="1" />
</tuple>
<tuple index="2" >
<dvar index="start" domain="1 .. 8" />
<integer index="dur" value="1" />
<integer index="res" value="1" />
</tuple>
<tuple index="3" >
<dvar index="start" domain="1 .. 8" />
<integer index="dur" value="1" />
<integer index="res" value="1" />
</tuple>
<tuple index="4" >
<dvar index="start" domain="1 .. 8" />
<integer index="dur" value="1" />
```

Visualization Log Schema

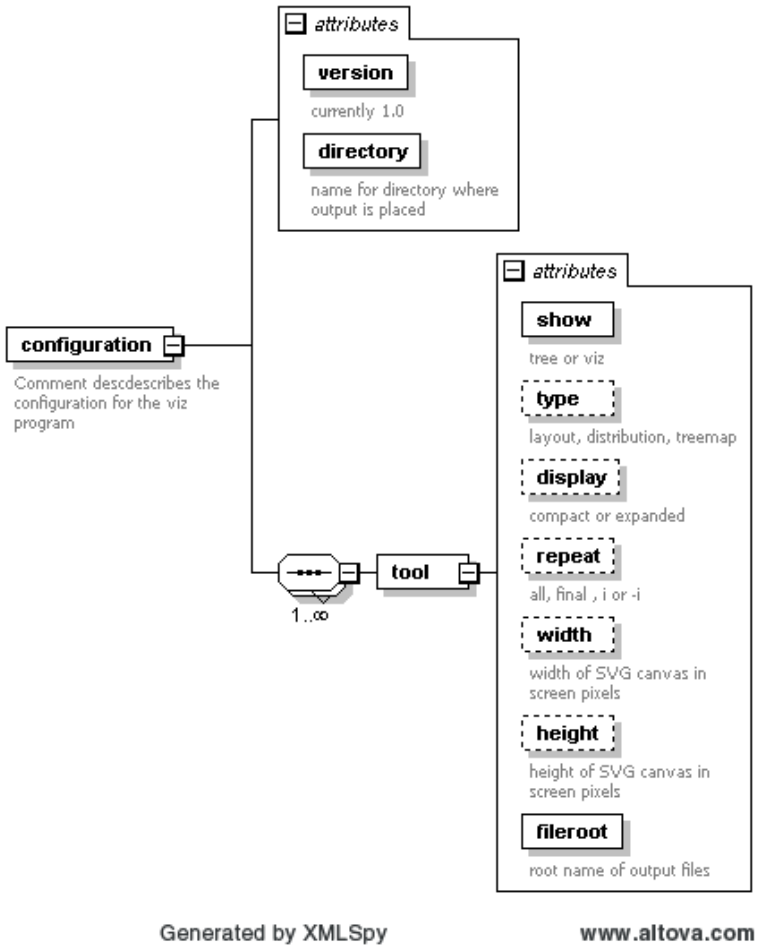


Outline

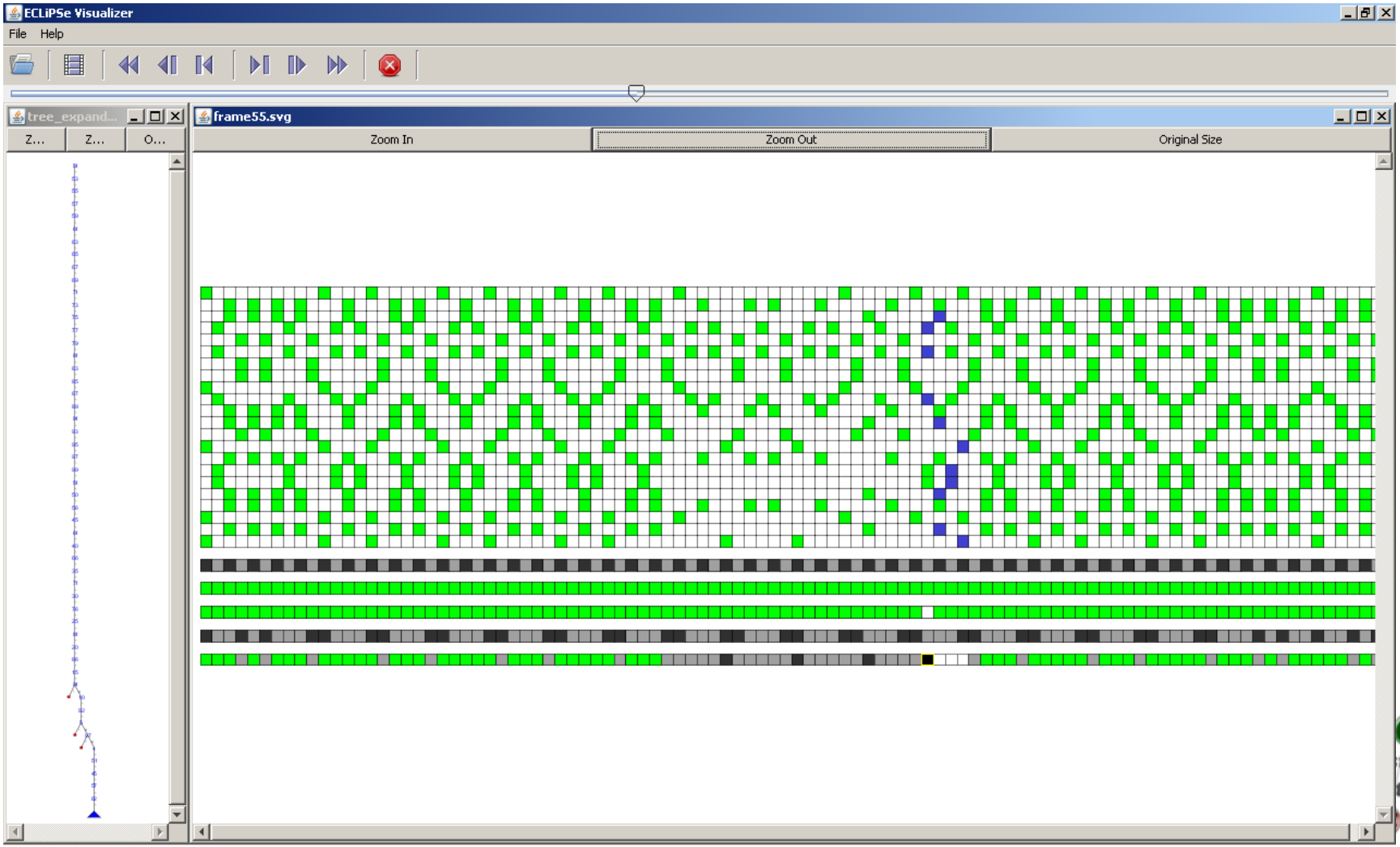
Configuration Sample File

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2010 (http://www.altova.com)-
<configuration version="1.0" directory="examples/mix/RESULT"
               xsi:noNamespaceSchemaLocation="configuration.xsd"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<tool show="tree" type="layout" display="expanded" repeat="all"
             width="700" height="700" fileroot="tree"  />
<tool show="viz" type="layout" display="compact" repeat="final"
             width="900" height="900" fileroot="viz"  />
</configuration>
```

Configuration Schema



VizTool: Car Sequencing



How to Interpret Visualization

- Search tree
  - Good/bad choices
  - Place of backtracking
- State
  - Missing propagation

Background

- Provided as challenge by Roberto Nieuwenhuis, Barcelona
- Production problem in steel industry
- Modelled with SMT (Barcelogic)
- They could not find optimal solution with CP

Problem

- Jobs to be processed (different types, multiple instances per type)
- Nearly flow shop: Loading, Oven, Cooling
- 2 Ovens, 1 Cooling unit
- 3 activities at same time (all types)
- No waiting between loading, oven and cooling
- Minimize makespan

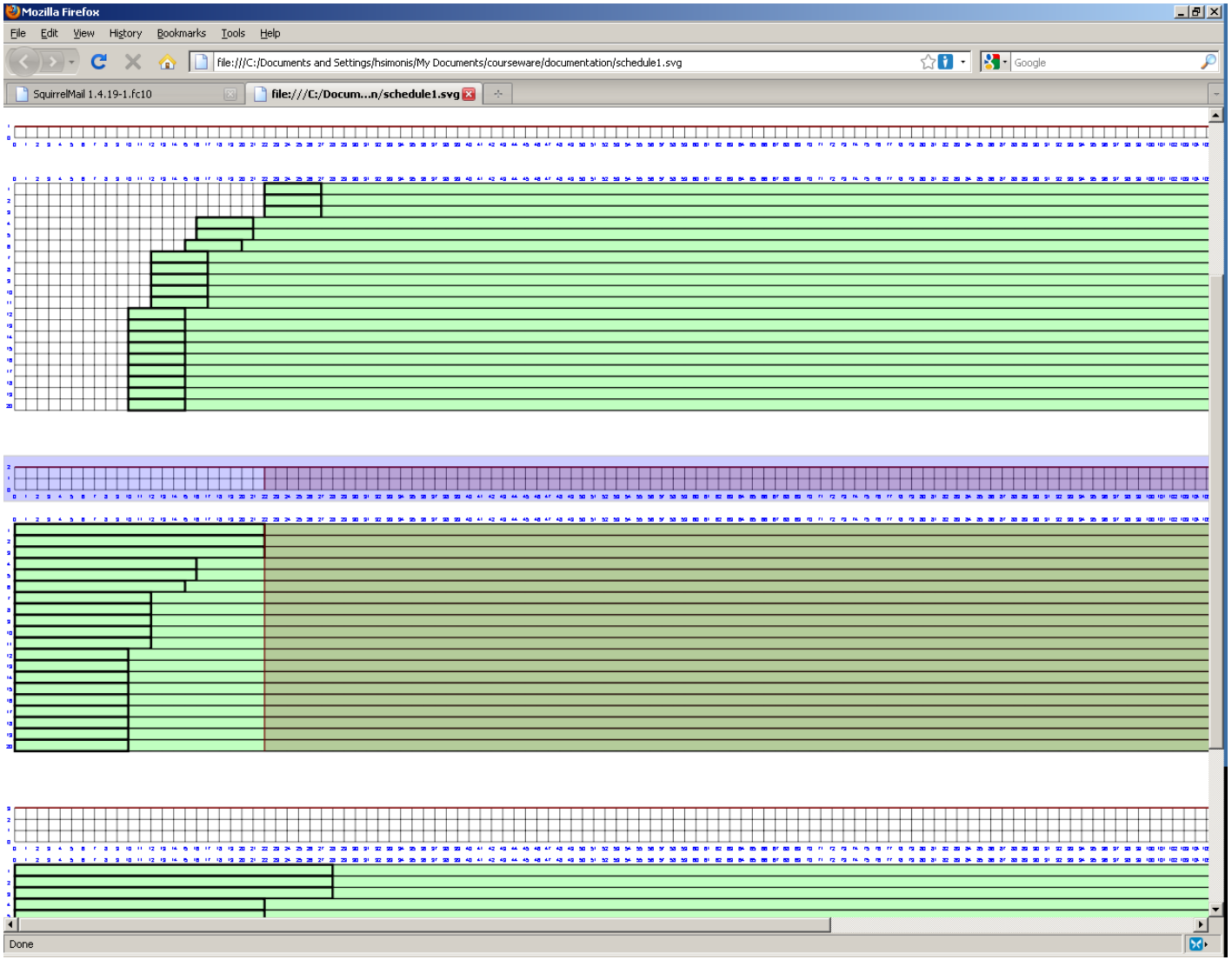
Model

- One variable per job, *Start* time in oven
- Three cumulative constraints: Ovens, Cooling, All
- Overall *End* is max of all job ends
- Try and find solution with optimal makespan (imposed)

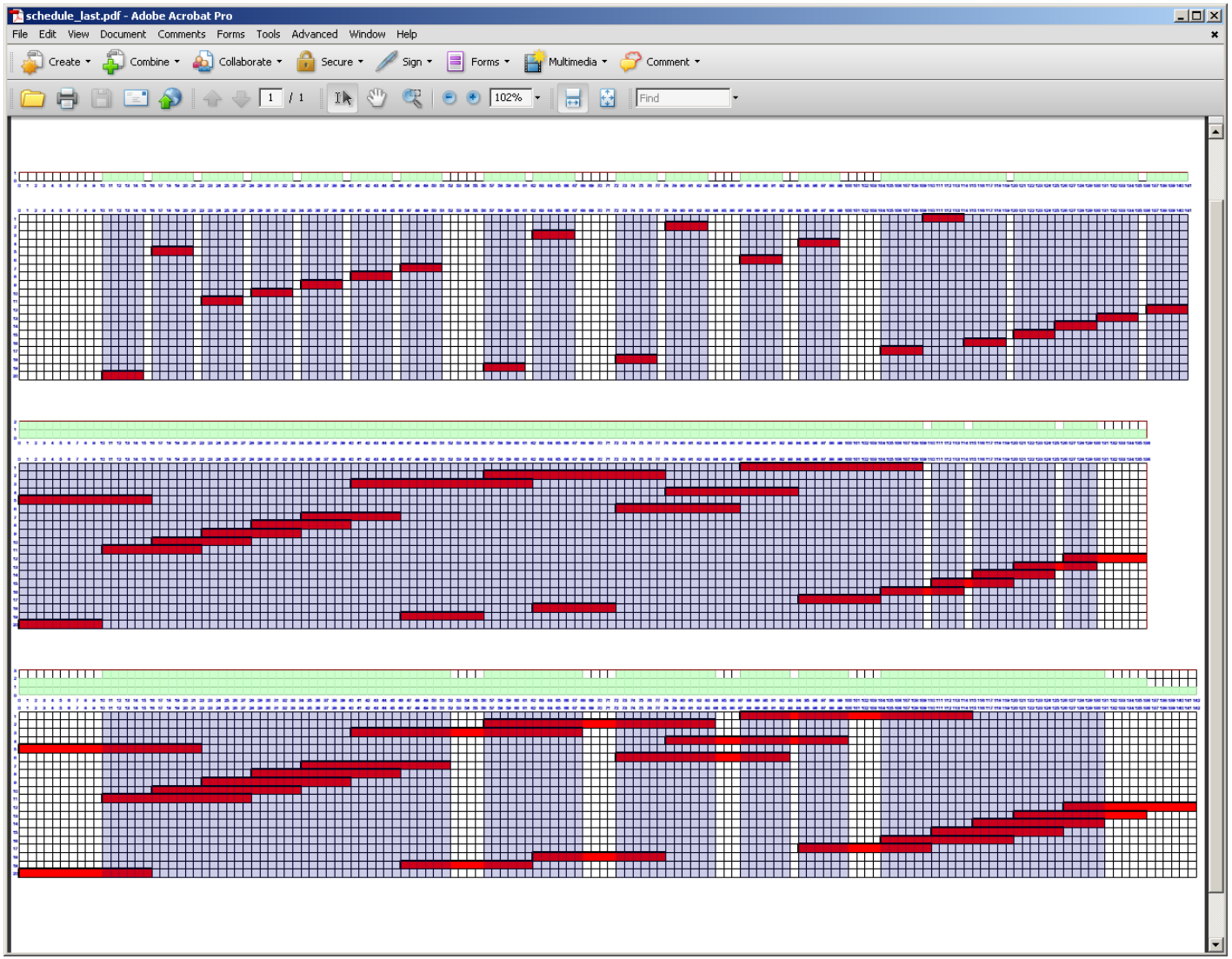
Search

- Standard variable assignment hopeless
- 20 Jobs, 150 time points
- Custom routine:
  - Select job (non-determ)
  - Fix start as early as possible (first feasible value)

Initial State



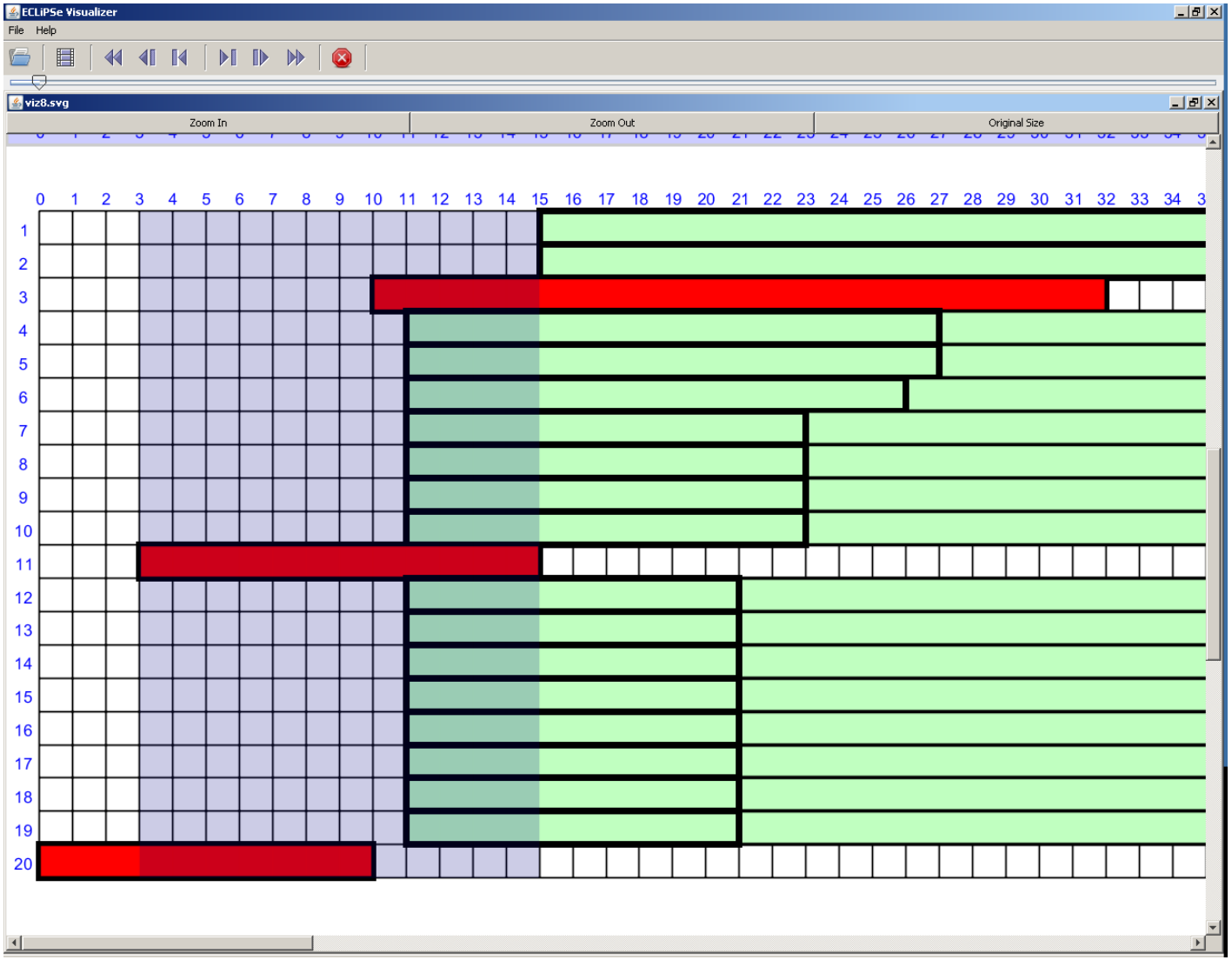
Solution Found (80 seconds)



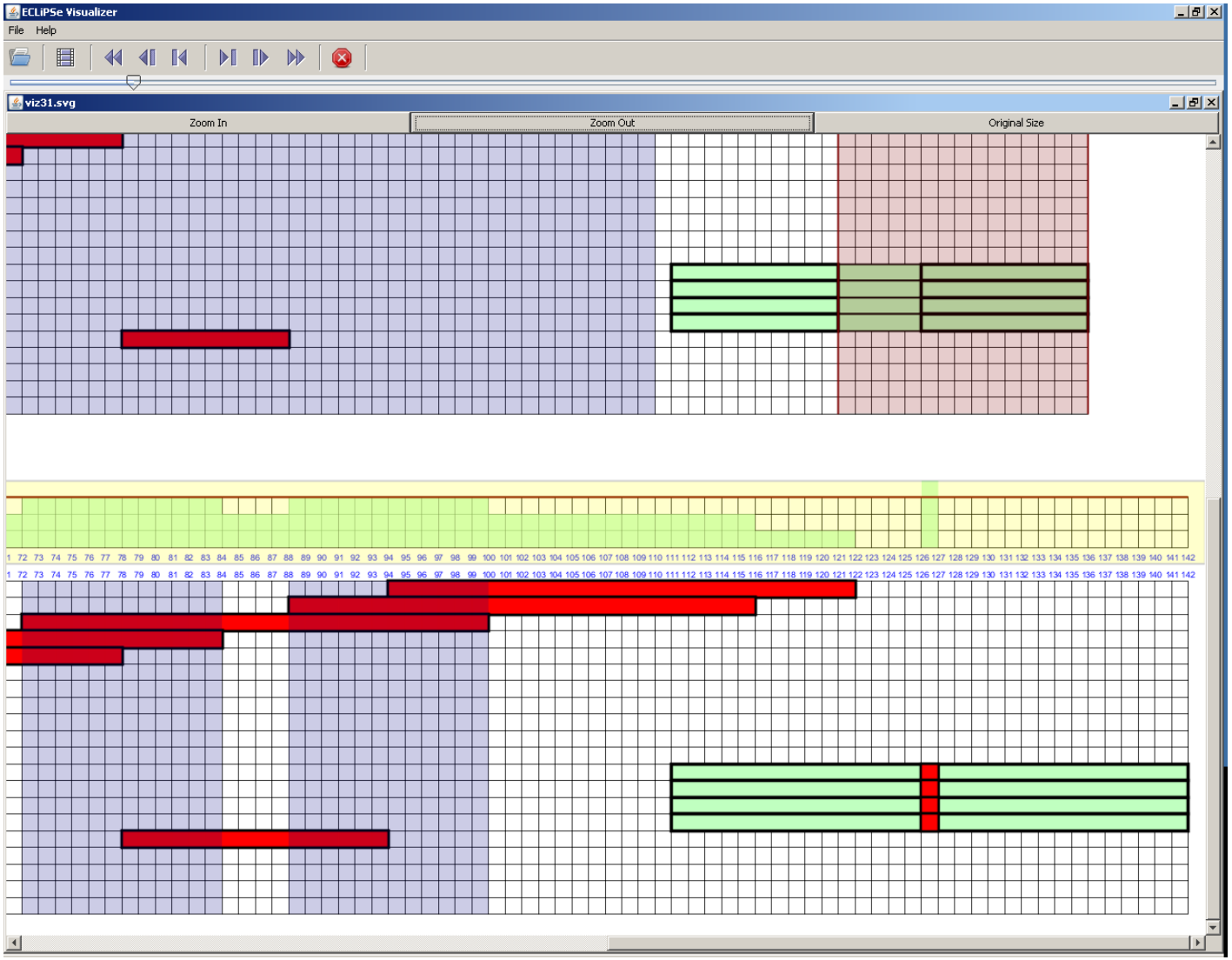
Observation

- Initial estimated *End* pathetic
- No symmetry breaking
- Ovens very near capacity, except at very end
- Cooling resource has spare capacity, but imposes constraints

Problems with `cumulative` Implementation



Problems with `cumulative` Implementation



Need for Invariant Checking

- ECLiPSe cumulative does not do a good job
- Written in C, unchanged since 199x
- Slower than state-of-the-art, but why?
- Only implements edge finding, no obligatory part reasoning

What we know about cumulative

$$\forall t : \sum_{i \text{ s.t. } s_i \leq t < s_i + d_i} r_i \leq L$$

$$E = \max(s_k + d_k)$$

$$\sum d_i * r_i \leq L * E$$

Invariant 1

$$\sum d_i * r_i \leq L * E$$

$$E \geq \frac{\sum d_i * r_i}{L^+}$$

$$E^+ < \frac{\sum d_i * r_i}{L^+} \Rightarrow \textbf{inconsistent}$$

$$E^- < \frac{\sum d_i * r_i}{L^+} \Rightarrow \textbf{missing prop}$$

Invariant 2

$$\forall t : \sum_{i \text{ s.t. } s_i \leq t < s_i + d_i} r_i \leq L$$

$$L \geq \sum_{i \text{ s.t. } s_i^+ \leq t < s_i^- + d_i} r_i$$

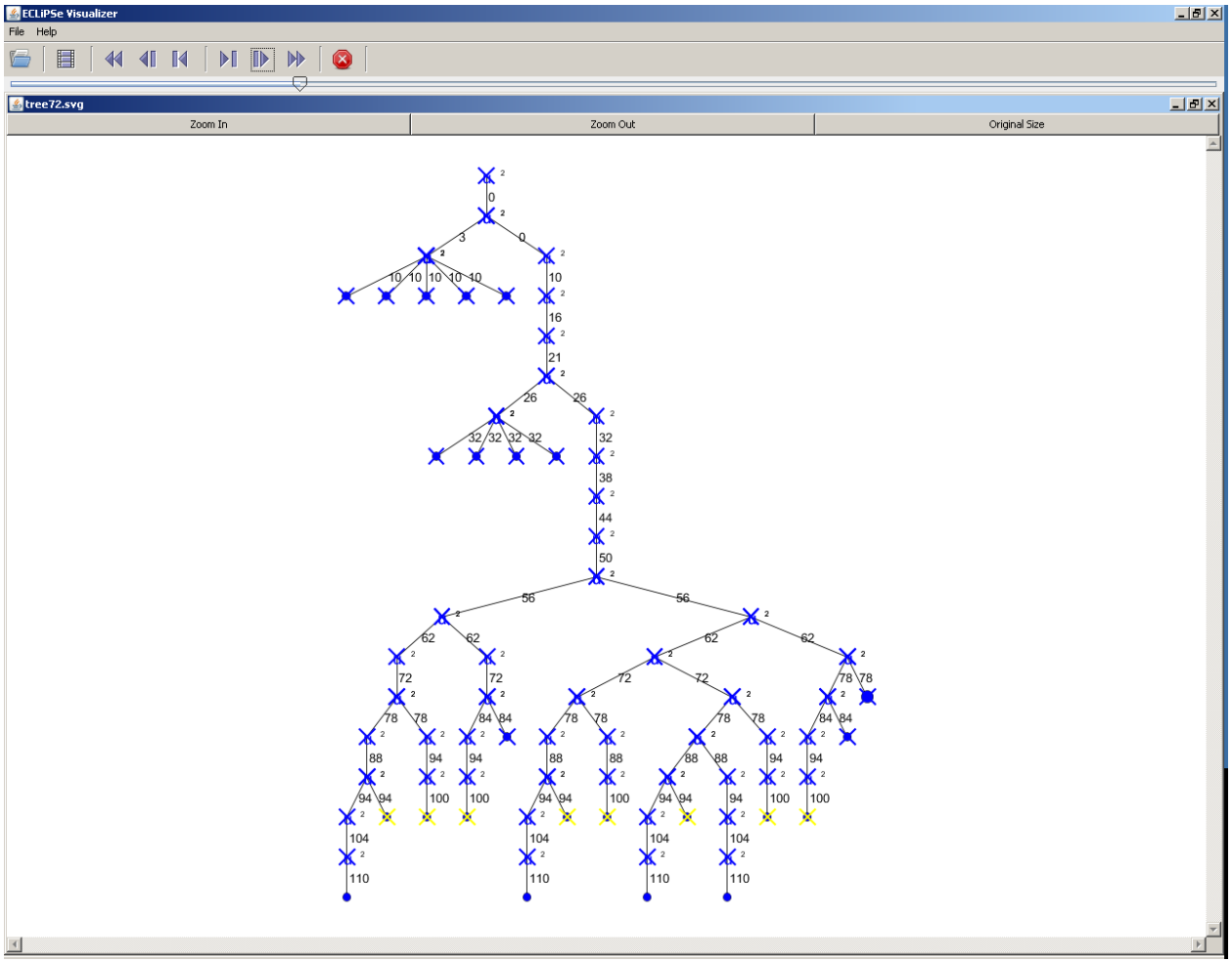
$$L^+ < \sum_{i \text{ s.t. } s_i^+ \leq t < s_i^- + d_i} r_i \Rightarrow \textbf{inconsistent}$$

$$L^- < \sum_{i \text{ s.t. } s_i^+ \leq t < s_i^- + d_i} r_i \Rightarrow \textbf{missing prop}$$

Invariant Checking

- At each state, check invariants for each constraint
- Mark violations in log, also in tree
- Different levels
  - Ground instance violates invariant: bug
  - State inconsistent: node could be pruned
  - Missing propagation: domain could be reduced

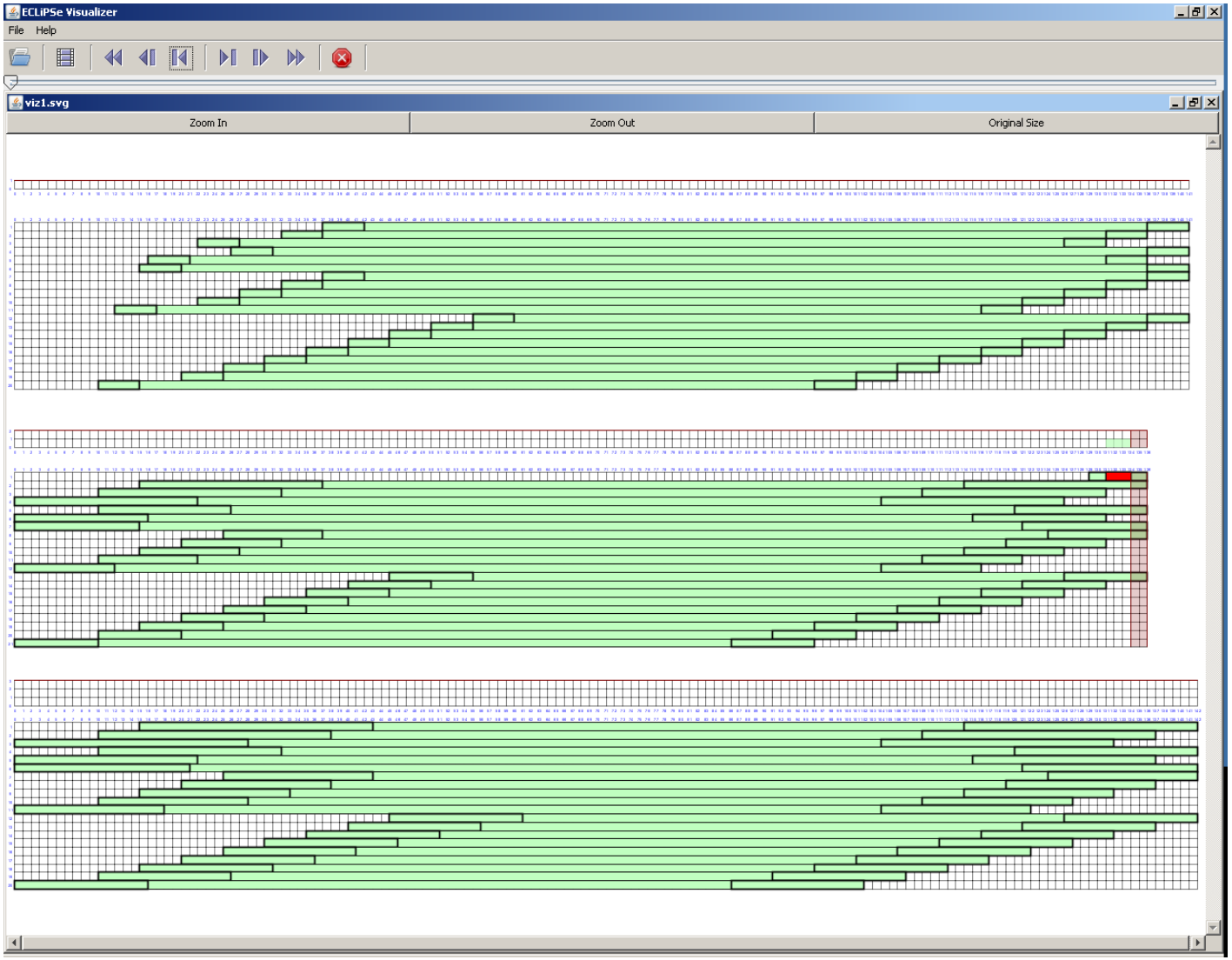
Tree with Invariant Warnings



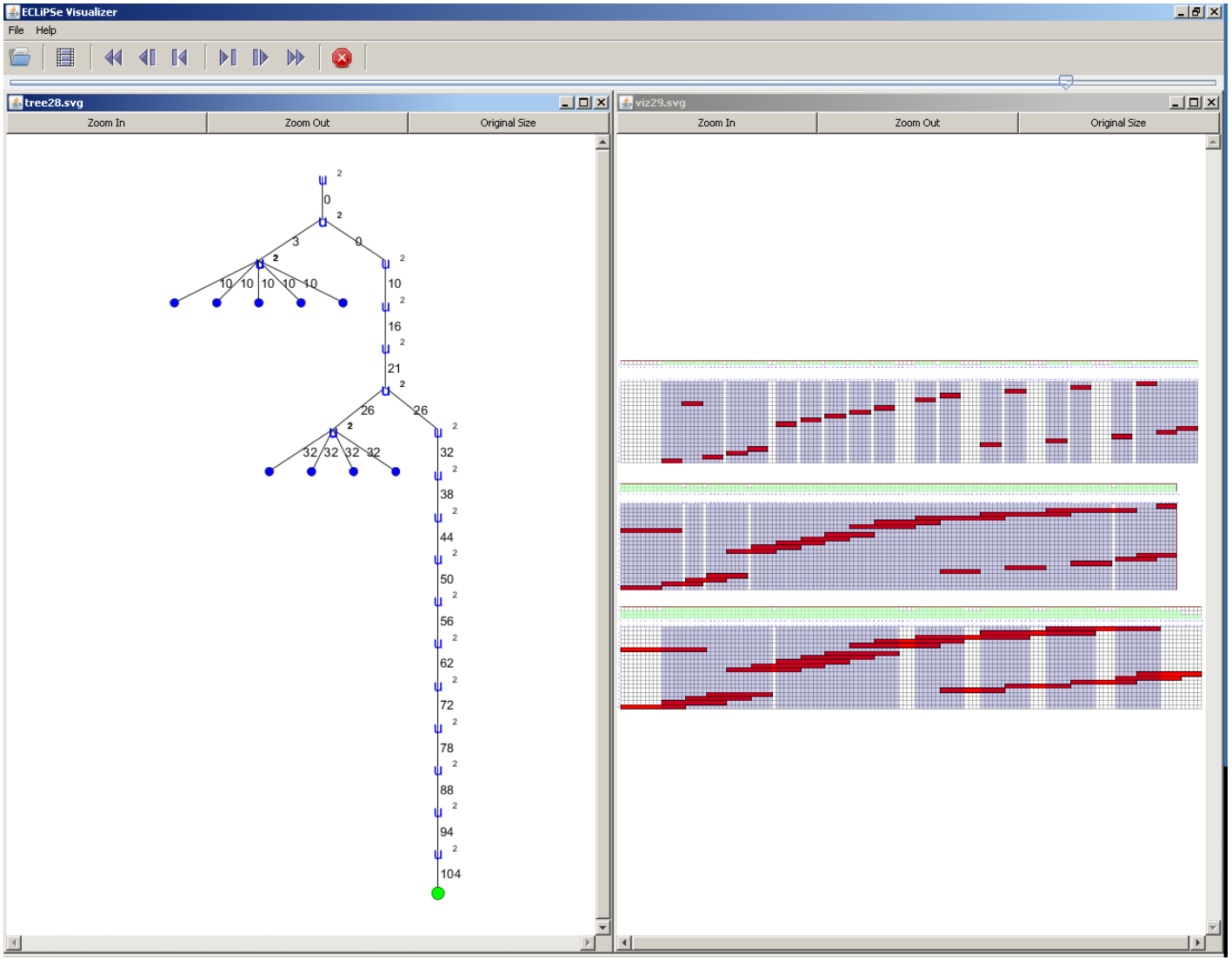
Improving the model

- Better lower bound for *End*
- Symmetry breaking by ordering jobs of same type
- Dummy oven task at end to make profile near-perfect
- Discover and fix bug in search routine
- (Add obligatory part reasoning to cumulative)
- (Change search to order cooling tasks)

Improved Model - Initial State



Improved Model - Solution Found (0.02 seconds)



Future Work

- Interfaces to other systems
- Direct Java interface without XML intermediate
- Better visualizations for some global constraints
- Derive invariants automatically
- Use Java-based constraint system to check invariants

More CP-Viz Ads