Mathematics of Infrastructure Planning

ZIB Optimization Suite

Stefan Heinz

Zuse Institute Berlin (ZIB)

DFG Research Center MATHEON
Mathematics for key technologies

16/April/2012

ZIB Optimization Suite = SCIP + SoPlex + ZIMPL

Toolbox for generating and solving constraint integer programs

ZIMPL

- a mixed integer programming modeling language
- easily generate LPs, MIPs, and ...

SCIP

- a MIP and CP solver, branch-cut-and-price framework
- ZIMPL models can directly be loaded into SCIP and solved

SoPlex

- a linear programming solver
- SCIP uses SoPlex as underlying LP solver

ZIMPL – Modeling Language

- distinguish between data and model
- easily generate LPs, MIPs, and ...
- fast prototyping
- `http://zimpl.zib.de`
- AIMMS, AMPL, GAMS, MOSEL, OPL, . . .

SoPlex – Linear Programming Solver

- dual and primal simplex
- has a warm start
- `http://soplex.zib.de`
- CLP, CPLEX, GUROBI, MOSEK, XPRESS, . . .

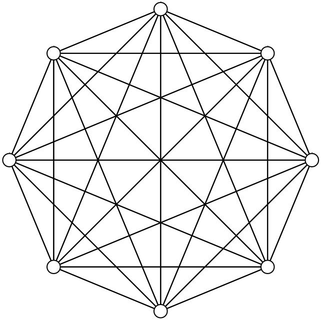
SCIP is a framework for Constraint Integer Programming oriented towards the needs of Mathematical Programming experts who want to have total control of the solution process and access detailed information down to the guts of the solver.

- framework to solve constraint integer programs
- branch-and-bound framework
- branch-and-cut framework
- branch-and-propagate framework
- branch-and-price framework
- black box MIP solver
- http://scip.zib.de
- CBC, CPLEX, GUROBI, MOSEK, XPRESS, ...

Definition (TSP)

Given a complete graph $G = (V, E)$ and distances d_e for all $e \in E$:

Find a Hamiltonian cycle (cycle containing all nodes, tour) of minimum length.



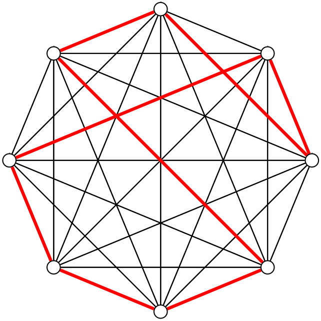
K_8

An example: the Traveling Salesman Problem

Definition (TSP)

Given a complete graph $G = (V, E)$ and distances d_e for all $e \in E$:

Find a Hamiltonian cycle (cycle containing all nodes, tour) of minimum length.



K_8

An example: the Traveling Salesman Problem

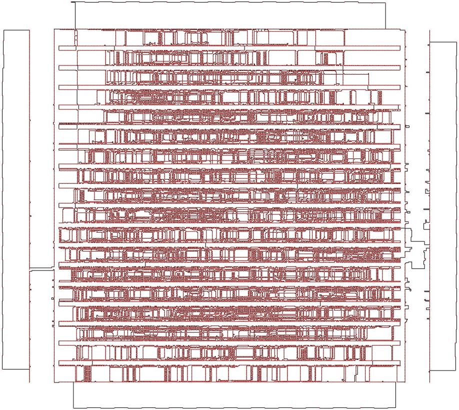
Definition (TSP)

Given a complete graph $G = (V, E)$ and distances d_e for all $e \in E$:

Find a Hamiltonian cycle (cycle containing all nodes, tour) of minimum length.

$\frac{(n-1)!}{2}$ possible solutions: finite, but enumeration intractable!

TSP world record



by Bill Cook et al.

The largest solved instance of the traveling salesman problem consists of a tour through **85,900 cities** in a VLSI application that arose in Bell Laboratories in the late 1980s.

The total amount of computer usage for the computations was appx. **136 CPU years**.

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

Constraint Program

Objective function:

- ▷ arbitrary function

Feasible set:

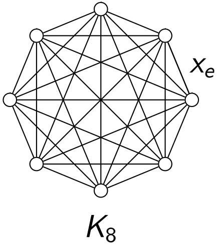
- ▷ given by arbitrary constraints

Variable domains:

- ▷ arbitrary (usually finite)

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

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



Given

- complete graph $G = (V, E)$
- distances $d_e > 0$ for all $e \in E$

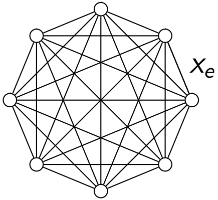
Binary variables

- $x_e = 1$ if edge e is used

TSP – Integer Programming Formulation

Given

- complete graph $G = (V, E)$
- distances $d_e > 0$ for all $e \in E$



K_8

Binary variables

- $x_e = 1$ if edge e is used

$$
\begin{aligned}
\min \quad & \sum_{e \in E} d_e \, x_e & \\
\text{subject to} \quad & \sum_{e \in \delta(v)} x_e = 2 & \forall v \in V \\
& \sum_{e \in \delta(S)} x_e \geq 2 & \forall S \subset V, S \neq \varnothing \\
& x_e \in \{0, 1\} & \forall e \in E
\end{aligned}
$$

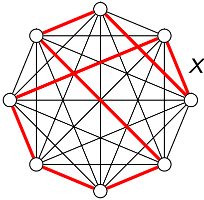
TSP – Integer Programming Formulation

Given

- complete graph $G = (V, E)$
- distances $d_e > 0$ for all $e \in E$

Binary variables

- $x_e = 1$ if edge e is used



K_8

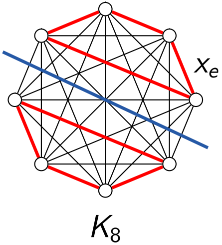
$$\min \sum_{e \in E} d_e \, x_e$$

subject to	$\sum_{e \in \delta(v)} x_e = 2$	$\forall v \in V$	node degree
	$\sum_{e \in \delta(S)} x_e \geq 2$	$\forall S \subset V, S \neq \emptyset$	
	$x_e \in \{0, 1\}$	$\forall e \in E$	

TSP – Integer Programming Formulation

Given

- complete graph $G = (V, E)$
- distances $d_e > 0$ for all $e \in E$



Binary variables

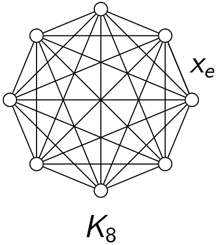
- $x_e = 1$ if edge e is used

$$\begin{aligned} \min \quad & \sum_{e \in E} d_e \, x_e & \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e = 2 & \forall v \in V \\ & \sum_{e \in \delta(S)} x_e \geq 2 & \forall S \subset V, S \neq \varnothing \\ & & \text{subtour elimination} \\ & x_e \in \{0, 1\} & \forall e \in E \end{aligned}$$

TSP – Integer Programming Formulation

Given

- complete graph $G = (V, E)$
- distances $d_e > 0$ for all $e \in E$



Binary variables

- $x_e = 1$ if edge e is used

$$\min \sum_{e \in E} d_e \, x_e \qquad \text{distance}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{e \in \delta(v)} x_e = 2 && \forall v \in V \\
& \sum_{e \in \delta(S)} x_e \geq 2 && \forall S \subset V, S \neq \emptyset \\
& x_e \in \{0, 1\} && \forall e \in E
\end{aligned}$$

TSP – Integer Programming Formulation

Given

- complete graph $G = (V, E)$
- distances $d_e > 0$ for all $e \in E$

Binary variables

- $x_e = 1$ if edge e is used

$$
\begin{aligned}
\min \quad & \sum_{e \in E} d_e \, x_e \\
\text{subject to} \quad & \sum_{e \in \delta(v)} x_e = 2 && \forall v \in V \\
& \sum_{e \in \delta(S)} x_e \geq 2 && \forall S \subset V, S \neq \emptyset \\
& x_e \in \{0, 1\} && \forall e \in E
\end{aligned}
$$

Given

- complete graph $G = (V, E)$
- distances $d_e > 0$ for all $e \in E$

Binary variables

- $x_e = 1$ if edge e is used

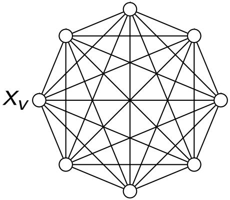
$$\begin{aligned} \min \quad & \sum_{e \in E} d_e \, x_e & \\ \text{subject to} \quad & \sum_{e \in \delta(v)} x_e = 2 & \forall v \in V \\ & \sum_{e \in \delta(S)} x_e \geq 2 & \forall S \subset V, S \neq \emptyset \\ & x_e \in \{0, 1\} & \forall e \in E \end{aligned}$$

Given

- ▷ complete graph $G = (V, E)$
- ▷ for each $e \in E$ a distance $d_e > 0$

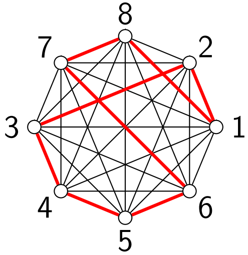
Integer variables

- ▷ x_v position of $v \in V$ in tour



K_8

TSP – Constraint Programming Formulation



Given

- complete graph $G = (V, E)$
- for each $e \in E$ a distance $d_e > 0$

Integer variables

- x_v position of $v \in V$ in tour

$$
\begin{aligned}
\min \quad & \text{length}(x_1, \ldots, x_n) & \\
\text{subject to} \quad & \text{alldifferent}(x_1, \ldots, x_n) & \\
& x_v \in \{1, \ldots, n\} & \forall v \in V
\end{aligned}
$$

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

After fixing all integer variables:

- 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 Program

Objective function:

- ▷ linear function

Feasible set:

- ▷ described by arbitrary constraints

Variable domains:

- ▷ real or integer values

After fixing all integer variables:

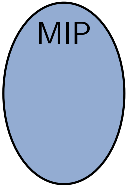
- ▷ CIP becomes an LP

$$
\begin{aligned}
\min \quad & \sum_{e \in E} d_e \, x_e \\
s.t. \quad & \sum_{e \in \delta(v)} x_e = 2 && \forall\, v \in V \\
& \text{nosubtour}(x) \\
& x_e \in \{0,1\} && \forall\, e \in E
\end{aligned}
$$

(CIP formulation of TSP)

Single nosubtour constraint rules out subtours (e.g. by domain propagation). It may also separate subtour elimination inequalities.

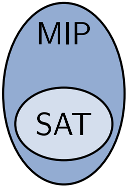
- **M**ixed **I**nteger **P**rograms



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"*

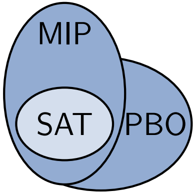
- **M**ixed **I**nteger **P**rograms
- **SAT**isfiability problems



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"*

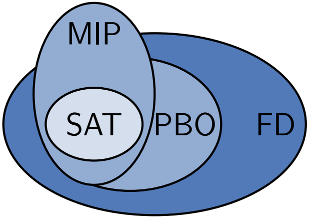
- Mixed Integer Programs
- SATisfiability problems
- Pseudo-Boolean Optimization



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"*

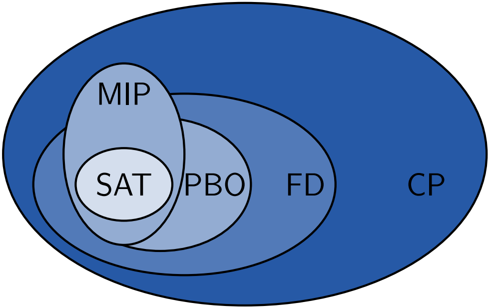
- **M**ixed **I**nteger **P**rograms
- **SAT**isfiability problems
- **P**seudo-**B**oolean **O**ptimization
- **F**inite **D**omain



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"*

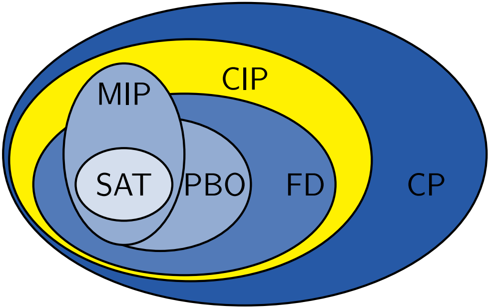
- Mixed Integer Programs
- SATisfiability problems
- Pseudo-Boolean Optimization
- Finite Domain
- Constraint Programming



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"*

- 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"*

MIP

- LP relaxation
- cutting planes

MIP, CP, and SAT

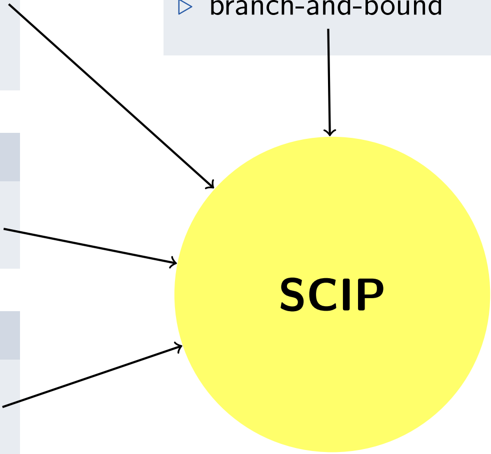
- branch-and-bound

CP

- domain propagation

SAT

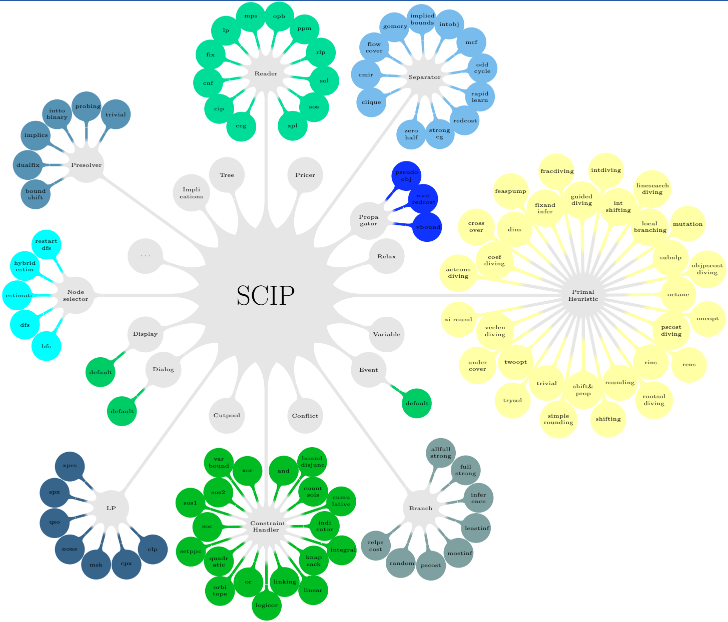
- conflict analysis
- periodic restarts

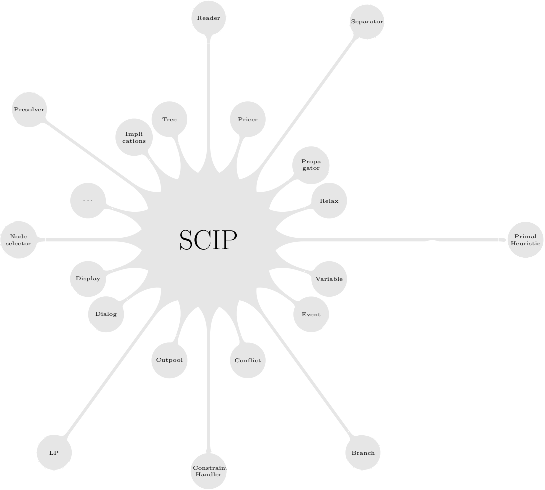


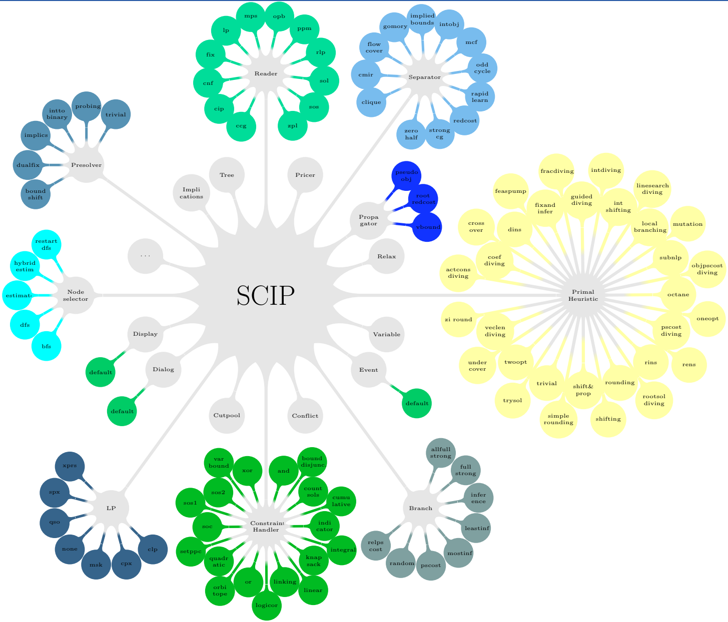
SCIP (Solving Constraint Integer Programs) ...

- is a branch-cut-and-price framework,
- is constraint based,
- incorporates
  - CP features (domain propagation),
  - MIP features (cutting planes, LP relaxation), and
  - SAT-solving features (conflict analysis, restarts),
- has a modular structure via plugins,
- provides a full-scale MIP solver,
- is free for academic purposes,
- and is available in source-code under `http://scip.zib.de` !

Structure of SCIP




SCIP
Reader
Separator
Presolver
Tree
Pricer
Impli cations
Propa gator
...
Relax
Node selector
Primal Heuristic
Display
Variable
Dialog
Event
Cutpool
Conflict
LP
Constraint Handler
Branch


SCIP
Reader
mps
opb
ppm
rlp
sol
sos
zpl
ccg
cip
cnf
fix
lp
Separator
implied bounds
intobj
mcf
odd cycle
rapid learn
redcost
strong cg
zero half
clique
cmir
flow cover
gomory
Presolver
probing
trivial
intto binary
implics
dualfix
bound shift
Tree
Pricer
Impli cations
Propa gator
pseudo obj
root redcost
vbound
Relax
...
Node selector
restart dfs
hybrid estim
estimate
dfs
bfs
Primal Heuristic
fracdiving
intdiving
linesearch diving
feaspump
fixand infer
guided diving
int shifting
cross over
dins
local branching
mutation
coef diving
subnlp
objpscost diving
actcons diving
octane
oneopt
zi round
veclen diving
pscost diving
under cover
twoopt
rins
rens
trivial
shift& prop
rounding
trysol
rootsol diving
simple rounding
shifting
Display
default
Dialog
default
Variable
Event
default
Cutpool
Conflict
LP
xprs
spx
qso
none
msk
cpx
clp
Constraint Handler
var bound
xor
and
bound disjunc.
sos2
count sols
sos1
cumu lative
soc
indi cator
setppc
integral
quadr atic
knap sack
orbi tope
or
linking
linear
logicor
Branch
allfull strong
full strong
infer ence
leastinf
mostinf
pscost
random
relps cost

History of SCIP

1998	SIP – Solving Integer Programs (Alexander Martin)
10/2002	Start of SCIP development (Tobias Achterberg)
02/2003	First version to solve MIPs
09/2005	First public version 0.80
09/2006	Version 0.90
07/2007	Tobias Achterberg left the SCIP developer team
09/2007	Version 1.00
09/2007	Part of the ZIB Optimization Suite
09/2008	Version 1.1.0
10/2008	Nonlinear support
09/2009	Version 1.2.0
05/2010	First global constraints
09/2010	Version 2.0
10/2012	Version 2.1

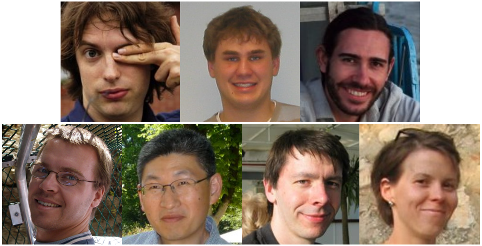
Number of Default Plugins

	SCIP version							
plugin type	0.7	0.8	0.9	1.0	1.1	1.2	2.0	2.1
branching rules	6	7	8	8	8	8	8	8
constraint handlers	10	10	11	11	14	16	23	25
node selectors	3	7	3	5	5	5	5	5
presolvers	2	7	5	5	6	6	6	5
primal heuristics	9	14	21	23	24	27	32	33
propagators	0	1	2	2	2	2	3	5
readers	2	4	6	6	11	13	15	16
separators	3	6	7	8	10	10	12	13

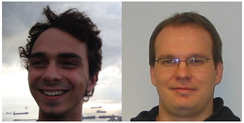
Involved Persons

- Thorsten Koch
- Marc Pfetsch (TU Darmstadt)

- Timo Berthold
- Gerald Gamrath
- Ambros Gleixner
- Stefan Heinz
- Yuji Shinano
- Stefan Vigerske
- Kati Wolter

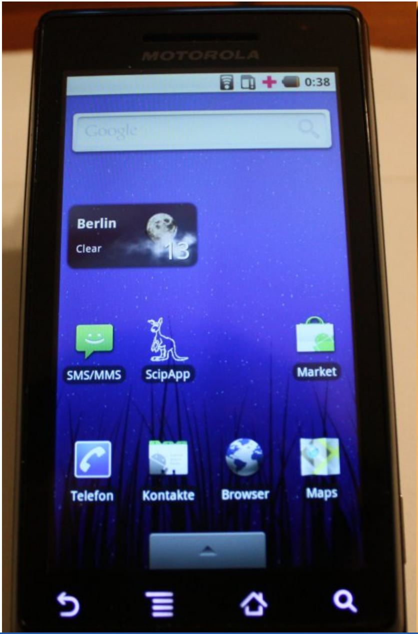


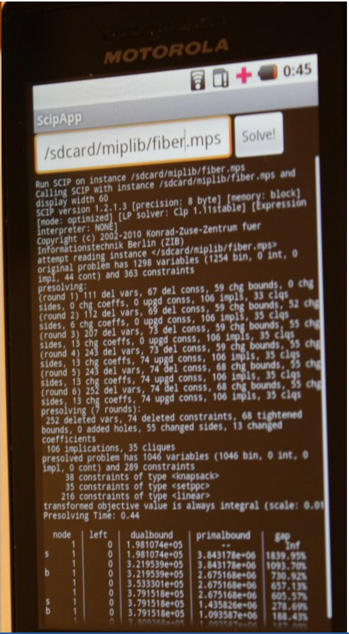
- Gregor Hendel
- Michael Winkler



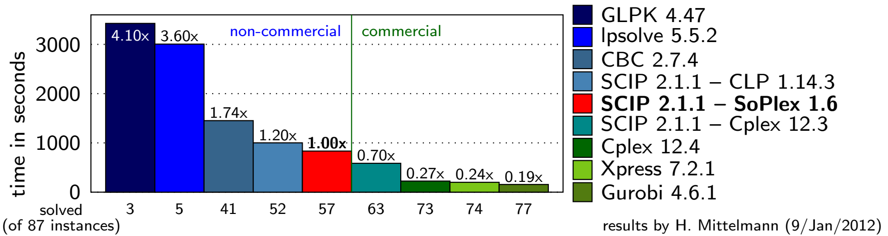
SCIP Facts

- more than 300 000 lines of C code
  → 18% documentation
  → 20% assertions
- 7 examples illustrating the use of SCIP
- HowTos: each plugin type, debugging, automatic testing, . . .
- C++ wrapper classes, (experimental) python interface
- 7 interfaces to external linear programming solvers
  → CLP, CPLEX, Gurobi, Mosek, QSopt, SoPlex, XPRESS
- 10 different input formats
  → cip, cnf, flatzinc, rlp, lp, mps, opb, pip, wbo, zimpl
- more than 1000 parameters, 15 "emphasis" settings
- active mailing list
- runs on Linux, Windows, Mac (Darwin+PPC), SunOS, . . .

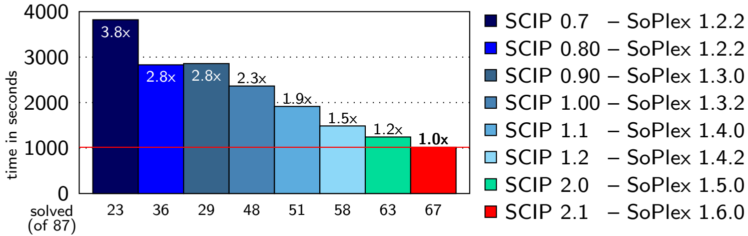

MOTOROLA
0:38
Google
Berlin
Clear
13
SMS/MMS
ScipApp
Market
Telefon
Kontakte
Browser
Maps


MOTOROLA
0:45
ScipApp
/sdcard/miplib/fiber.mps
Solve!
Run SCIP on instance /sdcard/miplib/fiber.mps
Calling SCIP with instance /sdcard/miplib/fiber.mps and
display width 60
SCIP version 1.2.1.3 [precision: 8 byte] [memory: block]
[mode: optimized] [LP solver: Clp 1.11stable] [Expression
interpreter: NONE]
Copyright (c) 2002-2010 Konrad-Zuse-Zentrum fuer
Informationstechnik Berlin (ZIB)
attempt reading instance </sdcard/miplib/fiber.mps>
original problem has 1298 variables (1254 bin, 0 int, 0
impl, 44 cont) and 363 constraints
presolving:
(round 1) 111 del vars, 67 del conss, 59 chg bounds, 0 chg
sides, 0 chg coeffs, 0 upgd conss, 106 impls, 33 clqs
(round 2) 112 del vars, 69 del conss, 59 chg bounds, 52 chg
sides, 6 chg coeffs, 0 upgd conss, 106 impls, 35 clqs
(round 3) 207 del vars, 73 del conss, 59 chg bounds, 55 chg
sides, 13 chg coeffs, 0 upgd conss, 106 impls, 35 clqs
(round 4) 243 del vars, 73 del conss, 59 chg bounds, 55 chg
sides, 13 chg coeffs, 74 upgd conss, 106 impls, 35 clqs
(round 5) 243 del vars, 74 del conss, 68 chg bounds, 55 chg
sides, 13 chg coeffs, 74 upgd conss, 106 impls, 35 clqs
(round 6) 252 del vars, 74 del conss, 68 chg bounds, 55 chg
sides, 13 chg coeffs, 74 upgd conss, 106 impls, 35 clqs
presolving (7 rounds):
252 deleted vars, 74 deleted constraints, 68 tightened
bounds, 0 added holes, 55 changed sides, 13 changed
coefficients
106 implications, 35 cliques
presolved problem has 1046 variables (1046 bin, 0 int, 0
impl, 0 cont) and 289 constraints
38 constraints of type <knapsack>
35 constraints of type <setppc>
216 constraints of type <linear>
transformed objective value is always integral (scale: 0.0
Presolving Time: 0.44
node | left | dualbound | primalbound | gap
1 | 0 | 1.981074e+05 | -- | Inf
s 1 | 0 | 1.981074e+05 | 3.843178e+06 | 1839.95%
1 | 0 | 3.219539e+05 | 3.843178e+06 | 1093.70%
b 1 | 0 | 3.219539e+05 | 2.675168e+06 | 730.92%
1 | 0 | 3.533301e+05 | 2.675168e+06 | 657.13%
1 | 0 | 3.791518e+05 | 2.675168e+06 | 605.57%
s 1 | 0 | 3.791518e+05 | 1.435826e+06 | 278.69%
b 1 | 0 | 3.791518e+05 | 1.093587e+06 | 188.43%

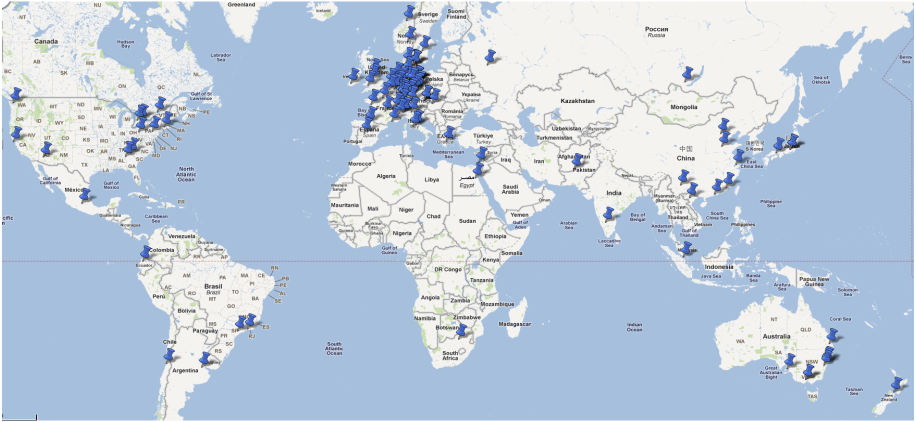
- fastest non-commercial MIP solver



results by H. Mittelmann (9/Jan/2012)



Some universities and institutes using the ZIB Optimization Suite:



Linux and Mac users

- download the ZIB Optimization Suite 2.0.1
  `http://zibopt.zib.de`
- read the `INSTALL`
  - `tar xvf ziboptsuite-2.0.1.tgz`
  - `cd ziboptsuite-2.0.1`
  - `make`
  - `make test`
- requirements: `readline` and `zlib`
- you can also use the virtual machine (see next slide)

Windows user

- download the virtual machine (VM) form the course web page
  - `CIPvmware.zip` (Attention 2,7 GB)
  - >1 GB main memory
  - 5–8 GB disk space
- follow the instruction stated in the `README.txt`
  - download the VMware Player (free software)
  - load the VM into the VMware Player
  - power on the VM
- ZIB Optimization Suite is already installed
- Eclipse, emacs, LaTeX, JAVA, Kate, Kile, ...

Mathematics of Infrastructure Planning

ZIB Optimization Suite

Stefan Heinz

Zuse Institute Berlin (ZIB)