



Gecode

an open constraint solving library



Christian Schulte

cschulte@kth.se

KTH Royal Institute of Technology
School of Electrical Engineering and Computer Science
Stockholm, Sweden

Joint Work With...

- 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, Kevin Leo, Linnea Ingmar, Lubomir Moric, Matthias Balzer, Maxim Shishmarev, Patrick Pekczynski, Raphael Reischuk, Stefano Gualandi, Tias Guns, Vincent Barichard.
 - fixes and help: Adam Russell, Ahmed Kamal, Alberto Delgado, Alejandro Arbelaez, Alex Elliott, Alexander Kleff, Alexandre Fayolle, Alin Gherman, Ananja Muddukrishna, Anden Blah, Morten Boysen, Andrea Pretto, Andrea Rendl, Andreas Schutt, Anne Meyer, Bauke Conijn, Brian Kell, Carleton Coffrin, Chris Mears, Cliff Yapp, Conrad Drescher, David Barton, David Hotham, David Rijsman, David Zaremby, Denys Duchier, Dominik Brill, Duane Leslie, Duong Khanh Chuong, Farshid Hassani Bijarbooneh, Felix Brandt, Filip Konvicka, Frank Imeson, Gabriel Hjort Blindell, Geoffrey Chu, George Katsirelos, Gilles Pesant, Gustavo Guiterrez, Helmut Simonis, Jan Kelbel, Jan Wolter, Jaroslav Mlejnek, Javier Mena, Jean-Christophe Godart, Jip J. Dekker, Jonathan Cederberg, Jorge Marques Pelizzoni, Josef Eisl, Joseph Scott, Kari Pahula, Kish Shen, Kurt Van Den Branden, Lin Yong, Luca Di Gaspero, Luis Otero, Manuel Baclet, Manuel Loth, Marco Correia, Mark Manser, Martin Mann, Massimo Morara, Mats Carlsson, Matthias Balzer, Max Ostrowski, Maxim Shishmarev, Michal Dobrogost, Mirko Rahn, Mohamad Rabbath, Nicholas Tung, Nina Narodytska, Olof Sivertsson, Pascal Francq, Patrick Berg, Patrick Pekczynski, Peter Nightingale, Peter Penchev, Peter Stuckey, Peter Van Beek, Petter Strandmark, Philippe Kezirian, Pierre Talbot, Rafael Meneses, Raphael Reischuk, Roland Stigge, Roland Yap Hock Chuan, Ruben Zilibowitz, Samuel Gagnon, Serge Le Huitouze, Simon Ekv, Stanimir Dragiev, Stefano Gualandi, Thibaut Feydy, Tias Guns, Tommy Persson, Tony Kelman, Victor Zverovich, Vincent Barichard, Vivian De Smedt, Willem Van Onsem, Willem-Jan Van Hoeve, Zandra Norman, Zichen Zhu.

- Documentation

- contributions: Christopher Mears.
 - fixes: Andreas Karlsson, Benjamin Negrevergne, Chris Mears, Dan Scott, David Rijsman, Flutra Osmani, Gabriel Hjort Blindell, Gregory Crosswhite, Gustavo Gutierrez, Håkan Kjellerstrand, Kish Shen, Léonard Benedetti, Markus Böhm, Max Ostrowski, Pavel Bochman, Pierre Flener, Roberto Castañeda Lozano, Sverker Janson, Vincent Barichard.

WHAT IS GECODE?

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, action, CHB)
 - many search engines (parallel, interactive graphical, restarts, portfolios)
 - automatic symmetry breaking (LDSB)
 - no-goods from restarts
 - MiniZinc support
 - tracing support (propagation and search)
 - support for CP Profiler
 - ...

Gecode

Generic Constraint Development Environment

- **efficient**
 - *all* gold medals in *all* categories at MiniZinc Challenges 2008-2012
- **documented**
 - tutorial (> 570 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
- 5.0.0
 - October 2016
- 6.0.0
 - February 2018
- 6.0.1 (current)
 - May 2018



43 kloc, 21 klod

77 kloc, 41 klod

46 releases


81 kloc, 41 klod
164 kloc, 69 klod

288 kloc, 86 klod

295 kloc, 89 klod

296 kloc, 85 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	Modeling with Gecode (98 pages)	klod
• March 2009		
• 4.0.0		164 kloc, 69 klod
• March 2013		
• 5.0.0		288 kloc, 86 klod
• October 2016		
• 6.0.0		295 kloc, 89 klod
• February 2018		
• 6.0.1 (current)	Modeling & Programming with Gecode (572 pages)	295 kloc, 89 klod
• May 2018		

Deployment & Distribution

- Open source \neq 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 (old)
 - software: between 25 to 125 per day
 - documentation: between 50 to 300 per day
- Recently started migration to GitHub
 - code all there, webpages on github.io
 - old server to be switched off in June
- 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!)
- Collaboration platform

Strengths of CP

- Modeling
 - many constraints available to capture problem structure
- Solving
 - dedicated propagation algorithms for many (most) constraints
 - powerful search methods available: heuristics, randomization, restarts, portfolios, no-goods, ...
- Extending
 - design and implement new constraints
 - design and implement new heuristics
 - design and implement new search engines
 - design and implement new variable types

Strengths of CP

- **Modeling**

- many constraints available to capture problem structure

- **Solving**

- dedicated propagation algorithms for many (most) constraints
- powerful search methods available: heuristics, restarts, portfolios, etc.

- **Extending**

- design and implement new constraints
- design and implement new variable types
- design and implement new solvers
- design and implement new variable types

Use MiniZinc!
Do not use Gecode!

[Gecode included in MiniZinc distribution]

Strengths of CP

- Modeling
 - many constraints available to capture problem structure
- Solving
 - dedicated propagation algorithms for many (most) constraints
 - **powerful search methods available: heuristics, randomization, restarts, portfolios, no-goods, ...**
- Extending
 - design and implement new constraints
 - design and implement new solvers
 - design and implement new variable types
 - design and implement new variable types



Strengths of CP

- Modeling

- many constraints available to express

- Solving

- dedicated propagation
 - powerful search methods available: minimization, restarts, portfolios, no-goods, ...



Use Gecode!

[pretty much only choice]

- **Extending**

- design and implement new constraints
 - design and implement new heuristics
 - design and implement new search engines
 - design and implement new variable types

Users

- Research
 - own papers
 - papers by others: experiments and comparison
 - Google scholar: some 1760 references to Gecode (May 2018)
- 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)
 - preparing press release with “X” among top ten software companies

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 (case studies)

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

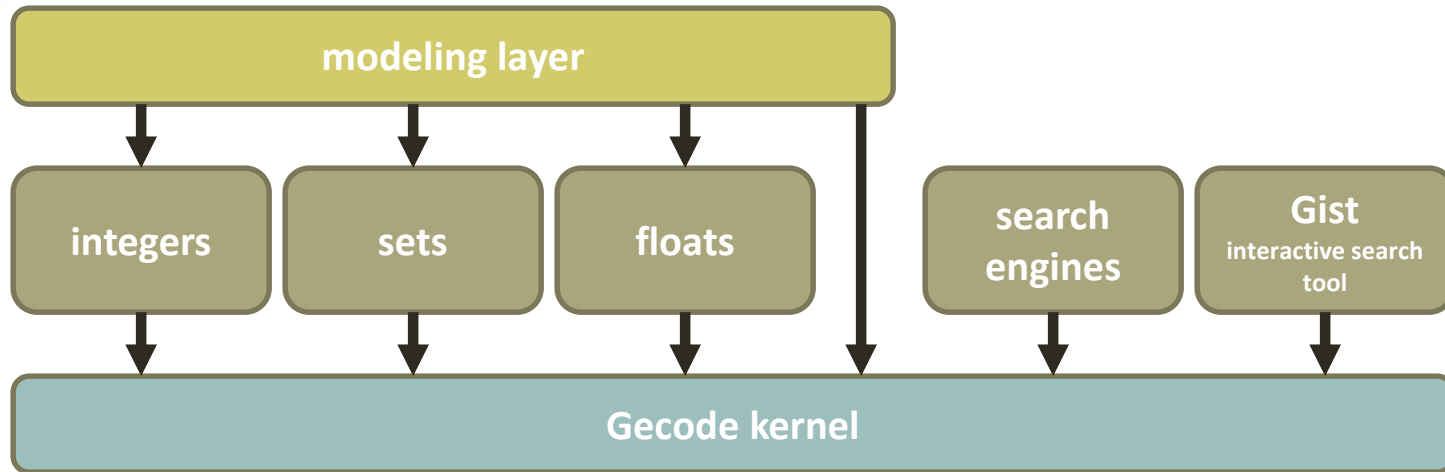
- Benchmarking platform for models
 - lots of people (majority?)
- Benchmarking platform for implementations
 - lots of people
 - requires open source (improve what Gecode implements itself)
- Gecode models as reference
 - Castineiras, De Cauwer, O'Sullivan, Weibull-based Benchmarks for *Bin Packing*. CP 2012.
- Base system for extensions
 - Qecode: quantified constraints (Benedetti, Lalouet, Vautard)
 - Gelato: hybrid of propagation and local search (Cipriano, Di Gaspero, Dovier)
 - Bounded-length sequence variables (Scott, Flener, Pearson, Schulte)
 - **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, regular expressions, ...
- Use predefined
 - constraints
 - search heuristics and engines
- Documentation
 - getting started ≈ 30 pages
 - concepts and functionality ≈ 180 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 \Leftrightarrow 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
 - impact on industry
- 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

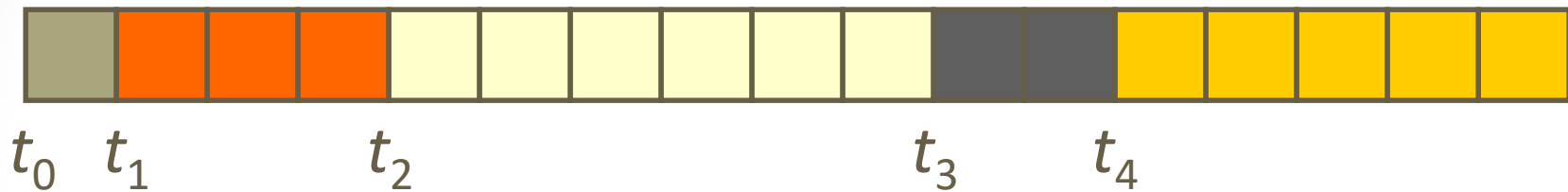
Contributions

- We take over means
 - fully documented
 - fully tested
 - following Gecode style to a certain extent
- We might not accept if we feel that we cannot maintain it!
- Even after we accept we might remove if we learn that we cannot maintain it

Golomb rulers (CSPlib problem 006) à la Gecode

EXAMPLE MODEL

Golomb Rulers



- Find n ticks t_i on ruler such that
 - distance $|t_j - t_i|$ between all ticks pairwise distinct
 - length of ruler minimal
- Extremely hard problem
 - applications in crystallography, ...

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
 - INT_VAR_NONE: no selection strategy (left-to-right)
 - INT_VAL_MIN: try smallest value first

Golomb Ruler: Cost Function

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

- Return last mark as cost

Running

- Some 10-20
 - followi

- After comp

- golom
- fi

- golom

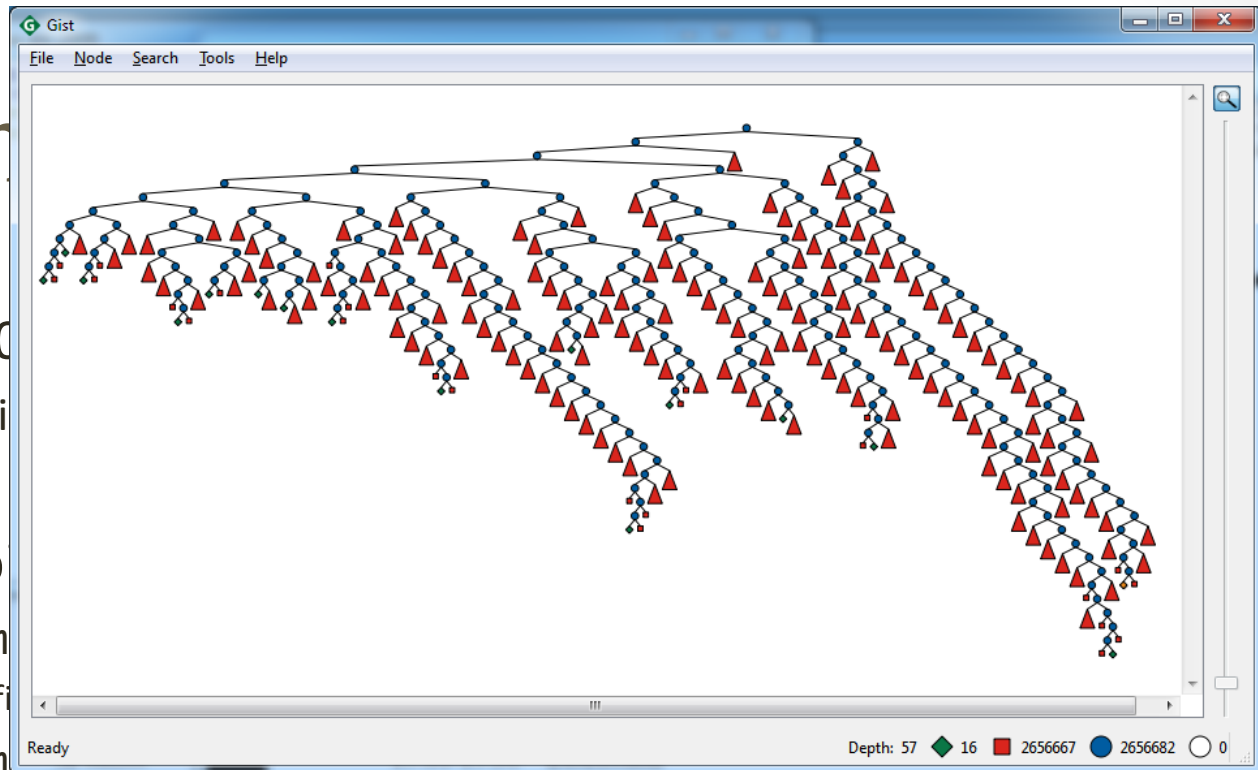
- use four threads for parallel search for 12 marks

- `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