Basic concepts of SCIP

CO@Work Berlin

Stefan Heinz

MATHEON/ZIB

09/22/2009
Outline

SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

What is SCIP?

Solving Constraint Integer Programs

Summary and Outlook
SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

What is SCIP?

Solving Constraint Integer Programs

Summary and Outlook
Traveling Salesman Problem (TSP)

**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.

$K_8$
Traveling Salesman Problem (TSP)

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.
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.
Traveling Salesman Problem (TSP)

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.
Traveling Salesman Problem (TSP)

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.
Mona Lisa TSP Challenge

- 100,000 nodes
- Best solution:
  Tour: 5,757,191
  Bound: 5,757,025
  Gap: 166 (0.0029%)

$100 prize to the first person to find a tour shorter than 5,757,191.

http://www.tsp.gatech.edu/data/ml/monalisa.html
### Mixed Integer Program (MIP)

#### Characteristics

**Objective function:**
- linear function

**Feasible region:**
- described by linear constraints

**Variable domains:**
- real or integer values

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

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

Binary variables
- $x_e = 1$ if edge $e$ is used
TSP – Integer Programming Formulation

Given

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

Binary variables

- $x_e = 1$ if edge $e$ is used

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

Given

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

Binary variables

- $x_e = 1$ if edge $e$ is used

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

Given

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

Binary variables

- \( x_e = 1 \) if edge \( e \) is used

\[
\begin{align*}
\text{min} & \quad \sum_{e \in E} d_e x_e \\
\text{subject to} & \quad \sum_{e \in \delta(v)} x_e = 2 \quad \forall v \in V \\
& \quad \sum_{e \in \delta(S)} x_e \geq 2 \quad \forall S \subset V, S \neq \emptyset \\
& \quad x_e \in \{0, 1\} \quad \forall e \in E
\end{align*}
\]
TSP – Integer Programming Formulation

Given

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

Binary variables

- \( x_e = 1 \) if edge \( e \) is used

\[
\begin{align*}
\text{min} & \quad \sum_{e \in E} d_e x_e \\
\text{subject to} & \quad \sum_{e \in \delta(v)} x_e = 2 \quad \forall v \in V \\
& \quad \sum_{e \in \delta(S)} x_e \geq 2 \quad \forall S \subset V, S \neq \emptyset \\
& \quad x_e \in \{0, 1\} \quad \forall e \in E
\end{align*}
\]
Constraint Program (CP)

Characteristics

Objective function:
- arbitrary function

Feasible region:
- described by arbitrary constraints

Variable domains:
- discrete values

\[
\begin{align*}
\text{min} & \quad c(x) \\
\text{s.t.} & \quad x \in F \\
& \quad x \in \mathbb{Z}^n
\end{align*}
\]
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
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

\[
\begin{align*}
\min & \quad \text{length}(x_1, \ldots, x_n) \\
\text{subject to} & \quad \text{alldifferent}(x_1, \ldots, x_n) \\
& \quad x_v \in \{1, \ldots, n\} \\
& \quad \forall v \in V
\end{align*}
\]
Traveling Salesman Problem

- Given a complete graph $G = (V, E)$ and distances $d_e$.
- Find a Hamiltonian cycle (cycle containing all nodes, tour) with minimum length.

**MIP Formulation**

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

$$\text{s.t.} \sum_{e \in \delta(v)} x_e = 2 \quad \forall v \in V$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall S \subset V, \quad S \neq \emptyset$$

$$x_e \in \{0, 1\} \quad \forall e \in E$$

**CP Formulation**

$$\min \text{length}(x_1, \ldots, x_n)$$

$$\text{s.t.} \quad \text{alldifferent}(x_1, \ldots, x_n)$$

$$x_v \in \{1, \ldots, n\}$$
## Constraint Integer Program (CIP)

### Characteristics

**Objective function:**
- linear function

**Feasible region:**
- described by arbitrary constraints
- after fixing all integer vars: CIP becomes an linear program (LP)

**Variable domains:**
- real or integer values

### Mathematical Formulation

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

Characteristics

Objective function:
- linear function

Feasible region:
- described by arbitrary constraints
- after fixing all integer vars: CIP becomes an linear program (LP)

Variable domains:
- real or integer values

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

(CIP formulation of TSP)

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

- TSPLIB
  
  http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html

- 108 Instances
- Time limit 1800 seconds
- Memory limit 6 GB
- Flow based IP formulation versus CIP

<table>
<thead>
<tr>
<th></th>
<th>CPLEX</th>
<th>SCIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Solved</td>
<td>33</td>
<td>64</td>
</tr>
<tr>
<td>Total Time [h]</td>
<td>40</td>
<td>34</td>
</tr>
<tr>
<td>Geom. Time [s]</td>
<td>475.0</td>
<td>164.7</td>
</tr>
</tbody>
</table>
Computational Results

- **TSPLIB**
  - [TSPLIB](http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html)
  - 108 Instances
  - Time limit 1800 seconds
  - Memory limit 6 GB
  - Flow based IP formulation versus CIP

<table>
<thead>
<tr>
<th></th>
<th>CPLEX</th>
<th>SCIP</th>
<th>Concorde</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Solved</td>
<td>33</td>
<td>64</td>
<td>108</td>
</tr>
<tr>
<td>Total Time [h]</td>
<td>40</td>
<td>34</td>
<td>&lt;2</td>
</tr>
<tr>
<td>Geom. Time [s]</td>
<td>475.0</td>
<td>164.7</td>
<td>3.0</td>
</tr>
</tbody>
</table>

[Concorde](http://www.tsp.gatech.edu/concorde.html)
Constraint Integer Programming

- Mixed Integer Programs

MIP
Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
- Pseudo-Boolean

Diagram:
- MIP
- SAT
- PB
Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
- Pseudo-Boolean
- Finite Domain
Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
- Pseudo-Boolean
- Finite Domain
- Constraint Programming
Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
- Pseudo-Boolean
- Finite Domain
- Constraint Programming
- Constraint Integer Programming
SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

What is SCIP?

Solving Constraint Integer Programs

Summary and Outlook
Combination of...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
MIP Solving Techniques

Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ... 

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of...

- LP relaxation
- branch-and-bound
- cutting planes
MIP Solving Techniques

Combination of...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ... 

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- LP relaxation
- branch-and-bound
- cutting planes
Combination of ...

- branch-and-bound
- domain propagation
Combination of …

- branch-and-bound
- domain propagation

Analyze …

- constraints of subproblem
- current domains of variables in order to infer domain reductions. (→ restrict the search space)
CP Solving Techniques

Combination of ...

- branch-and-bound
- domain propagation

Alldifferent constraint:
\[ \{x_1, x_2, x_3, x_4\} \in \{1, 2, 3, 4\} \]
pairwise different.

\[
\begin{align*}
x_1 & \quad x_1 \\
x_2 & \quad x_2 \\
x_3 & \quad \Rightarrow \\
x_4 & \quad x_4
\end{align*}
\]
CIP Solving Techniques

MIP
- LP relaxation
- Cutting planes

CP
- Domain propagation

SAT
- Conflict analysis
- Periodic restarts

MIP, CP, and SAT
- Branch-and-bound

SCIP
SCIP – Solving Constraint Integer Programs

Constraint Integer Programs
Solving Techniques
What is SCIP?
Solving Constraint Integer Programs
Summary and Outlook
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
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
03/2004 Knapsack cover separator (Kati Wolter)
01/2005 Objective feasibility pump (Timo Berthold)
09/2005 First public version 0.80
09/2006 Version 0.90
07/2007 Tobias Achterberg finished his PhD.
09/2007 Version 1.00
01/2008 Solution counting (Stefan Heinz)
09/2008 Version 1.1.0
10/2008 Nonlinear support (Stefan Vigerske)
09/2009 Version 1.2.0
SCIP Developers

- Tobias Achterberg (IBM)
- Marc Pfetsch (TU Braunschweig)
- Thorsten Koch
- Timo Berthold
- Stefan Heinz
- Stefan Vigerske
- Kati Wolter
- Gerald Gamrath
- Robert Waniek
- Michael Winkler
SCIP Contributors

- Andreas Bley (TU Berlin)  
  VRP example
- Daniel Espinoza (U. Chile)  
  Interface to QSopt
- John Forrest (IBM)  
  Interface to CLP
- Thorsten Gellermann (TU Darmstadt)  
  Generic NLP interface
- Ambros Gleixner (ZIB)  
  GAMS Reader, SoPlex support
- Bo Jensen (Mosek)  
  Interface to MOSEK
- Manuel Kutschka (RWTH Aachen)  
  Separator for 0,1/2-cuts
- Michael Perregaard (FICO/Xpress)  
  Interface to FICO/Xpress
- Christian Raack (ZIB)  
  Separator for MCF cuts
- Jörg Rambau (U. Bayreuth)  
  SamplePricer example
- Cornelius Schwarz (U. Bayreuth)  
  Queens example
- Andreas Tuchscherer (ZIB)  
  SamplePricer example
SCIP Contributors

- Andreas Bley (TU Berlin) — VRP example
- Daniel Espinoza (U. Chile) — Interface to QSopt
- John Forrest (IBM) — Interface to CLP
- Thorsten Gellermann (TU Darmstadt) — Generic NLP interface
- Ambros Gleixner (ZIB) — GAMS Reader, SoPlex support
- Bo Jensen (Mosek) — Interface to MOSEK
- Manuel Kutschka (RWTH Aachen) — Separator for 0,1/2-cuts
- Michael Perregaard (FICO/Xpress) — Interface to FICO/Xpress
- Christian Raack (ZIB) — Separator for MCF cuts
- Jörg Rambau (U. Bayreuth) — SamplePricer example
- Cornelius Schwarz (U. Bayreuth) — Queens example
- Andreas Tuchscherer (ZIB) — SamplePricer example

YOU?
Performance of SCIP

One of the fastest noncommercial MIP solver

Performance development

results taken from H. Mittelmann (09/17/2009)
Performance of SCIP

▷ One of the fastest noncommercial MIP solver

![Graph comparing SCIP with other solvers]

- **Non-commercial solvers:**
  - BLIS 0.91: 16.5x
  - GLPK 4.39: 15.9x
  - lpsolve 5.5: 15.2x
  - Symphony 5.2.0: 4.48x
  - CBC 2.3.1: 1.80x
  - SCIP 1.2 – SoPlex 1.4.2: 1.43x
  - SCIP 1.2 – CLP 1.10.1: 1.00x
  - Minto 3.1 – Cplex 9: 0.62x
  - SCIP 1.2 – Cplex 12.1: 0.23x
  - Gurobi 1.1.2: 0.21x
  - Cplex 12.1: 0.00x

- **Commercial solvers:**
  -的结果 from H. Mittelmann (09/17/2009)

▷ Performance development

![Graph comparing SCIP versions]

- SCIP 0.7: 7.48x
- SCIP 0.80: 3.07x
- SCIP 0.90: 2.17x
- SCIP 0.91: 2.38x
- SCIP 1.00: 1.00x
- SCIP 1.1: 0.62x

- Results taken from H. Mittelmann (09/17/2009)

Test set: Miplib2003
LP solver: SoPlex
Performance of SCIP

▷ One of the fastest noncommercial MIP solver

![Graph showing performance comparison between SCIP 1.2 and other solvers.]

- SCIP 1.2 – SoPlex 1.4.2: 16.5x
- SCIP 1.2 – CLP 1.10.1: 15.9x
- SCIP 0.7: 16.2x
- GLPK 4.39: 15.2x
- lpsolve 5.5: 15.0x
- Symphony 5.2.0: 15.0x
- CBC 2.3.1: 4.48x
- BLIS 0.91: 1.80x
- Minto 3.1 – Cplex 9: 1.43x
- SCIP 1.2 – Cplex 12.1: 1.00x
- Gurobi 1.1.2: 0.62x
- Cplex 12.1: 0.23x
- SoPlex: 0.21x

- Not solved: 77% for non-commercial, 77% for commercial.
- Time in seconds: 0, 500, 1000, 1500.

Results taken from H. Mittelmann (09/17/2009)

▷ Performance development

![Graph showing performance development over time.]

- SCIP 0.7: 7.48x
- SCIP 0.80: 3.07x
- SCIP 0.90: 2.17x
- SCIP 1.00: 2.38x
- SCIP 1.1: 1.00x


Winner of the “Pseudo-Boolean Evaluation 2009”

http://www.cril.univ-artois.fr/PB09/
Approx. 275.640 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

7 interfaces to external linear programming solvers
   → CLP, CPLEX, Gurobi, Mosek, QSopt, SoPlex, XPRESS

6 different input formats
   → cnf, flatzinc, lp, mps, opb (pseudo-boolean), zimpl

876 parameters to play with

Part of the ZIB Optimization Suite http://zibopt.zib.de
ZIP Optimization Suite = SCIP + SoPlex + ZIMPL

Tool for generating and solving constraint integer programs

**ZIMPL**
- A mixed integer programming modeling language
- Easily generating linear programs and mixed integer programs

**SCIP**
- A mixed integer programming solver and constraint programming
- ZIMPL models can directly be loaded into SCIP and solved

**SoPlex**
- A linear programming (LP) solver
- Solution process SCIP may use SoPlex as underlying LP solver

All three tools are available in source code and free for academic use
Outline

SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

What is SCIP?

Solving Constraint Integer Programs

Summary and Outlook
Important Aspects

- SCIP is constraint based
  - Advantage: flexibility
  - Disadvantage: limited global view

- A constraint knows its variables, but a variable does not know the constraints it appears in.

- A single constraint may represent a whole set of inequalities, not only a single one.

- From the constraint programming perspective:
  - LP-relaxation is only an add-on!
  - Many constraints do not separate inequalities at all.
Operational Stages

- Init
- Problem
- Transforming
- Presolving
- Solving
- Free Transform
- Free Solve

Basic data structures are allocated and initialized.

User includes required plugins (or just takes default plugins).
Basic data structures are allocated and initialized.

User includes required plugins (or just takes default plugins).
Main SCIP interface: **plugins**.

- Perform all problem specific actions.
- Each step calls user defined plugins.
- SCIP knows of plugins through “include” functions.
- Plugins may have private data.
- User defined callback functions (virtual functions in C++-interface).
- Yields modular structure.
- The MIP solver is realized through plugins.

Everything SCIP knows, it knows through plugins.
Types of Plugins

- **Constraint handler**: assures feasibility, strengthens formulation
- **Separator**: adds cuts, improves dual bound
- **Pricer**: allows dynamic generation of variables
- **Heuristic**: searches solutions, improves primal bound
- **Branching rule**: how to divide the problem?
- **Node selection**: which subproblem should be regarded next?
- **Presolver**: simplifies the problem in advance, strengthens structure
- **Propagator**: simplifies problem, improves dual bound locally
- **Reader**: reads problems from different formats
- **Event handler**: catches events (e.g., bound changes, new solutions)
- **Display**: allows modification of output
- ...
User creates and modifies the original problem instance.
Problem creation is usually done in file readers (SCIPreadProb()).
SCIP Structures

SCIP

Reader

Separator

Presolver

Implications

Varible

Event

Node selector

Display

Dialog

Cutpool

Conflict

Branch

Heuristic

Boundary

Thread

Cutpool

Presolver

Implications

Varible

Event

Node selector

Display

Dialog

Cutpool

Conflict

Branch

Heuristic

Boundary

Thread

SCIP

Stefan Heinz (ZIB)
▶ Creates a working copy of the original problem.
- data are copied into separate memory area
- presolving and solving operate on transformed problem
- original data can only be modified in problem modification stage
Presolvers provide global presolving (e.g. dual fixing)

Constraint Handlers provide constraint specific presolving, e.g.:
- Domain tightening,
- Coefficient modification,
- Deletion of redundant constraints,
- Constraint upgrading.
- ...
Solving

Init

Problem -> Transforming -> Presolving

Free Transform

Init Solve

Solving

Free Solve
Flow Chart SCIP

Presolving → Stop → Node selection → Processing → Branching

Conflict analysis → Primal heuristics

Solve LP → Domain propagation → Pricing → Cuts → Enforce constraints
Flow Chart SCIP

Flow Chart Details:

- **Presolving**
- **Stop**
- **Node selection**
- **Conflict analysis**
- **Processing**
- **Branching**
- **Primal heuristics**
- **Solve LP**
- **Domain propagation**
- **Pricing**
- **Cuts**
- **Enforce constraints**

Diagram Description:

- Presolving leads to Stop.
- Stop is connected to Node selection.
- Node selection connects to Conflict analysis and Processing.
- Conflict analysis leads to Branching.
- Branching connects to Primal heuristics.
- Processing leads to Solve LP and Enforce constraints.
- Solve LP is connected to Cuts and Pricing.
- Cuts and Pricing connect to Enforce constraints.
- Enforce constraints leads to Solve LP.
Node Selection

- Node Selectors called in order of priority
- Choose:
  - Child node → subproblem’s processing is less expensive
  - Sibling node → subproblem’s processing is less expensive
  - Other open leaf node
- Two operation modes:
  - Standard mode (default: best estimate)
  - Memory-saving mode (default: depth first search)

Switching between modes is automatic

```
SCIP> set limits memory 100
SCIP> set memory savefrac 0.8
```
Flow Chart SCIP

Presolving

Stop

Node selection

Conflict analysis

Processing

Primal heuristics

Branching

Solve LP

Pricing

Cuts

Domain propagation

Enforce constraints
Domain Propagation (Node Presolving)

- Propagatore solvers provide local presolving
- Constraint Handlers provide constraint specific propagation, e.g.:
  - Domain tightening,
  - Coefficient modification,
  - Deletion of redundant constraints,
  - Constraint upgrading.
  - ...

Stefan Heinz (ZIB)
Solving CIPs
09/22/2009 44 / 71
Solve Linear Program (LP)

- LP Interface to various LP solvers:
  → CLP, CPLEX, Gurobi, Mosek, QSopt, SoPlex, XPRESS
- Primal or dual simplex
- Barrier with or without crossover (if supported)
- LP stability checked:
  resolution by changing parameters:
  scaling, tolerances, solving from scratch, other simplex method.
SCIP Structures
Constraint Enforcement

LP solution may violate a constraint not contained in the relaxation.

Enforcing is necessary for a correct implementation!

**Constraint handler** resolves the infeasibility by ...

- Reducing a variable’s domain,
- Separating a cutting plane (may use integrality),
- Adding a (local) constraint,
- Creating a branching,
- Concluding that the subproblem is infeasible and can be cut off, or
- Just saying “solution infeasible”.

Stefan Heinz (ZIB)
Constraint Enforcement

Presolving

Stop

Node selection

Processing

Branching

Conflict analysis

Primal heuristics

Domain propagation

Solve LP

Pricing

Cuts

Enforce constraints

▷ Reduced domain  ▷ Added cut  ▷ Cutoff  ▷ Infeasible
▷ Added constraint  ▷ Branched  ▷ Feasible

Stefan Heinz (ZIB)
Solving CIPs
09/22/2009 52 / 71
Constraint Enforcement

- Presolving
- Stop
- Node selection
  - Conflict analysis
  - Primal heuristics
  - Branching
    - Reduced domain
    - Added constraint
  - Processing
    - Added cut
    - Branched
- Solve LP
  - Pricing
  - Cuts
  - Enforce constraints
    - Cutoff
    - Infeasible
    - Feasible
Constraint Enforcement

- Presolving
- Node selection
- Processing
- Branching
- Conflict analysis
- Primal heuristics
- Solve LP
- Domain propagation
- Pricing
- Cuts
- Enforce constraints
- Infeasible
- Feasible
- Cutoff
- Branched
- Added cut
- Reduced domain
- Added constraint
Constraint Enforcement

- Presolving
- Stop
- Node selection
- Conflict analysis
- Processing
- Branching
- Primal heuristics

- Solve LP
- Domain propagation
- Pricing
- Cuts
- Enforce constraints

△ Reduced domain △ Added cut △ Cutoff △ Infeasible
△ Added constraint △ Branched △ Feasible
Constraint Enforcement

- Presolving
- Stop
- Node selection
- Conflict analysis
- Processing
- Primal heuristics
- Branching
  - Reduced domain
  - Added constraint
  - Added cut
  - Branched
  - Cutoff
  - Infeasible
  - Feasible
- Solve LP
  - Domain propagation
  - Pricing
  - Cuts
- Enforce constraints
Constraint Enforcement

- Presolving
- Node selection
- Processing
- Branching
- Conflict analysis
- Primal heuristics
- Enforce constraints
- Solve LP
  - Pricing
  - Cuts
- Domain propagation
- Stop

- Reduced domain
- Added cut
- Cutoff
- Infeasible
- Added constraint
- Branched
- Feasible
Constraint Enforcement

- **Presolving**
- **Stop**
- **Node selection**
- **Processing**
- **Branching**
- **Domain propagation**
- **Solve LP**
- **Pricing**
- **Cuts**
- **Enforce constraints**
  - Reduced domain
  - Added constraint
  - Added cut
  - Branched
  - Cutoff
  - Infeasible
  - Feasible
Flow Chart SCIP

Presolving

Stop

Node selection

Conflict analysis

Processing

Primal heuristics

Branching

Solve LP

Domain propagation

Pricing

Cuts

Enforce constraints
Flow Chart SCIP

Presolving

Stop

Node selection

Conflict analysis

Processing

Branching

Primal heuristics

Solve LP

Domain propagation

Pricing

Cuts

Enforce constraints

Pricing: allows dynamic generation of variables
Flow Chart SCIP

- Presolving
- Node selection
- Processing
- Branching
- Conflict analysis
- Primal heuristics
- Solve LP
- Pricing
- Cuts
- Domain propagation
- Enforce constraints
- Stop

▶ Creates a branching with the infeasible solution no longer being feasible in the relaxations of the child nodes.
SCIP Structures

Variable

... 

SCIP Heuristic

actcons
diving
coef
diving
cross
over
dins
feaspump
fixand
infer
fracdiving
guided
diving

simple
rounding
shifting
rootsol
diving

rounding
rens

rins

SCIP

Variable

. . .

SCIP Heuristic

actcons
diving
coef
diving
cross
over
dins
feaspump
fixand
infer
fracdiving
guided
diving

simple
rounding
shifting
rootsol
diving

rounding
rens

rins

SCIP

Variable

. . .

SCIP Heuristic

actcons
diving
coef
diving
cross
over
dins
feaspump
fixand
infer
fracdiving
guided
diving

simple
rounding
shifting
rootsol
diving

rounding
rens

rins

Stefan Heinz (ZIB)
Solving CIPs
Primal heuristics try to find feasible solutions (in addition to feasible LP solutions).
Conflict analysis learns from infeasible subproblems
SCIP – Solving Constraint Integer Programs

Constraint Integer Programs

Solving Techniques

What is SCIP?

Solving Constraint Integer Programs

Summary and Outlook
Constraint Integer Programming

- Mixed Integer Programs
- Satisfiability
- Pseudo-Boolean
- Finite Domain
- Constraint Programming
- Constraint Integer Programming
Advantages and Disadvantages of SCIP

Advantages

▷ Can be used as a black box MIP solver
Advantages and Disadvantages of SCIP

**Advantages**
- Can be used as a **black box** MIP solver
- **Hundreds of parameters** to play with (876)

**Disadvantages**
- **Hundreds of parameters** to play with (876)
- In general variables do not know in which constraint they appear since constraint based...
Advantages and Disadvantages of SCIP

Advantages

- Can be used as a black box MIP solver
- Hundreds of parameters to play with (876)
- Robust, fast, and well documented
Advantages and Disadvantages of SCIP

Advantages

- Can be used as a black box MIP solver
- Hundreds of parameters to play with (876)
- Robust, fast, and well documented
- Extendable in any direction since constraint based
  - Adding global constraints
  - Adding problem specific plugins
Advantages and Disadvantages of SCIP

Advantages

- Can be used as a black box MIP solver
- Hundreds of parameters to play with (876)
- Robust, fast, and well documented
- Extendable in any direction since constraint based
  - Adding global constraints
  - Adding problem specific plugins

Disadvantages

- Hundreds of parameters to play with (876)
- In general variables do not know in which constraint they appear since constraint based
### Applications

Other universities and institutes using SCIP:

- Universität Bayreuth
- FU Berlin
- TU Berlin
- TU Braunschweig
- TU Chemnitz
- TU Darmstadt
- TU Dresden
- Fachhochschule Lausitz
- OvGU Magdeburg
- TU München
- Universität Osnabrück
- Universität Stuttgart
- Università dell’Aquila
- University College Cork
- Danmarks TU
- Rijksuniversiteit Groningen
- Københavns Universitet
- University of Melbourne
- Monash University
- University of Newcastle
- University of Nottingham
- IASI CNR
- Universidad SF de Quito
- Universidade de Sao Paulo
Documentation

- Constraint Integer Programming
  PhD. dissertation of Tobias Achterberg, 2007
- Primal Heuristics for Mixed Integer Programs
- Implementation of Cutting Plane Separators for MIPs
- CIP: a New Approach to Integrate CP and MIP
  Achterberg, Berthold, Koch, and Wolter, Proceedings CPAIOR08
- CIP: Techniques and Applications
  Achterberg, Berthold, Heinz, Koch, and Wolter, submitted

- http://scip.zib.de Doxygen documentation, HowTos, FAQ, Additional Examples
Automatically run test

SCIP comes with a set of scripts which allow automatic testing:
```
scip> make test
```
- TEST state test set
- SETTINGS state setting file
- TIME state a time limit
- NODE state a node limit
- MEM state a memory limit
- ...

```
scip> make test TEST=miplib MEM=1024 TIME=3600
```
- Result file summarizing the test run
- Comparison of different runs

See Doxygen documentation for more information
Visualize the Branch-and-Cut Tree

SCIP can visualize the branch-and-bound tree
SCIP> set vbc filename stein45.tree

▶ SCIP creates a file named “stein45.tree”
▶ File is readable with vbctool
  ▶ vbc – visualization of branching and cut Algorithms
  ▶ Drawing a tree during computational process
  ▶ Emulating a tree growing process after the computation is finished
  ▶ http://www.informatik.uni-koeln.de/old-ls_juenger/projects/vbctool.html
  ▶ Just Google for “vbctool”
  ▶ Group of Prof. Dr. Jünger in Cologne (Germany)
Future Plans and Research

Future plans:
- More cuts (e.g., odd hole), more heuristics (combinatorial)
- Infeasibility analysis
- Improve column generation
- Scheduling capabilities
- CP constraints (alldifferent, element)
- extend nonlinear support

Research:
- Exact integer programming (Project within DFG program “Algorithm Engineering”)
- Mixed integer nonlinear programming (MATHEON)
Conclusion

Why should you use SCIP?

- You need a full-scale MIP solver
- You want to combine SAT, MIP, and CP features
- You need a solver which is free for academic uses
- You need a solver which is competitive to state-of-the-art solvers
- You want to have total control over the solving process
- You need a solver which is easy to use and well documented

You can contribute by . . .

- Using SCIP and report publications/projects,
- Sending bug-reports – better: bug-fixes,
- Giving feedback on documentation,
- Implementing plugins.
SCIP at CO@Work

▷ Tuseday (today):
  ▷ Get ZIB Optimization Suite running
  ▷ Solve a ZIMPL model with SCIP
  ▷ Create a SCIP project

▷ Saturday (09/26/09):
  ▷ Talks: Node selections, branching rules, and constraint handlers
  ▷ Implementing a constraint handler (TSP)

▷ Wendsday (09/30/09)
  ▷ Talks: Cuts and heuristics
  ▷ Implementing a heuristics (TSP)
  ▷ Implementing a pricer (Binpacking)
SCIP Structures

SCIP

Heuristic nearest neighbor
opt2
cross
over
dins
feaspump
fixand
infer
fracdiving
guided
diving
intdiving
linesearch
diving
local
branching
mutation
objpscost
diving
octane
oneopt
pscost
diving
rens
rins
rootsol
diving
rounding
shifting
simple rounding
default
Branch

Conflict

Constraint

Handler

Subtour

bound
disjunc.
count
sols
indicator
integral
knap
sack
linear
logicor
or
setppc
sos1
sos2
var
bound
default

Cutpool

LP

Presolver

Implcations

Tree

Reader

Branch

Propagator

Relaxer

Variable

...