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

\[
\begin{align*}
\min & \quad c^T x \\
\text{s.t.} & \quad Ax \leq b \\
& (x_I, x_C) \in \mathbb{Z}^I \times \mathbb{R}^C
\end{align*}
\]

**Constraint Program**

Objective function:
▷ arbitrary function

Feasible set:
▷ given by arbitrary constraints

Variable domains:
▷ arbitrary (usually finite)

\[
\begin{align*}
\min & \quad c(x) \\
\text{s.t.} & \quad x \in F \\
& (x_I, x_N) \in \mathbb{Z}^I \times X
\end{align*}
\]
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

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

Remark:
- arbitrary objective or variables modeled by constraints
Constraint Integer Programming

- Mixed Integer Programs

\[ \text{MIP} \]

"Every MIP is a CIP."

"Every CP over a finite domain space is a CIP."

Gerald Gamrath (ZIB): The SCIP Optimization Suite
Constraint Integer Programming

- Mixed Integer Programs
- SAT satisfiability problems

Relation to CP and MIP

- Every MIP is a CIP. \( \text{MIP} \subseteq \text{CIP} \)
- Every CP over a finite domain space is a CIP. \( \text{FD} \subseteq \text{CIP} \)
Constraint Integer Programming

- Mixed Integer Programs
- SAT isfiability problems
- Pseudo-Boolean Optimization

MIP ⊊ CIP
FD ⊊ CIP

Gerald Gamrath (ZIB): The SCIP Optimization Suite
Constraint Integer Programming

- Mixed Integer Programs
- SAT isfiability problems
- Pseudo-Boolean Optimization
- Finite Domain

\[ \text{MIP} \subseteq \text{CIP} \]
\[ \text{FD} \subseteq \text{CIP} \]
Constraint Integer Programming

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

Relation to CP and MIP

- Every MIP is a CIP.
- "MIP $\subseteq$ CIP"

- Every CP over a finite domain space is a CIP.
- "FD $\subseteq$ CIP"
Constraint Integer Programming

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

Relation to CP and MIP

- Every MIP is a CIP.
- “MIP $\subseteq$ CIP”
- Every CP over a finite domain space is a CIP.
- “FD $\subseteq$ CIP”

Gerald Gamrath (ZIB): The SCIP Optimization Suite 6
Constraint Integer Programming

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

Relation to CP and MIP

- Every MIP is a CIP. \( MIP \subseteq CIP \)
- Every CP over a finite domain space is a CIP. \( FD \subseteq CIP \)
CIP Solving Techniques

Presolving

Primal Heuristics

Branch & Bound

Cutting Planes

Domain Propagation

Conflict Analysis

Gerald Gamrath (ZIB): The SCIP Optimization Suite
How do we solve CIPs?

MIP
- LP relaxation
- cutting planes

CP
- domain propagation

SAT
- conflict analysis
- periodic restarts

MIP, CP, and SAT
- branch-and-bound

SCIP
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
SCIP 3.1: Performance

- fastest non-commercial MIP solver

![Graph showing performance comparison between non-commercial and commercial solvers.]

- versionwise performance improvements

![Graph showing versionwise performance improvements.]

CBC 2.8.10  
SCIP 3.1.0 – CLP 1.15.6  
SCIP 3.1.0 – SoPlex 2.0.0  
SCIP 3.1.0 – Cplex 12.6.0  
Xpress 7.8.0  
Gurobi 6.0.0  
Cplex 12.6.1  
data: Hans Mittelmann  
graphics: ZIB
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

\[
\begin{align*}
P &= \max \quad c^T x \\
\text{s.t.} \quad Ax &= b \\
x &\geq 0
\end{align*}
\]

\[
\begin{align*}
\hat{P} &= \max \quad \Delta_{\text{dual}} \hat{c}^T x \\
\text{s.t.} \quad Ax &= \Delta_{\text{prim}} \hat{b} \\
x &\geq -\Delta_{\text{prim}} \hat{x}
\end{align*}
\]
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:**

- Boolean satisfiability
- Constraint Integer Programming

---

**Duration:** 2003-2008

**Constraints:**
- 422
- 152,026

**Variables:**
- 3,714
- 50,756

---

Gerald Gamrath (ZIB): The SCIP Optimization Suite
### History of the SCIP Optimization Suite

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

Goal:

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

Industry partner:

Open Grid Europe
The Gas Wheel

Start: 01/2009
Optimization of gas transport

Stochastic Mixed-Integer Nonlinear Constraint Program

over a large network.

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

Start: 01/2009
Mixed-Integer Nonlinear Programming

\[
\begin{align*}
\min & \quad c^T x \\
\text{s. t.} & \quad g_k(x) \leq 0 \\
& \quad x \in [\ell, u], \\
& \quad x_i \in \mathbb{Z} \quad \text{for } i \in \mathcal{I} \subseteq \{1, \ldots, n\}.
\end{align*}
\]

for \( c \in \mathbb{R}^n \),

for \( k = 1, \ldots, m \), \( g_k : [\ell, u] \rightarrow \mathbb{R} \in C^1 \),

\( g_k \) 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

[Graph showing performance comparison among solvers]

results on MINLPLIB (January 2013, all 252 instances that can be handled by all solvers)
### History of the SCIP Optimization Suite

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1996</td>
<td>SoPlex – Sequential obj. simPlex (R. Wunderling [now IBM])</td>
</tr>
<tr>
<td>1998</td>
<td>SIP – Solving Integer Programs (A. Martin [now U Erlangen])</td>
</tr>
<tr>
<td>10/2002</td>
<td>Beginning of SCIP development (T. Achterberg [now Gurobi])</td>
</tr>
<tr>
<td>08/2003</td>
<td>Chipdesign verification</td>
</tr>
<tr>
<td>09/2005</td>
<td>First public version 0.8 of SCIP</td>
</tr>
<tr>
<td>09/2007</td>
<td>SCIP 1.0 release, <strong>ZIB Optimization Suite</strong> (SoPlex, SCIP, ZIMPL)</td>
</tr>
<tr>
<td>11/2008</td>
<td>Development of GCG started (G. G.)</td>
</tr>
<tr>
<td>01/2009</td>
<td>Gas transport optimization</td>
</tr>
<tr>
<td>03/2009</td>
<td>Beginning of UG development (Y. Shinano)</td>
</tr>
</tbody>
</table>
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
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
  → 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
  → 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

![Bar graph](image-url)

- vertical axis: # independent subproblems
- horizontal axis: Logarithmic scale
Independent Components

- supply chain of a company often has multiple independent sections

MIP solving is \(NP\) 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

\[\begin{align*}
c_1 : & \quad x_1 + 2x_2 - x_3 \leq 5 \\
c_2 : & \quad 3x_2 + x_4 \geq 3 \\
c_3 : & \quad 3x_5 + 2x_6 - 5x_7 \leq 7
\end{align*}\]

The components presolver:
- MIP \rightarrow (undirected) graph
  - variable \rightarrow node
  - constraint \rightarrow 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**
  - exploit dual degeneracy
  - branch on “cloud of solutions”
  - reduce performance variability
  - save strong branching effort
  - improve reliability of pseudo costs

\[ z = c^T x \]

\[ \Delta \downarrow \]

\[ \Delta \uparrow \]

\[ l_j \quad x_j^* \quad u_j \quad [x_j^*] \]
## History of the SCIP Optimization Suite

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

**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
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
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: