Gecode

an open constraint solving library

Christian Schulte
KTH Royal Institute of Technology, Sweden

Constraint Programming

- What is constraint programming?

Sudoku is constraint programming

...is constraint programming!

SUDOKU

Sudoku

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- Assign blank fields digits such that:
  digits distinct per **rows**, columns, blocks

Sudoku

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- Assign blank fields digits such that:
  digits distinct per rows, **columns**, blocks

Sudoku

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- Assign blank fields digits such that:
  - digits distinct per rows, columns, **blocks**

Block Propagation

	8	
	6	3

- No field in block can take digits 3,6,8

Block Propagation

1,2,4,5,7,9	**8**	1,2,4,5,7,9
1,2,4,5,7,9	**6**	**3**
1,2,4,5,7,9	1,2,4,5,7,9	1,2,4,5,7,9

- No field in block can take digits 3,6,8
  - propagate to other fields in block
- Rows and columns: likewise

Propagation

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

1,2,3,4,5,6,7,8,9

- Prune digits from fields such that:

  digits distinct per rows, columns, blocks

Propagation

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

1,3,5,6,7,8

- Prune digits from fields such that:
  digits distinct per **rows**, columns, blocks

Propagation

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

1,3,6,7

- Prune digits from fields such that:
  - digits distinct per rows, **columns**, blocks

Propagation

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

1,3,6

- Prune digits from fields such that:
  digits distinct per rows, columns, **blocks**

Iterated Propagation

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- Iterate propagation for rows, columns, blocks
- What if no assignment: search... later

Sudoku is Constraint Programming

			2		5			
	9					7	3	
		2			9		6	
2						4		9
				7				
6		9						1
	8		4			1		
	6	3					8	
			6		8			

- Variables: fields
  - take values: digits
  - maintain set of possible values
- Constraints: distinct
  - relation among values for variables

- Modeling: variables, values, constraints
- Solving: propagation, search

Constraint Programming

- Variable domains
  - finite domain integer, finite sets, multisets, intervals, ...
- Constraints
  - distinct, arithmetic, scheduling, graphs, ...
- Solving
  - propagation, search, ...
- Modeling
  - variables, values, constraints, heuristics, symmetries, ...

This Talk...

- Constraint programming
    - key concepts
    - the constraint programmer’s toolbox
- Gecode
    - some facts
    - goals & use cases
    - modeling & programming
    - openness
    - example model (if time allows)
- Summary

KEY CONCEPTS

Running Example: SMM

- Find distinct digits for letters such that

```
    SEND
+   MORE
--------
=  MONEY
```

Constraint Model for SMM

- Variables:

  `S,E,N,D,M,O,R,Y ∈ {0,…,9}`
- Constraints:

```
distinct(S,E,N,D,M,O,R,Y)

            1000×S+100×E+10×N+D
 +          1000×M+100×O+10×R+E
 = 10000×M+1000×O+100×N+10×E+Y

S≠0            M≠0
```

Solving SMM

- Find values for variables

  such that

  all constraints satisfied

Finding a Solution

- Compute with possible values
  - rather than enumerating assignments
- Prune inconsistent values
  - constraint propagation
- Search
  - branch: define shape of search tree
  - explore: explore search tree for solution

constraint store

propagators

constraint propagation

CONSTRAINT PROPAGATION

Constraint Store

$x \in \{1,2,3,4\}$ $y \in \{1,2,3,4\}$ $z \in \{1,2,3,4\}$

- Maps variables to possible values

Constraint Store

finite domain constraints

$x \in \{1,2,3,4\}$ $y \in \{1,2,3,4\}$ $z \in \{1,2,3,4\}$

- Maps variables to possible values
  - other domains: finite sets, float intervals, graphs, ...

Propagators

- Implement constraints

  distinct(x_1, …, x_n)

  $x + 2 \times y = z$

  schedule(t_1, …, t_n)

Propagators

distinct(x, y, z) $x + y = 3$

$x \in \{1,2,3,4\}$ $y \in \{1,2,3,4\}$ $z \in \{1,2,3,4\}$

- Strengthen store by constraint propagation
  - prune values in conflict with implemented constraint

Propagators

distinct(x, y, z)

$x + y = 3$

$x \in \{1,2\}$ $y \in \{1,2\}$ $z \in \{1,2,3,4\}$

- Strengthen store by constraint propagation
  - prune values in conflict with implemented constraint

Propagators

distinct(x, y, z)

$x + y = 3$

$x \in \{1,2\}$ $y \in \{1,2\}$ $z \in \{3,4\}$

- Iterate propagator execution until fixpoint
    - no more pruning possible

Propagation for SMM

- Results in store

```
S∈{9}   E∈{4,…,7}      N∈{5,…,8} D∈{2,…,8}
M∈{1}   O∈{0}          R∈{2,…,8} Y∈{2,…,8}
```

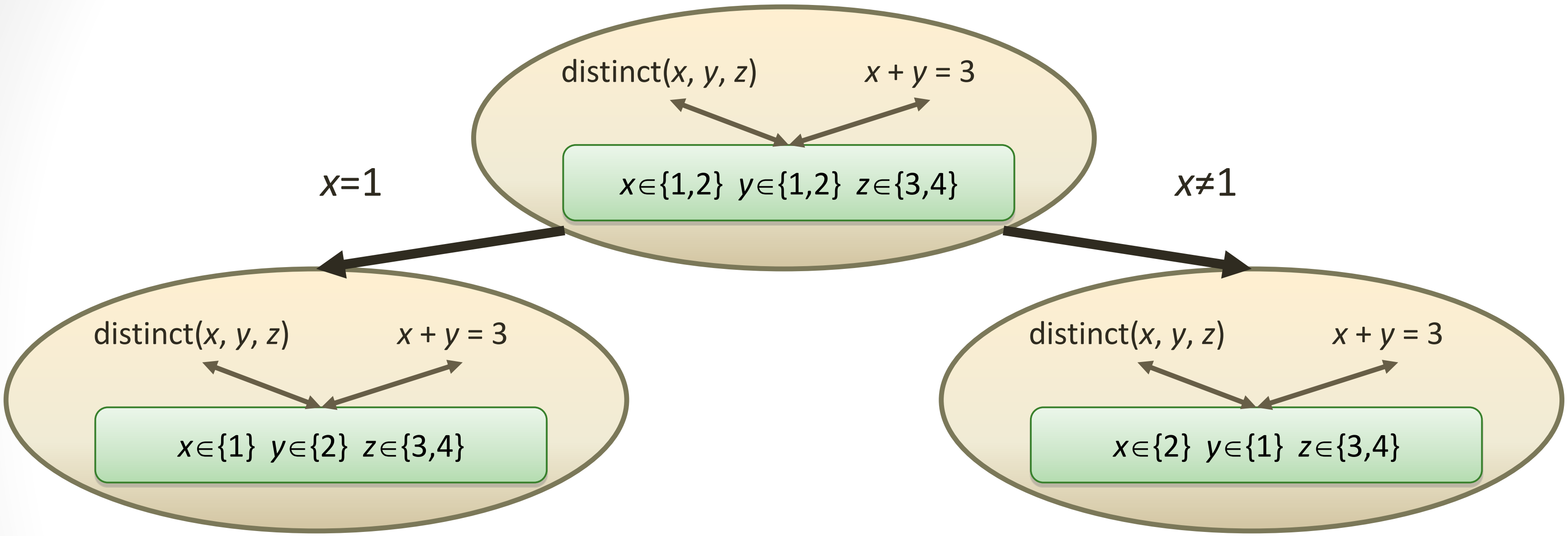
- Propagation **alone** not sufficient!
  - decompose into simpler sub-problems
  - branching and exploration for search

branching

exploration

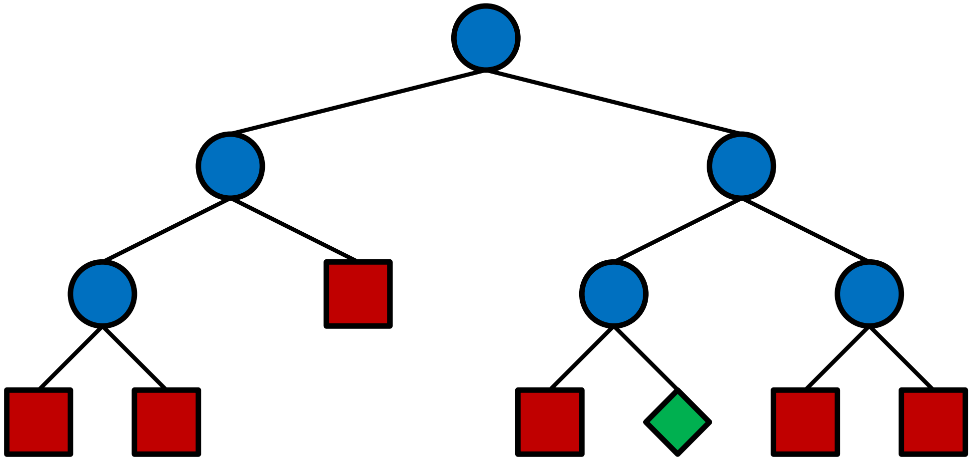
SEARCH

Branching



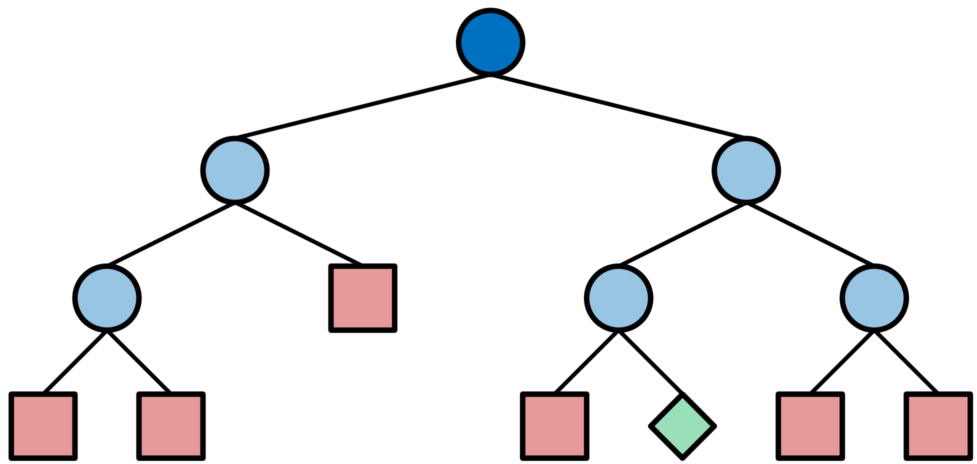
- Create subproblems with additional constraints
  - enables further propagation
  - defines search tree

Heuristic Branching



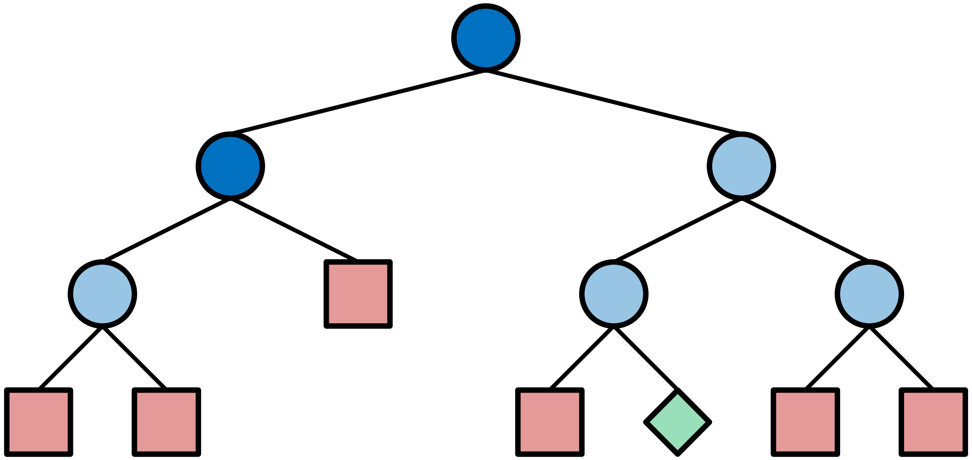
- Example branching
  - pick variable x (at least two values)
  - pick value n (from domain of x)
  - branch with $x = n$ and $x \neq n$
- Heuristic needed
  - which variable to select?
  - which value to select?

Search: Heuristic ⇆ Exploration



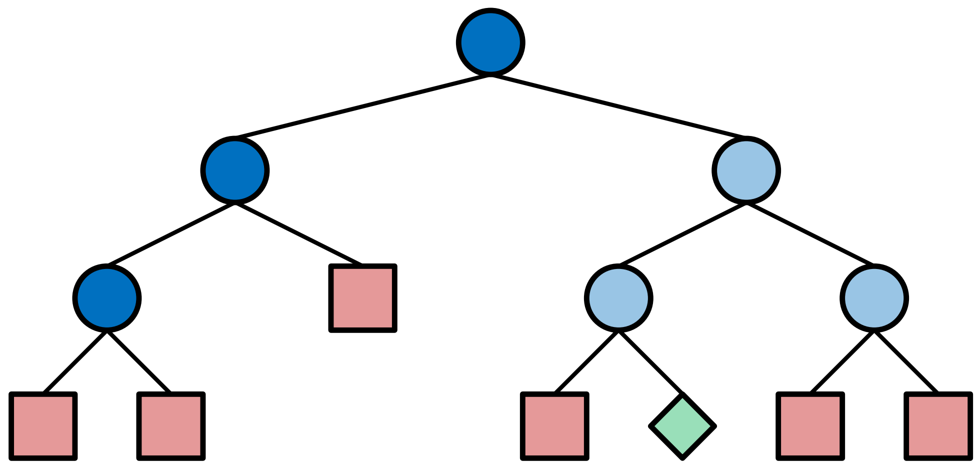
- Heuristic branching
  - defines tree shape
- Exploration of search tree
  - orthogonal aspect: DFS,

Search: Heuristic ⇆ Exploration



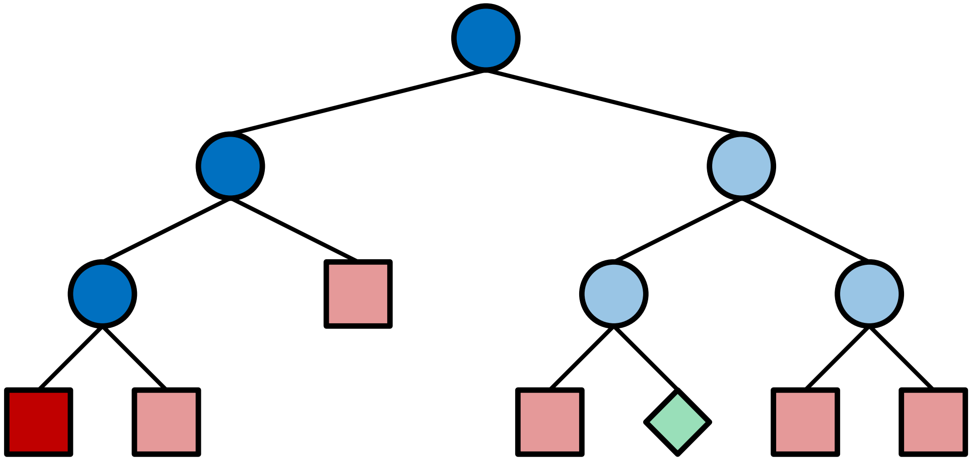
- Heuristic branching
  - defines tree shape
- Exploration of search tree
  - orthogonal aspect: DFS,

Search: Heuristic ⇆ Exploration



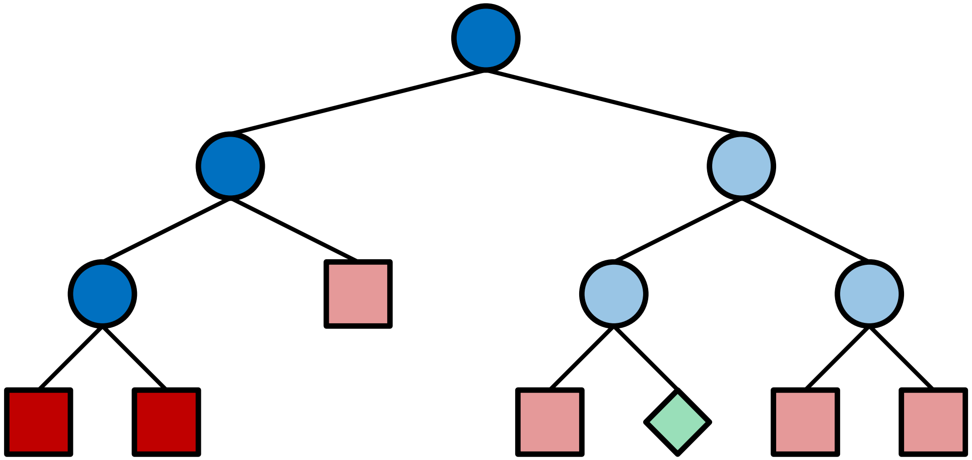
- Heuristic branching
  - defines tree shape
- Exploration of search tree
  - orthogonal aspect: DFS,

Search: Heuristic ⇆ Exploration



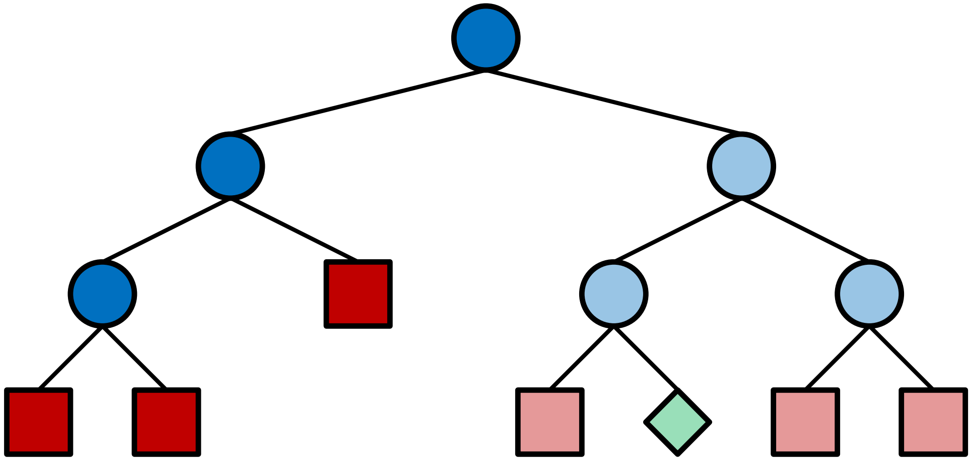
- Heuristic branching
  - defines tree shape
- Exploration of search tree
  - orthogonal aspect: DFS,

Search: Heuristic ⇆ Exploration



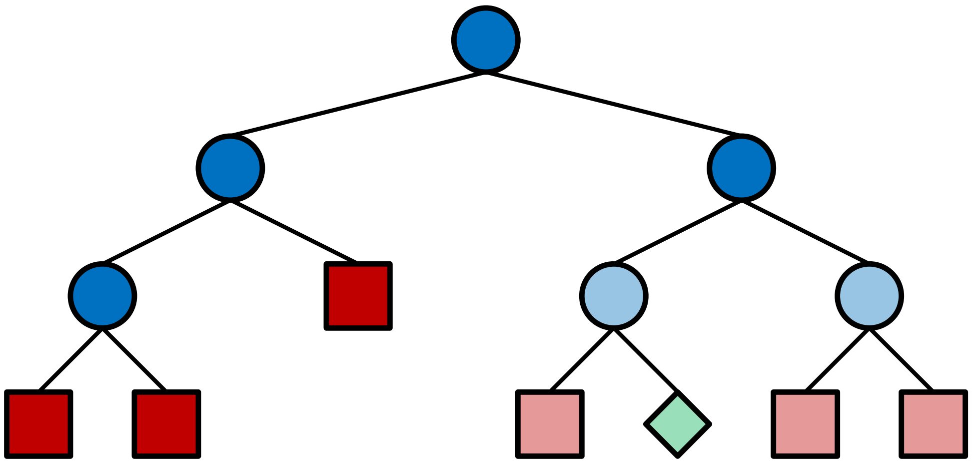
- Heuristic branching
    - defines tree shape
- Exploration of search tree
    - orthogonal aspect: DFS,

Search: Heuristic ⇆ Exploration



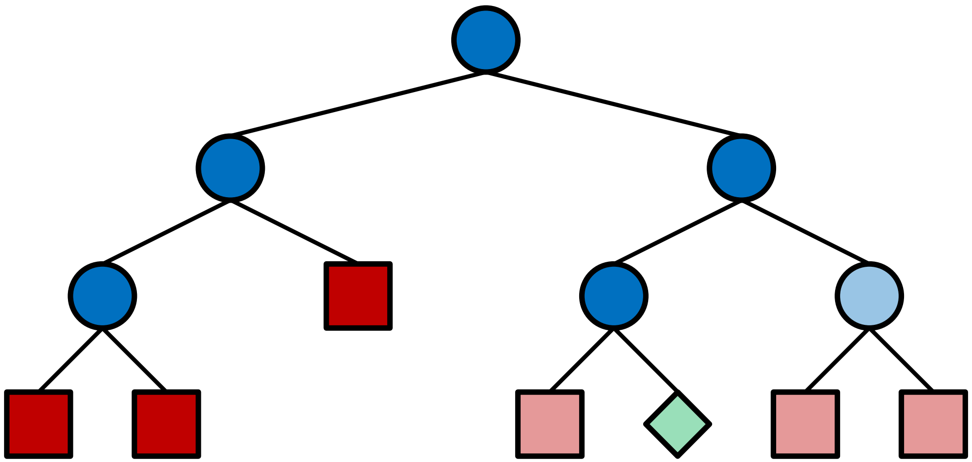
- Heuristic branching
  - defines tree shape
- Exploration of search tree
  - orthogonal aspect: DFS,

Search: Heuristic ⇆ Exploration



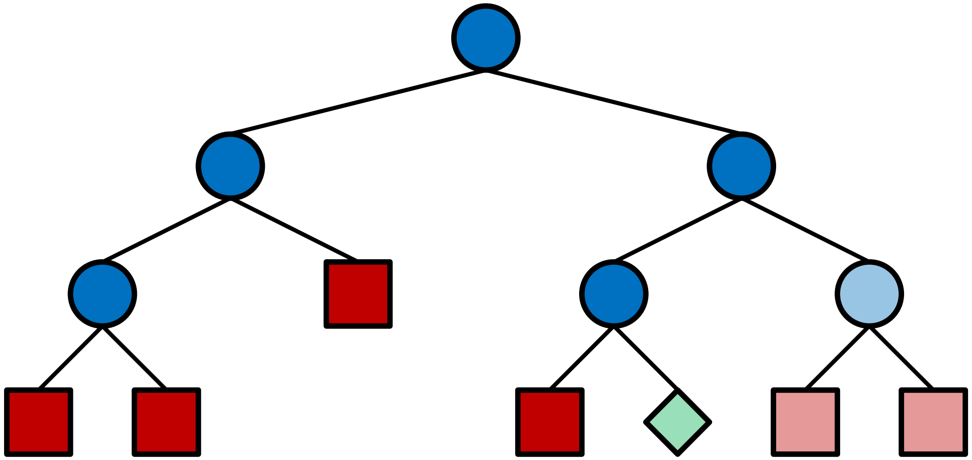
- Heuristic branching
    - defines tree shape
- Exploration of search tree
    - orthogonal aspect: DFS,

Search: Heuristic ⇆ Exploration



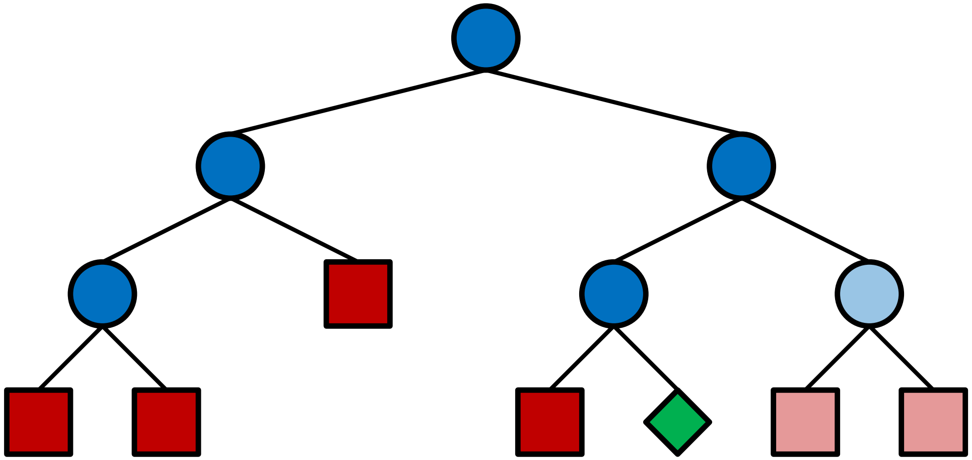
- Heuristic branching
  - defines tree shape
- Exploration of search tree
  - orthogonal aspect: DFS,

Search: Heuristic ⇆ Exploration



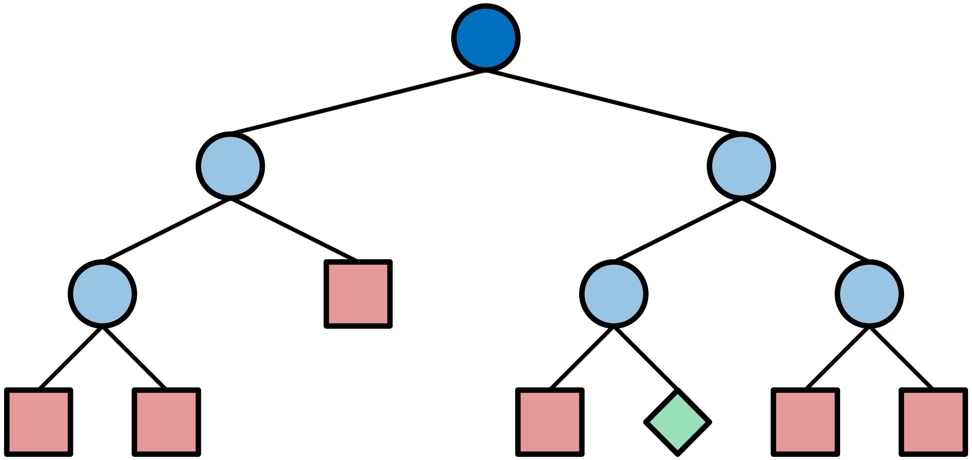
- Heuristic branching
  - defines tree shape
- Exploration of search tree
  - orthogonal aspect: DFS,

Search: Heuristic ⇆ Exploration



- Heuristic branching
  - defines tree shape
- Exploration of search tree
  - orthogonal aspect: DFS,

Search: Heuristic ⇆ Exploration



- Heuristic branching
  - defines tree shape
- Exploration of search tree
  - orthogonal aspect: DFS, BFS, IDFS, LDS, parallel, ...

Summary

- Modeling
  - variables with domain
  - constraints to state relations
  - branching strategy
  - in real: an array of modeling techniques…
- Solving
  - constraint propagation
  - branching
  - search tree exploration
  - in real: an array of solving techniques…

THE CONSTRAINT PROGRAMMER'S TOOLBOX

Widely Applicable

- Timetabling
- Scheduling
- Personnel and crew rostering
- Resource allocation
- Workflow planning and optimization
- Gate allocation at airports
- Sports-event scheduling
- Railroad: track allocation, train allocation, schedules
- Automatic composition of music
- Genome sequencing
- Frequency allocation
- …

Problems Are Hard

- The problems are NP hard
  - no efficient algorithm is likely to exist
- Tremendously difficult to
  - always solve any problem instance
  - scale to large instances
  - have single silver bullet method
- Property of problems…

  …not of method

  …hence no silver bullet

Why Is a Toolbox Needed?

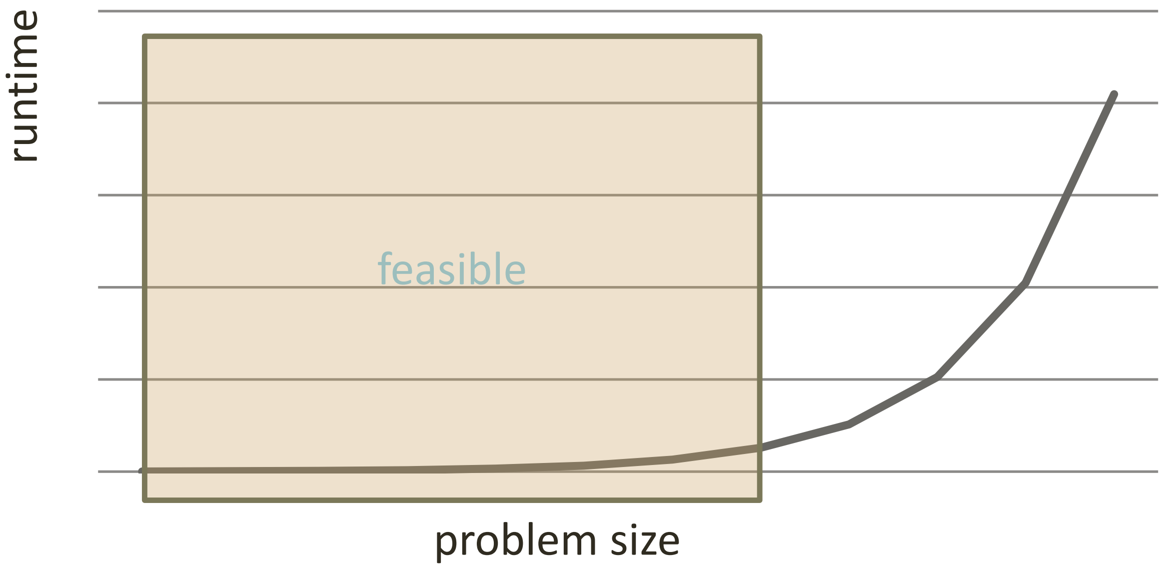
- Initial model: model to capture problem
  - correctness
- Improved model: model to solve problem
  - robustness and scalability
  - often difficult
- Tools in the toolbox are needed for…

  …**modeling to solve problems**

Parts of the Toolbox

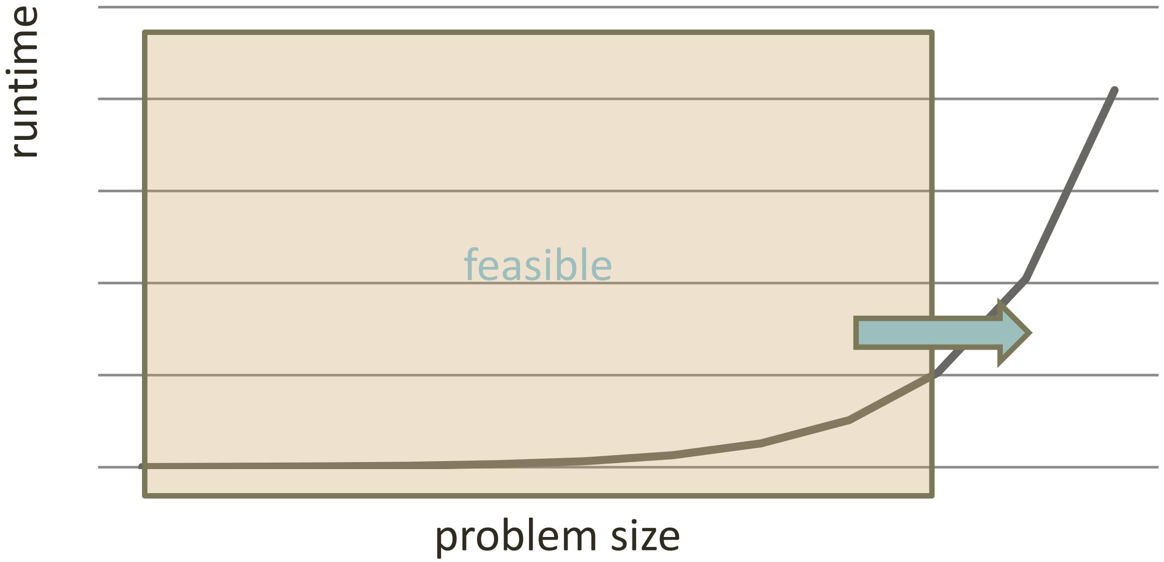
- "Global" constraints
    - capture structure during modeling
    - provide strong constraint propagation
- Application specific search heuristics
- Symmetries and dominance relations
    - reduce size of search space
- Propagation-boosting constraints
- Randomized restarts during search
    - including no-goods from restarts
- …

The Best We Can Hope for…



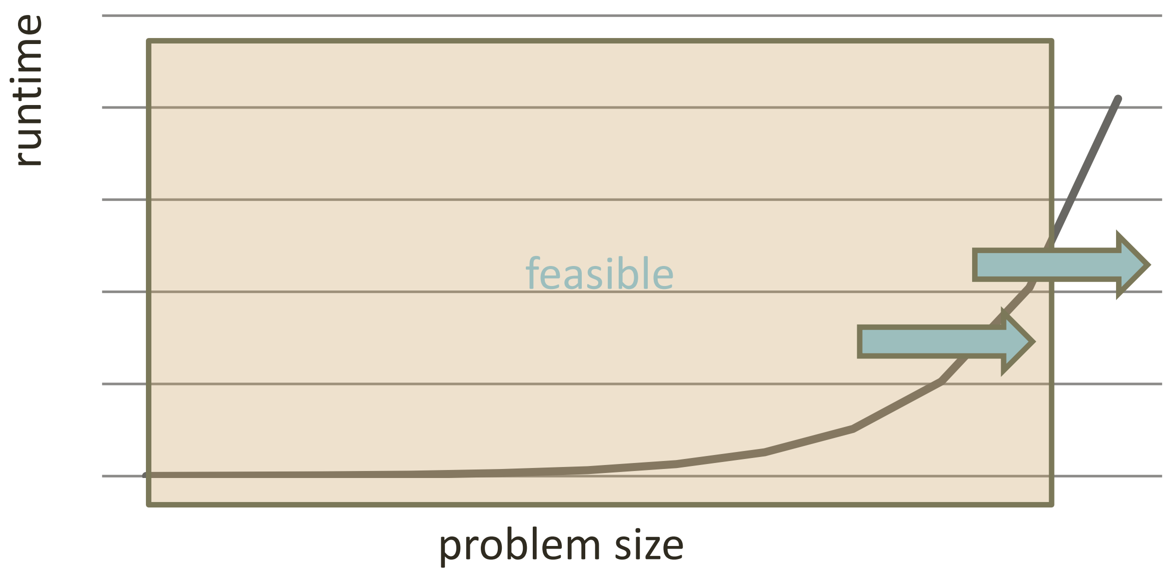
- Exponential growth in runtime
- **Without** using tools

The Best We Can Hope for…



- Exponential growth in runtime
- **With** propagation and heuristic search

The Best We Can Hope for…



- Exponential growth in runtime
- **With** propagation, heuristic search, symmetry breaking, restarts, …

Essence

- Constraint programming is about…

  ...local reasoning exploiting structure

  ...an array of modeling tools for solving

- Strength
  - simplicity, compositionality, exploiting structure
  - rich toolbox of techniques

- Challenges
  - lack of global picture during search
  - difficult to find global picture due to rich structure

SOME GECODE FACTS

Gecode People

- Core team
  - Christian Schulte, Guido Tack, Mikael Z. Lagerkvist.
- Code
  - contributions: Christopher Mears, David Rijsman, Denys Duchier, Filip Konvicka, Gabor Szokoli, Gabriel Hjort Blindell, Gregory Crosswhite, Håkan Kjellerstrand, Joseph Scott, Lubomir Moric, Patrick Pekczynski, Raphael Reischuk, Stefano Gualandi, Tias Guns, Vincent Barichard.
  - fixes: Alexander Samoilov, David Rijsman, Geoffrey Chu, Grégoire Dooms, Gustavo Gutierrez, Olof Sivertsson, Zandra Norman.
- Documentation
  - contributions: Christopher Mears.
  - fixes: Seyed Hosein Attarzadeh Niaki, Vincent Barichard, Pavel Bochman, Felix Brandt, Markus Böhm, Roberto Castañeda Lozano, Gregory Crosswhite, Pierre Flener, Gustavo Gutierrez, Gabriel Hjort Blindell, Sverker Janson, Andreas Karlsson, Håkan Kjellerstrand, Chris Mears, Benjamin Negrevergne, Flutra Osmani, Max Ostrowski, David Rijsman, Dan Scott, Kish Shen.

Gecode

Generic Constraint Development Environment

- **open**
  - easy to interface with other systems
  - supports programming of: constraints, branching strategies, search engines, variable domains
- **comprehensive**
  - constraints over integers, Booleans, sets, and floats
    - different propagation strength, half and full reification, …
  - advanced branching heuristics (accumulated failure count, activity)
  - many search engines (parallel, interactive graphical, restarts)
  - automatic symmetry breaking (LDSB)
  - no-goods from restarts
  - MiniZinc support

Gecode

Generic Constraint Development Environment

- **efficient**
  - *all* gold medals in *all* categories at MiniZinc Challenges 2008-2012
- **documented**
  - tutorial (> 500 pages) and reference documentation
- **free**
  - MIT license, listed as free software by FSF
- **portable**
  - implemented in C++ that carefully follows the C++ standard
- **parallel**
  - exploits multiple cores of today's hardware for search
- **tested**
  - some 50000 test cases, coverage close to 100%

History

- 2002
  - development started
- 1.0.0
  - December 2005
- 2.0.0
  - November 2007
- 3.0.0
  - March 2009
- 4.0.0
  - March 2013
- 4.2.1 (current)
  - November 2013

43 kloc, 21 klod

77 kloc, 41 klod

35 releases

81 kloc, 41 klod

164 kloc, 69 klod

168 kloc, 71 klod

Tutorial Documentation

- 2002
  - development started
- 1.0.0 — 43 kloc, 21 klod
  - December 2005
- 2.0.0 — 77 kloc, 41 klod
  - November 2007
- 3.0.0 — 1 klod
  - March 2009
- 4.0.0 — 164 kloc, 69 klod
  - March 2013
- 4.2.1 (current)
  - November 2013

Modeling with Gecode (98 pages)

Modeling & Programming with Gecode (522 pages)

Future

- Large neighborhood search and other meta-heuristics
    - contribution expected
- Simple temporal networks for scheduling
    - contribution expected
- More expressive modeling layer on top of libmzn
- Grammar constraints
    - contribution expected
- Propagator groups
- …

- Contributions anyone?

Deployment & Distribution

- Open source ≠ Linux only
  - Gecode is native citizen of: Linux, Mac, Windows
- High-quality
  - extensive test infrastructure (around 16% of code base)
  - you have just one shot!
- Downloads from Gecode webpage
  - software: between 25 to 125 per day
  - documentation: between 50 to 300 per day
- Included in
  - Debian, Ubuntu, FreeBSD, …

GOALS & USE CASES

Initial Goals

- Research
  - architecture of constraint programming systems
  - propagation algorithms, search, modeling languages, ...
- Efficiency
  - competitive
  - proving architecture right
- Education
  - state-of-the-art, free platform for teaching
- CP community service
  - provide an open platform for research (think back to 2002!)

Users

- Research
    - own papers
    - papers by others: experiments and comparison
    - Google scholar: some 1250 references to Gecode (March 2014)

- Education: teaching
    - KTH, Uppsala U, U Freiburg, UC Louvain, Saarland U, American U Cairo, U Waterloo, U Javeriana-Cali, …

- Industry
    - several companies have integrated Gecode into products (part of hybrid solvers)

Use Case: Education

- Courses feasible that include
  - modeling
  - principles

but also

  - programming search heuristics (branchers)
  - programming constraints (propagators)

- Essential for programming
  - accessible documentation…
  - …including many examples

Use Cases: Interfacing

- Quintiq integrates Gecode as CP component
    - available in their modeling language
- Cologne: A Declarative Distributed Constraint Optimization Platform
    - U Penn, AT&T Labs, Raytheon
    - Datalog + constraints in distributed setup [Liu ea, VLDB 2012]
- Constraint Programming for Itemset Mining (CP4IM)
    - declarative approach to constraint-based itemset mining [Guns, Nijssen, De Raedt, KU Leuven]
- Whatever language
    - modeling: AMPL, MiniZinc, …
    - programming: Java, Prolog (> 1), Lisp (> 1), Ruby, Python (> 1), Haskell, …

Use Cases: Research

- Benchmarking platform for models
  - lots of people (majority?)
- Benchmarking platform for implementations
  - lots of people
  - requires open source (improve what Gecode implements itself)
- Gecode models as reference
  - Castineiras, De Cauwer, O'Sullivan, Weibull-based Benchmarks for *Bin Packing*. CP 2012.
- Base system for extensions
  - Qecode: quantified constraints (Benedetti, Lalouet, Vautard)
  - Gelato: hybrid of propagation and local search (Cipriano, Di Gaspero, Dovier)
  - Gecode interfaces powerful enough: no extension required

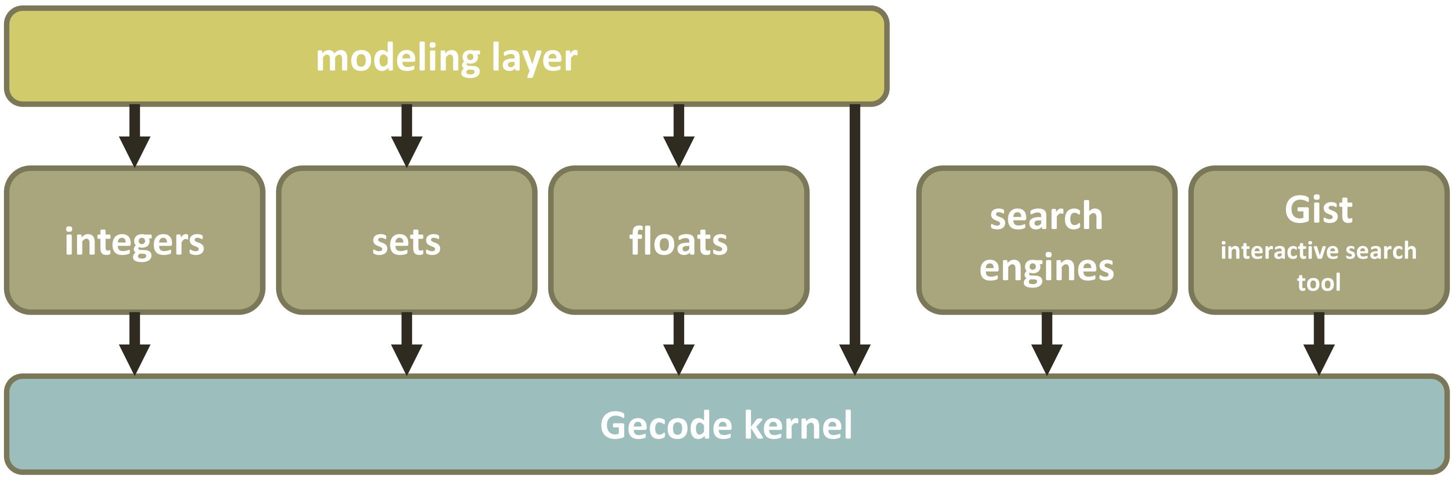
Use Cases: Other Systems

- Parts of Gecode integrated into other systems
    - Caspar (global constraint implementations)
    - Google or-tools
    - possibly more: that’s okay due to MIT license

- Gecode as starting point for other systems
    - Opturion’s CPX Discrete Optimizer
    - definitely more: that’s okay due to MIT license

MODELING & PROGRAMMING

Architecture



- Small domain-independent kernel
- Modules
  - per variable type: variables, constraint, branchings, …
  - search, FlatZinc support, …
- Modeling layer
  - arithmetic, set, Boolean operators; regular expressions; matrices, …
- All APIs are user-level and documented (tutorial + reference)

Modeling (interfacing)

- Use modeling layer in C++
    - matrices, operators for arithmetical and logical expressions, …
- Use predefined
    - constraints
    - search heuristics and engines
- Documentation
    - getting started 30 pages
    - concepts and functionality 130 pages
    - case studies 80 pages

Modeling (interfacing)

- Constraint families
  - arithmetics, Boolean, ordering, ….
  - alldifferent, count (global cardinality, …), element, scheduling, table and regular, sorted, sequence, circuit, channel, bin-packing, lex, geometrical packing, nvalue, lex, value precedence, …
- Families
  - different variants and different propagation strength
- “All” global constraints from MiniZinc have native implementation in Gecode

Gecode ⇆ Global Constraint Catalogue

- 74 constraints implemented:

  abs_value, all_equal, alldifferent, alldifferent_cst, among, among_seq, among_var, and, arith, atleast, atmost, bin_packing, bin_packing_capa, circuit, clause_and, clause_or, count, counts, cumulative, cumulatives, decreasing, diffn, disjunctive, domain, domain_constraint, elem, element, element_matrix, eq, eq_set, equivalent, exactly, geq, global_cardinality, gt, imply, in, in_interval, in_intervals, in_relation, in_set, increasing, int_value_precede, int_value_precede_chain, inverse, inverse_offset, leq, lex, lex_greater, lex_greatereq, lex_less, lex_lesseq, link_set_to_booleans, lt, maximum, minimum, nand, neq, nor, not_all_equal, not_in, nvalue, nvalues, or, roots, scalar_product, set_value_precede, sort, sort_permutation, strictly_decreasing, strictly_increasing, sum_ctr, sum_set, xor

Programming

- Interfaces for programming
  - propagators (for constraints)
  - branchers (for search heuristics)
  - variables
  - search engines

- Documentation

	intro	advanced
propagators	40 pages	60 pages
branchers	12 pages	8 pages
variables		44 pages
search engines	12 pages	26 pages

OPENNESS

Open Source

- MIT license
  - permits commercial, closed-source use
  - disclaims all liabilities (as far as possible)
- License motivation
  - public funding
  - focus on research
- Not a reason
  - attitude, politics, dogmatism
- Problem
  - cannot really use GPL-licensed software

Open Architecture

- More than a license
  - **license** restricts what users **may do**
  - **code and documentation** restrict what users **can do**
- Modular, structured, documented, readable
  - complete tutorial and reference documentation
  - ideas based on scientific publications
- Equal rights: clients are first-class citizens
  - you can do what we can do: APIs
  - you can know what we know: documentation
  - on every level of abstraction!

Open Development

- We encourage contributions
  - direct, small contributions

    → we take over maintenance and distribution
  - larger modules on top of Gecode

    → you maintain the code, we distribute it

- Prerequisites
  - MIT license
  - compiles and runs on platforms we support

Golomb ruler (CSPlib problem 006) à la Gecode

EXAMPLE MODEL

Golomb Ruler: Model

```
Golomb(int n) : m(*this, n, 0, Int::Limits::max) {









}
```

- Declare `n` variables with values between `0` and largest integer value

Golomb Ruler: Model

```
Golomb(int n) : m(*this, n, 0, Int::Limits::max) {
  rel(*this, m[0] == 0);
  rel(*this, m, IRT_LE);










}
```

- Constrain first mark `m[0]` to `0`
- Constrain marks `m` to be strictly increasing (`IRT_LE` = integer relation type for less)

Golomb Ruler: Model

```
Golomb(int n) : m(*this, n, 0, Int::Limits::max) {
  rel(*this, m[0] == 0);
  rel(*this, m, IRT_LE);

  IntVarArgs d;
  for (int k=0, i=0; i<n-1; i++)
    for (int j=i+1; j<n; j++, k++)
      d << expr(*this, m[j] - m[i]);
  distinct(*this, d);


}
```

- Collect variables for distances between marks in d
- Constrain d to be all different (`distinct` in Gecode)

Golomb Ruler: Model

```
Golomb(int n) : m(*this, n, 0, Int::Limits::max) {
  rel(*this, m[0] == 0);
  rel(*this, m, IRT_LE);

  IntVarArgs d;
  for (int k=0, i=0; i<n-1; i++)
    for (int j=i+1; j<n; j++, k++)
      d << expr(*this, m[j] - m[i]);
  distinct(*this, d);

  branch(*this, m, INT_VAR_NONE(), INT_VAL_MIN());
}
```

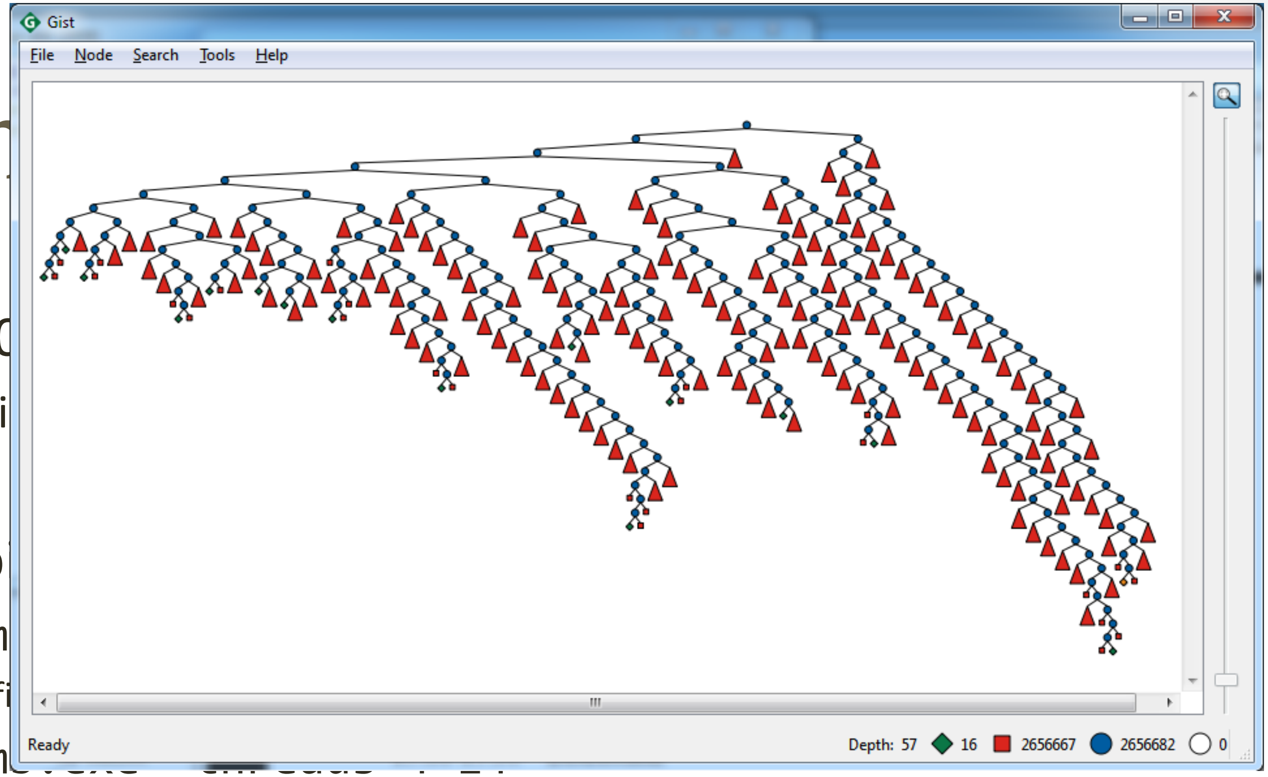
- Branch on marks `m` with no selection strategy (left-to-right) and try the smallest value first

Golomb Ruler: Cost Function

```
IntVar cost() const {
  return m[m.size()-1];
}
```

- Return last mark as cost

Runnin



- Some 10-2
  - followi
- After comp
  - golom
    - fi
  - golom
    - use four threads for parallel search for 12 marks
  - golomb.exe -mode gist 12
    - use graphical interactive search tool (scales to millions of nodes)

or

- use restarts… `-restart luby`
- use no-goods from restarts… `-no-goods 1`
- lots more (too many as some people say… ☹)

Summary

- Gecode is…

open **comprehensive** **efficient**
documented **free** **portable**
parallel **tested**

…and pretty cool for…

modeling **solving**
programming **interfacing**