The SCIP Optimization Suite

Gerald Gamrath

Zuse Institute Berlin

EWO Seminar

February 13, 2015

ZIB: Fast Algorithms – Fast Computers

Konrad-Zuse-Zentrum für Informationstechnik Berlin

- non-university research institute and computing center of the state of Berlin
- Research Units:
  - numerical analysis and modeling
  - visualization and data analysis
  - optimization: energy–traffic–telecommunication–linear and nonlinear IP
  - scientific information systems
  - distributed algorithms and supercomputing
- President: Martin Grötschel
- more information: `http://www.zib.de`

ZIB: Fast Algorithms – Fast Computers

Konrad-Zuse-Zentrum für Informationstechnik Berlin

- non-university research institute and computing center of the state of Berlin
- Research Units:
  - numerical analysis and modeling
  - visualization and data analysis
  - optimization: energy–traffic–telecommunication–**linear and nonlinear IP**
  - scientific information systems
  - distributed algorithms and supercomputing
- President: Martin Grötschel
- more information: `http://www.zib.de`

Outline

SCIP – Solving Constraint Integer Programs

Constraint Integer Programming

Solving Constraint Integer Programs

History and Applications

http://scip.zib.de

What is a Constraint Integer Program?

Mixed Integer Program

Objective function:

- linear function

Feasible set:

- described by linear constraints

Variable domains:

- real or integer values

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax \le b \\ & (x_I, x_C) \in \mathbb{Z}^I \times \mathbb{R}^C \end{aligned}$$

Constraint Program

Objective function:

- arbitrary function

Feasible set:

- given by arbitrary constraints

Variable domains:

- arbitrary (usually finite)

$$\begin{aligned} \min \quad & c(x) \\ s.t. \quad & x \in F \\ & (x_I, x_N) \in \mathbb{Z}^I \times X \end{aligned}$$

What is a Constraint Integer Program?

Constraint Integer Program

Objective function:

▷ linear function

Feasible set:

▷ described by arbitrary constraints

Variable domains:

▷ real or integer values

When all integer variables fixed:

▷ CIP becomes an LP

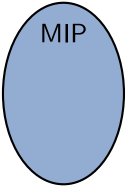
$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & x \in F \\ & (x_I, x_C) \in \mathbb{Z}^I \times \mathbb{R}^C \end{aligned}$$

Remark:

- arbitrary objective or variables modeled by constraints

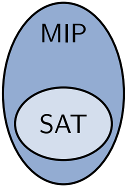
Constraint Integer Programming

- **M**ixed **I**nteger **P**rograms



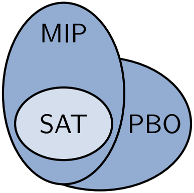
Constraint Integer Programming

- **M**ixed **I**nteger **P**rograms
- **SAT**isfiability problems



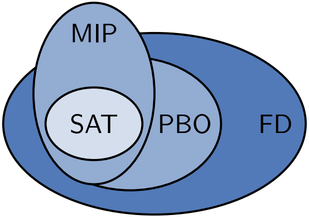
Constraint Integer Programming

- Mixed Integer Programs
- SATisfiability problems
- Pseudo-Boolean Optimization



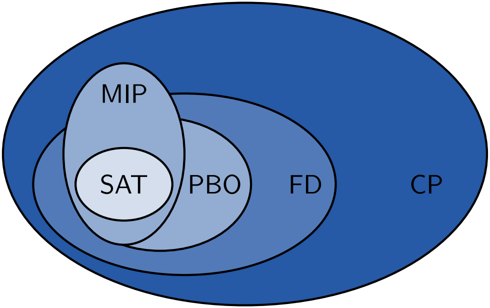
Constraint Integer Programming

- Mixed Integer Programs
- SATisfiability problems
- Pseudo-Boolean Optimization
- Finite Domain



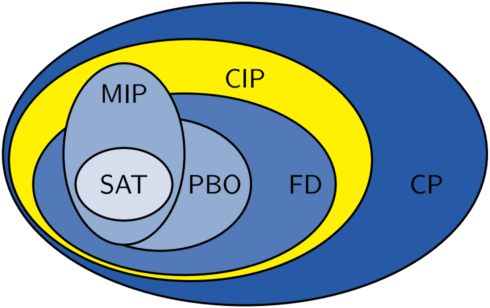
Constraint Integer Programming

- Mixed Integer Programs
- SATisfiability problems
- Pseudo-Boolean Optimization
- Finite Domain
- Constraint Programming



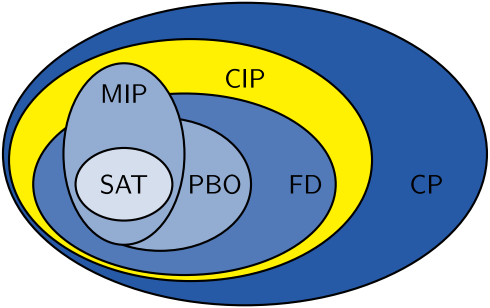
Constraint Integer Programming

- Mixed Integer Programs
- SATisfiability problems
- Pseudo-Boolean Optimization
- Finite Domain
- Constraint Programming
- Constraint Integer Programming



Constraint Integer Programming

- Mixed Integer Programs
- SATisfiability problems
- Pseudo-Boolean Optimization
- Finite Domain
- Constraint Programming
- Constraint Integer Programming

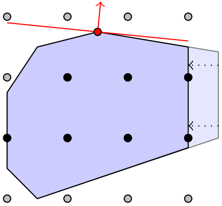


Relation to CP and MIP

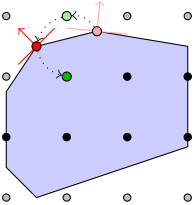
- Every MIP is a CIP. *"MIP $\subsetneq$ CIP"*
- Every CP over a finite domain space is a CIP. *"FD $\subsetneq$ CIP"*

CIP Solving Techniques

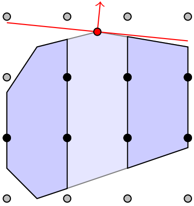
Presolving



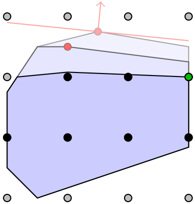
Primal Heuristics



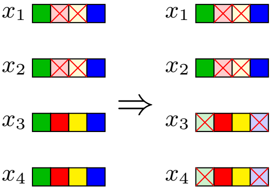
Branch & Bound



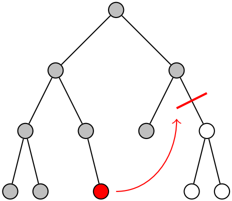
Cutting Planes



Domain Propagation



Conflict Analysis



MIP, CP, and SAT

- branch-and-bound

MIP

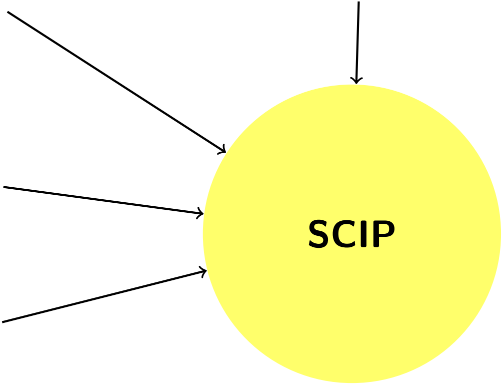
- LP relaxation
- cutting planes

CP

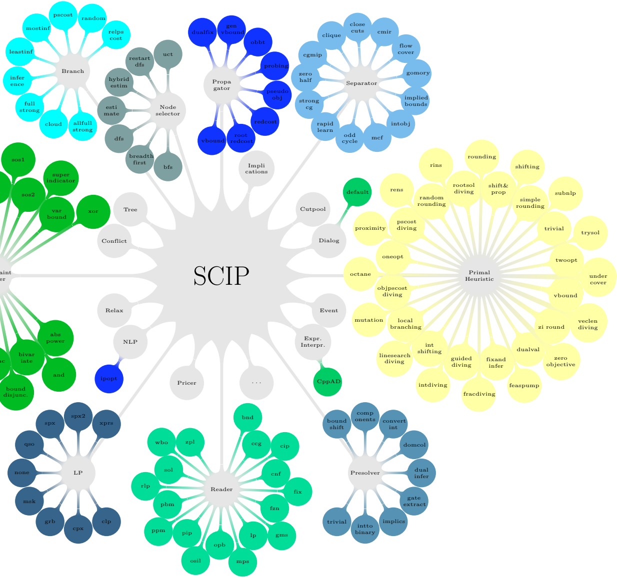
- domain propagation

SAT

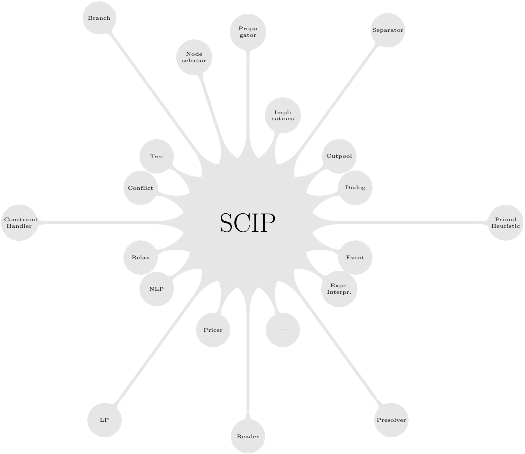
- conflict analysis
- periodic restarts



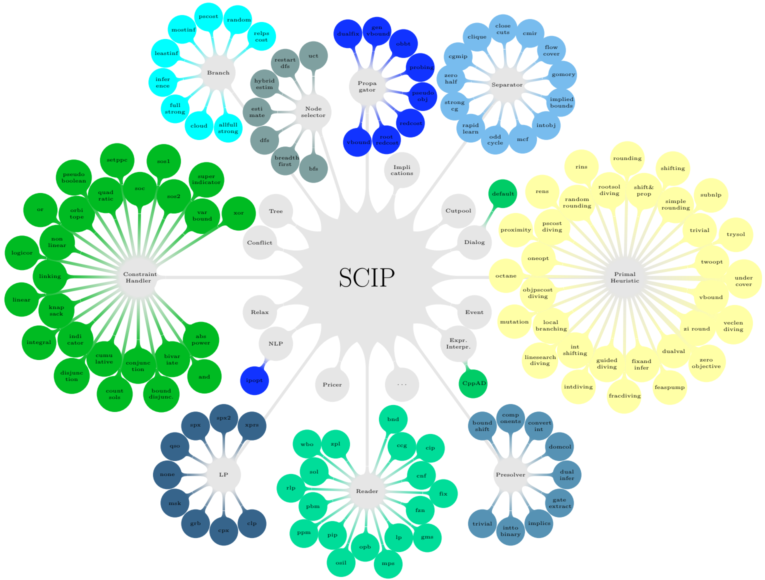
SCIP's Modular, Plugin-based Structure



SCIP's Modular, Plugin-based Structure



SCIP's Modular, Plugin-based Structure



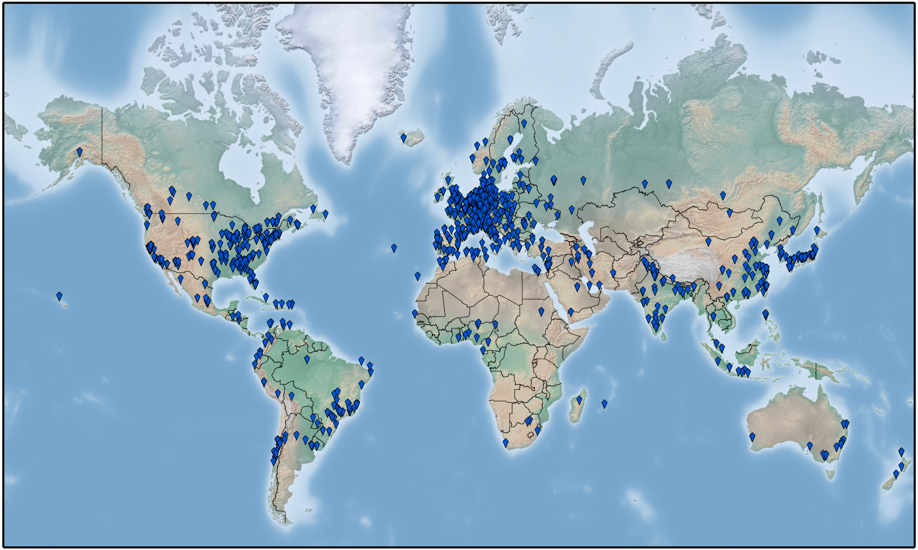
Some facts about SCIP

- general setup
  - plugin based system
  - default plugins handle MIPs and nonconvex MINLPs
  - support for branch-and-price and custom relaxations
- documentation and guidelines
  - more than 450 000 lines of C code, 20% documentation
    - 30 000 assertions, 4 000 debug messages
  - HowTos: plugins types, debugging, automatic testing
  - 11 examples illustrating the use of SCIP
  - active mailing list `scip@zib.de` (300 members)
- interface and usability
  - user-friendly interactive shell
  - interfaces to AMPL, GAMS, ZIMPL, MATLAB, Python and Java
  - C++ wrapper classes
  - LP solvers: CLP, CPLEX, Gurobi, MOSEK, QSopt, SoPlex, Xpress
  - over 1 600 parameters and 15 emphasis settings
- about 8000 downloads per year from 100+ countries

(Some) Universities and Institutes using SCIP

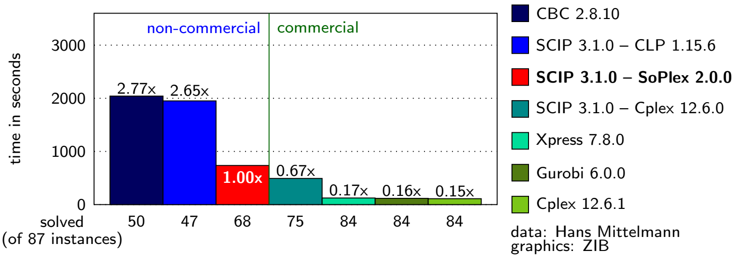
- RWTH Aaachen
- Universität Bayreuth
- FU Berlin
- TU Berlin
- HU Berlin
- WIAS Berlin
- TU Braunschweig
- TU Chemnitz
- TU Darmstadt
- TU Dortmund
- TU Dresden
- Universität Erlangen-Nürnberg
- Leibniz Universität Hannover
- Universität Heidelberg
- Fraunhofer ITWM Kaiserslautern
- Universität Karlsruhe
- Christian-Albrechts-Universität zu Kiel
- Hochschule Lausitz
- OvGU Magdeburg
- TU München
- Universität Osnabrück
- Universität Stuttgart
- Aarhus Universitet
- The University of Adelaide
- Università dell'Aquila
- Arizona State University
- University of Assiut
- National Technical University of Athens
- Georgia Institute of Technology
- Indian Institute of Science
- Tsinghua University
- UC Berkeley
- Lehigh University
- University of Bristol
- Eötvös Loránd Tudományegyetem
- Universidad de Buenos Aires
- Institut Français de Mécanique Avancée
- Chuo University
- Clemson University
- University College Cork
- Danmarks Tekniske Universitet
- Syddansk Universitet
- Kyushu University
- Fuzhou University
- Jinan University
- Rijksuniversiteit Groningen
- Hanoi Institute of Mathematics
- The Hong Kong Polytechnic University
- University of Hyogo
- The Irkutsk Scientific Center
- University of the Witwatersrand
- Københavens Universitet
- Kunming Botany Institute
- École Poly. Fédérale de Lausanne
- Linköpings universitet
- Université catholique de Louvain
- Universidad Rey Juan Carlos
- Université de la Mediterranée Aix-Marseille
- University of Melbourne
- UNAM
- Politecnico di Milano
- Università degli Studi di Milano
- Monash University
- Ikerlan
- Université de Montréal
- NIISI RAS
- Université de Nantes
- The University of Newcastle
- University of Nottingham
- Universitetet i Oslo
- Università degli Studi di Padova
- L'Université Sud de Paris
- Brown University
- The University of Queensland
- IASI CNR
- Erasmus Universiteit Rotterdam
- Carnegie Mellon University
- Universidad Diego Portales
- University of Balochistan
- Universidad San Francisco de Quito
- Universidade Federal do Rio de Janeiro
- Universidade de São Paulo
- Fudan University
- University of New South Wales
- Tel Aviv University
- The University of Tokyo
- Politecnico di Torino
- University of Toronto
- NTNU i Trondheim
- The University of York
- University of Washington
- University of Waterloo
- Massey University
- Austrian Institute of Technology
- TU Wien
- Universität Wien
- Wirtschaftsuniversität Wien
- ETH Zürich

(Some) Universities and Institutes using SCIP



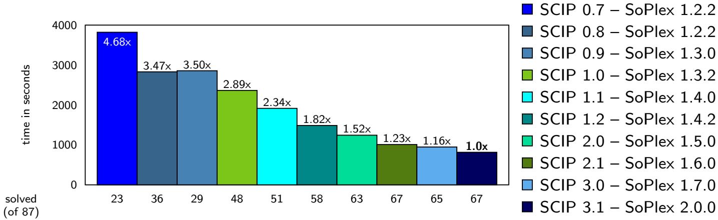
SCIP 3.1: Performance

▷ fastest non-commercial MIP solver



data: Hans Mittelmann
graphics: ZIB

▷ versionwise performance improvements



SCIP Optimization Suite

- Toolbox for generating and solving constraint integer programs
- free for academic use, available in source code

SCIP Optimization Suite

- Toolbox for generating and solving constraint integer programs
- free for academic use, available in source code

ZIMPL

- model and generate LPs, MIPs, and MINLPs

SCIP

- MIP, MINLP and CIP solver, branch-cut-and-price framework

SoPlex

- revised primal and dual simplex algorithm

GCG

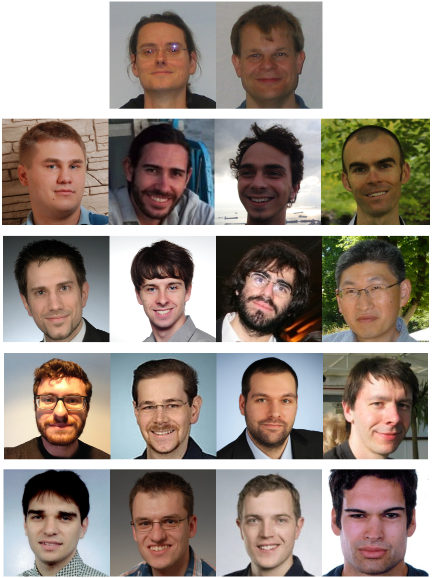
- generic branch-cut-and-price solver

UG

- framework for parallelization of MIP and MINLP solvers

Current Developers of the SCIP Optimization Suite

- Thorsten Koch
- Marc Pfetsch (TU Darmstadt)
- Gerald Gamrath
- Ambros Gleixner
- Gregor Hendel
- Stephen J. Maher
- Matthias Miltenberger
- Benjamin Müller
- Felipe Serrano
- Yuji Shinano
- Jakob Witzig
- Tobias Fischer (TU Darmstadt)
- Tristan Gally (TU Darmstadt)
- Stefan Vigerske (GAMS)
- Dieter Weninger (FAU Erlangen)
- Christian Puchert (RWTH Aachen)
- Jonas Witt (RWTH Aachen)
- Daniel Rehfeldt
- ...



History of the SCIP Optimization Suite

1996 SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])

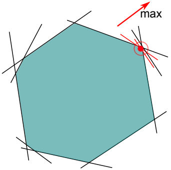
SoPlex – Sequential object-oriented simplex

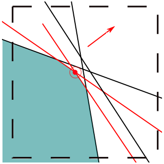
- implementation of the revised simplex algorithm
- primal and dual solving routines for linear programs
- iterative refinement to overcome numerical problems (Gleixner, Steffy, Wolter 2012)
  - fast and accurate solutions by repeated floating-point solves

$$x \mapsto \Delta_{\text{prim}}(x - \hat{x}) \qquad y \mapsto \Delta_{\text{dual}}(y - \hat{y})$$

$$P \left\{ \begin{array}{rl} \max & c^T x \\ \text{s.t.} & Ax = b \\ & x \geqslant 0 \end{array} \right. \qquad \begin{array}{rl} \max & \Delta_{dual} \hat{c}^T x \\ \text{s.t.} & Ax = \Delta_{prim} \hat{b} \\ & x \geqslant -\Delta_{prim} \hat{x} \end{array} \right\} \hat{P}$$

$$y \mapsto y/\Delta_{\text{dual}} + \hat{y} \qquad x \mapsto x/\Delta_{\text{prim}} + \hat{x}$$



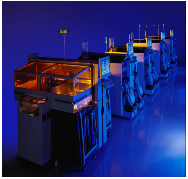


Siemens cooperation

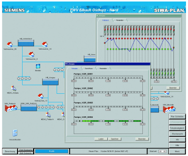
Longstanding Cooperation with department
Modeling, Simulation, Optimization

- first licensee (1996) of SoPlex
- steady use in various optimization modules

placement robots in
circuit board production



optimal planning of
water networks



History of the SCIP Optimization Suite

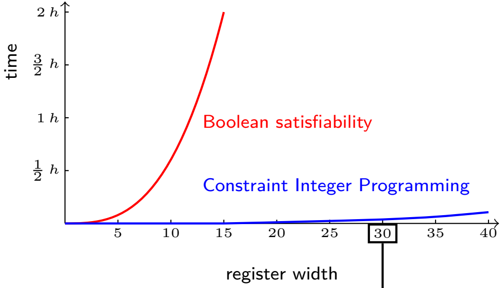
1996	SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])
1998	SIP – Solving Integer Programs (A. Martin [now U Erlangen])
10/2002	Beginning of SCIP development (T. Achterberg [now Gurobi])
08/2003	Chipdesign verification

Chipdesign verification

Goal: (computer-)proof, that a design is free of errors

Method: property checking using CIPs

Result:



constraints	422	152026
variables	3714	50756

Duration: 2003-2008

History of the SCIP Optimization Suite

1996	SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])
1998	SIP – Solving Integer Programs (A. Martin [now U Erlangen])
10/2002	Beginning of SCIP development (T. Achterberg [now Gurobi])
08/2003	Chipdesign verification
10/2004	ZIMPL – Zuse Inst. Math. Programming Language (T. Koch)

History of the SCIP Optimization Suite

1996	SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])
1998	SIP – Solving Integer Programs (A. Martin [now U Erlangen])
10/2002	Beginning of SCIP development (T. Achterberg [now Gurobi])
08/2003	Chipdesign verification
10/2004	ZIMPL – Zuse Inst. Math. Programming Language (T. Koch)
09/2005	First public version 0.8 of SCIP
09/2007	SCIP 1.0 release, ZIB Optimization Suite (SoPlex, SCIP, ZIMPL)

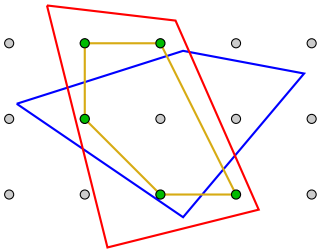
History of the SCIP Optimization Suite

1996	SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])
1998	SIP – Solving Integer Programs (A. Martin [now U Erlangen])
10/2002	Beginning of SCIP development (T. Achterberg [now Gurobi])
08/2003	Chipdesign verification
10/2004	ZIMPL – Zuse Inst. Math. Programming Language (T. Koch)
09/2005	First public version 0.8 of SCIP
09/2007	SCIP 1.0 release, ZIB Optimization Suite (SoPlex, SCIP, ZIMPL)
11/2008	Development of GCG started (G. G.)

GCG – a generic branch-cut-and-price solver

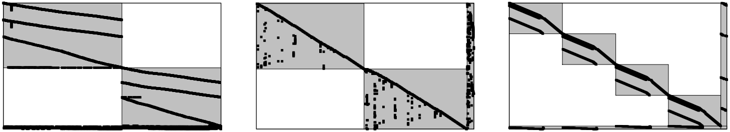
Goal of GCG:

- extend branch-cut-and-price framework SCIP to generic solver
- based on Dantzig-Wolfe decomposition
- easy use of branch-cut-and-price
- profit from powerful SCIP basics



How does it work?

- structure of problem provided or detected
- solve original and reformulated problem simultaneously
- pricing problems solved as general MIPs

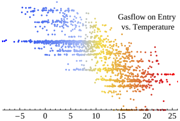


History of the SCIP Optimization Suite

1996	SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])
1998	SIP – Solving Integer Programs (A. Martin [now U Erlangen])
10/2002	Beginning of SCIP development (T. Achterberg [now Gurobi])
08/2003	Chipdesign verification
10/2004	ZIMPL – Zuse Inst. Math. Programming Language (T. Koch)
09/2005	First public version 0.8 of SCIP
09/2007	SCIP 1.0 release, ZIB Optimization Suite (SoPlex, SCIP, ZIMPL)
11/2008	Development of GCG started (G. G.)
01/2009	Gas transport optimization

Optimization of gas transport

Stochastic Mixed-Integer Nonlinear Constraint Program



over a **large** network.

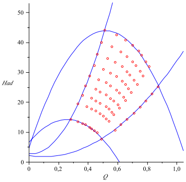
Start: 01/2009

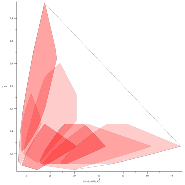
Goal:

- ▷ develop algorithms to solve such problems to “global optimality”!

Industry partner:

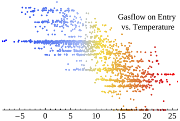
Open Grid Europe
The Gas Wheel





Optimization of gas transport

Stochastic Mixed-Integer Nonlinear Constraint Program



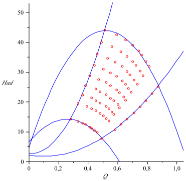
over a **large** network. Start: 01/2009

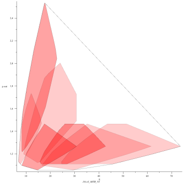
Goal:

- develop algorithms to solve such problems to "global optimality"!
- rather: attempt to integrate as many aspects as possible

Industry partner:

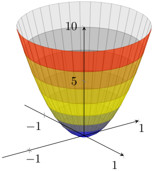
Open Grid Europe
The Gas Wheel

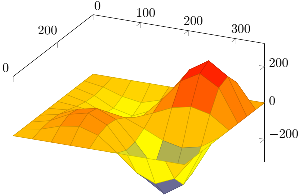




Mixed-Integer Nonlinear Programming

$$
\begin{array}{llll}
\min & c^t x & & \text{for } c \in \mathbb{R}^n, \\
\text{s. t.} & \mathbf{g_k(x) \leqslant 0} & & \text{for } k = 1, \ldots, m,\ g_k : [\ell, u] \to \mathbb{R} \in C^1, \\
& x \in [\ell, u], & & \\
& x_i \in \mathbb{Z} & & \text{for } i \in \mathcal{I} \subseteq \{1, \ldots, n\}.
\end{array}
$$





g_k **convex**

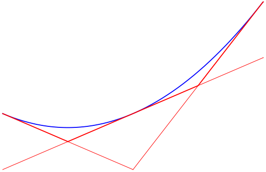
local = global optimality

g_k **nonconvex**

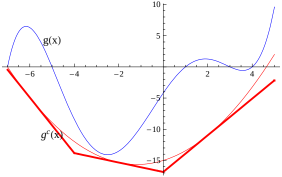
suboptimal local optima

MINLP solving techniques

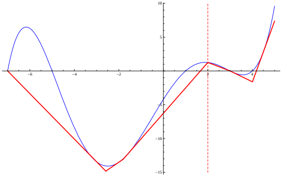
Gradient cuts



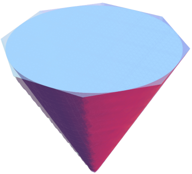
Underestimators



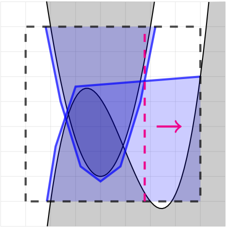
Spatial branching



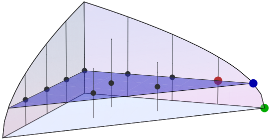
Presolving



Bound tightening



Primal heuristics



MINLP solving with SCIP

SCIP scope was extended over CIP in order to solve (nonconvex) MINLP:

CIP Definition:

When all integer variables fixed:

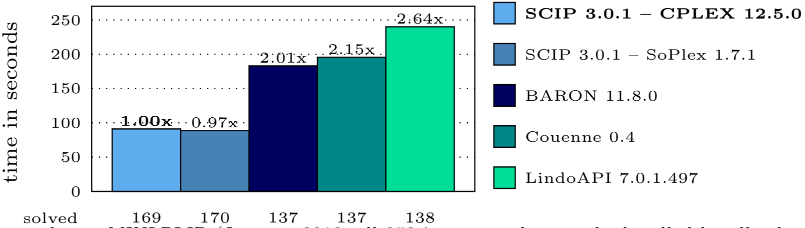
- ▷ CIP becomes an LP

SCIP solves:

When no branching was performed:

- ▷ remaining problem is tractable (LP/ conv. NLP)

MINLP performance of SCIP



results on MINLPLIB (January 2013, all 252 instances that can be handled by all solvers)

History of the SCIP Optimization Suite

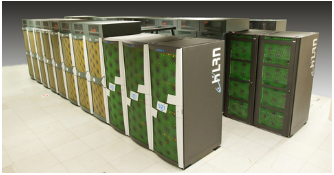
1996 SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])
1998 SIP – Solving Integer Programs (A. Martin [now U Erlangen])
10/2002 Beginning of SCIP development (T. Achterberg [now Gurobi])
08/2003 Chipdesign verification
10/2004 ZIMPL – Zuse Inst. Math. Programming Language (T. Koch)
09/2005 First public version 0.8 of SCIP
09/2007 SCIP 1.0 release, ZIB Optimization Suite (SoPlex, SCIP, ZIMPL)
11/2008 Development of GCG started (G. G.)
01/2009 Gas transport optimization
03/2009 Beginning of UG development (Y. Shinano)

ug[SCIP] — the parallel version of SCIP

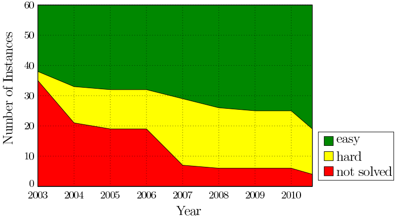
Some facts and results:

- shared ("FiberSCIP") and distributed memory version ("ParaSCIP")
- solves MIP and MINLP
- successful runs with up to 80.000 SCIP solvers
- solved 2 previously unsolved MIPLIB 2003 instances
  - ds: 4096 cores, about 76 hours, 3 billion nodes
  - stp3d: 7186 cores, about 33 hours, 10 million nodes (optimal solution given)
- and many MIPLIB 2010 instances

HLRN II:



MIPLIB 2003:

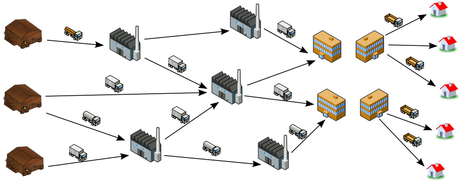


History of the SCIP Optimization Suite

1996	SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])
1998	SIP – Solving Integer Programs (A. Martin [now U Erlangen])
10/2002	Beginning of SCIP development (T. Achterberg [now Gurobi])
08/2003	Chipdesign verification
10/2004	ZIMPL – Zuse Inst. Math. Programming Language (T. Koch)
09/2005	First public version 0.8 of SCIP
09/2007	SCIP 1.0 release, ZIB Optimization Suite (SoPlex, SCIP, ZIMPL)
11/2008	Development of GCG started (G. G.)
01/2009	Gas transport optimization
03/2009	Beginning of UG development (Y. Shinano)
09/2009	Beale-Orchard-Hays Prize (T. Achterberg)
04/2010	Supply Chain management

Supply Chain Management

Huge, numerically challenging problems



Topics:

- overall LP performance
- improved numerical stability
- new presolving techniques
- decomposition approaches
- better branching schemes

Duration:

- 2010 – 2019 (at least)

Cooperation:

SAP

FAU

Presolving

- some of the regarded instances are huge
- MIPs often contain a lot of redundancy
- even more when a generic model is used for all types of supply chains

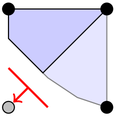
  $\rightarrow$ presolving

Presolving is one of the most important parts of a MIP solver:

- reduce problem size before the actual solving
- remove infeasible regions of the search space
- remove suboptimal regions of the search space

Two new presolvers

- dominated columns
- connected components



Presolving

- some of the regarded instances are huge
- MIPs often contain a lot of redundancy
- even more when a generic model is used for all types of supply chains

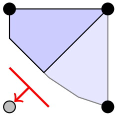
  $\rightarrow$ presolving

Presolving is one of the most important parts of a MIP solver:

- reduce problem size before the actual solving
- remove infeasible regions of the search space
- remove suboptimal regions of the search space

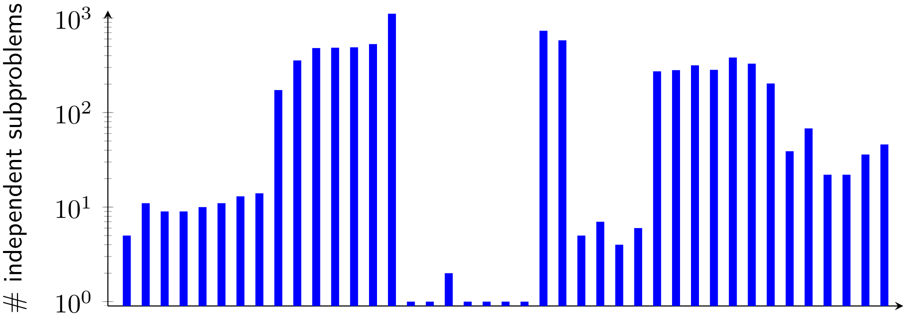
Two new presolvers

- dominated columns
- connected components



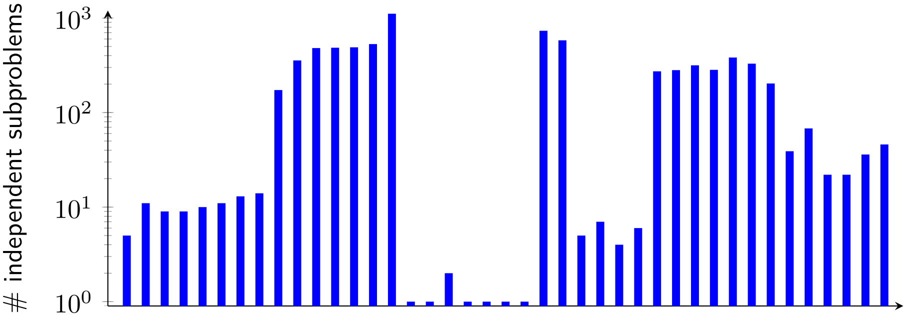
Independent Components

- supply chain of a company often has multiple independent sections



Independent Components

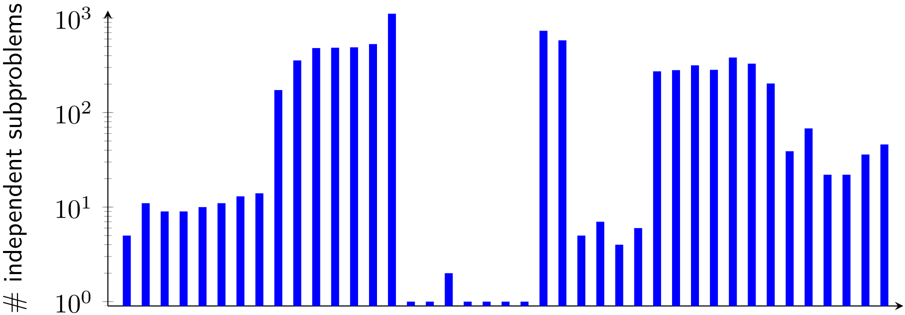
- supply chain of a company often has multiple independent sections



- MIP solving is $\mathcal{NP}$ hard $\rightarrow$ better solve them individually
- but: customer wants his complete supply chain in one model
- one global time limit
- some problems split up during presolving

Independent Components

- supply chain of a company often has multiple independent sections

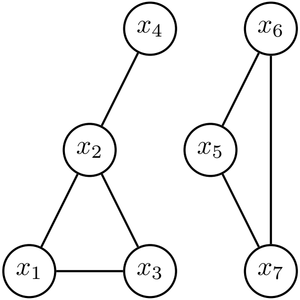


- MIP solving is $\mathcal{NP}$ hard $\rightarrow$ better solve them individually
- but: customer wants his complete supply chain in one model
- one global time limit
- some problems split up during presolving

  $\rightarrow$ solution: components presolver

The Components Presolver

$$\begin{array}{lrcrcrcl} c1: & x_1 & + & 2x_2 & - & x_3 & \leq & 5 \\ c2: & 3x_2 & + & x_4 & & & \geq & 3 \\ c3: & 3x_5 & + & 2x_6 & - & 5x_7 & \leq & 7 \end{array} \Rightarrow$$



The components presolver:

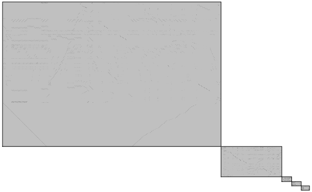
- MIP → (undirected) graph
  - variable → node
  - constraint → edges
- compute connected components
- solve "small" components during presolving
- remove solved components

Improvements:

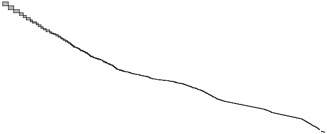
- solve components
  - in parallel
  - alternatingly
- branch to force splitting

Examples

Some Structures:

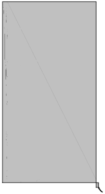


001-series: 5 components



002-series: 1164 components

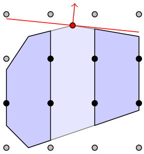
p2756 (MIPLIB 2003):
18 components



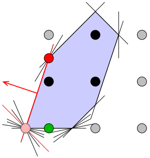
tanglegram2 (MIPLIB 2010):
37 components

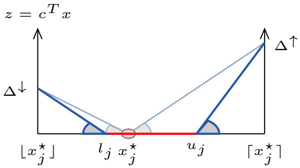
Branching Improvements

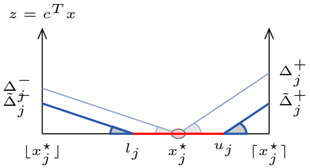
- Strong Branching with Domain Propagation
  - perform propagation within strong branching
  - improved predictions
  - reduces tree size + solving time



- Cloud Branching
  - expoit dual degeneracy
  - branch on "cloud of solutions"
  - reduce performance variability
  - save strong branching effort
  - improve reliability of pseudo costs







History of the SCIP Optimization Suite

1996	SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])
1998	SIP – Solving Integer Programs (A. Martin [now U Erlangen])
10/2002	Beginning of SCIP development (T. Achterberg [now Gurobi])
08/2003	Chipdesign verification
10/2004	ZIMPL – Zuse Inst. Math. Programming Language (T. Koch)
09/2005	First public version 0.8 of SCIP
09/2007	SCIP 1.0 release, ZIB Optimization Suite (SoPlex, SCIP, ZIMPL)
11/2008	Development of GCG started (G. G.)
01/2009	Gas transport optimization
03/2009	Beginning of UG development (Y. Shinano)
09/2009	Beale-Orchard-Hays Prize (T. Achterberg)
04/2010	Supply Chain management
12/2010	Google Research Award 2011
8/2012	Version 3.0, first releases of GCG, UG, SCIP-SDP
	SCIP Optimization Suite (SoPlex, SCIP, ZIMPL, GCG, UG)

History of the SCIP Optimization Suite

1996	SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])
1998	SIP – Solving Integer Programs (A. Martin [now U Erlangen])
10/2002	Beginning of SCIP development (T. Achterberg [now Gurobi])
08/2003	Chipdesign verification
10/2004	ZIMPL – Zuse Inst. Math. Programming Language (T. Koch)
09/2005	First public version 0.8 of SCIP
09/2007	SCIP 1.0 release, ZIB Optimization Suite (SoPlex, SCIP, ZIMPL)
11/2008	Development of GCG started (G. G.)
01/2009	Gas transport optimization
03/2009	Beginning of UG development (Y. Shinano)
09/2009	Beale-Orchard-Hays Prize (T. Achterberg)
04/2010	Supply Chain management
12/2010	Google Research Award 2011
8/2012	Version 3.0, first releases of GCG, UG, SCIP-SDP SCIP Optimization Suite (SoPlex, SCIP, ZIMPL, GCG, UG)
10/2014	Google Optimization uses SCIP

Google Optimization

https://developers.google.com/optimization/docs/lp

Google Developers

Google Optimization X Search

Sign in

Products > Google Optimization

Google Optimization

Report documentation issue | Take developer survey

- Overview
- Linear Optimization
  - Linear Optimization Overview
  - Linear Optimization Add-on for Google Sheets
  - Linear Optimization Service in Google Apps Script
  - Linear Optimization with Glop
- Puzzles
- Bin Packing and Cutting
- Traveling Salesman Problem

Linear Optimization

Linear optimization or *linear programming* is the name given to computing the best solution to a problem modeled as a set of linear relationships. These problems arise in many scientific and engineering disciplines. (The word "programming" is a bit of a misnomer, similar to how "computer" once meant "a person who computes." Here, "programming" refers to the arrangement of a plan, rather than programming in a computer language.)

Google provides three ways to solve linear optimization problems: the Linear Optimization add-on for Google Sheets, the Linear Optimization Service in Google Apps Script, and the open-source library Glop.

The Linear Optimization add-on for Google Sheets lets users solve linear-optimization problems by entering variables and constraints in a spreadsheet. Under the hood, it uses Apps Script's Linear Optimization Service.

The Linear Optimization Service in Google Apps Script lets developers make function calls to solve linear optimization problems. It relies on Glop for pure linear-optimization problems where all variables can take on real values. If any variables are constrained to integers, the service uses SCIP, from Zuse-Institut Berlin.

Glop is Google's in-house linear solver, available as open source.

To be continued...

- Which application do **you** want to solve with SCIP?
- Download SCIP and try it out!
- Register for the mailing list: `scip@zib.de`
- Join all these SCIP users: