DM87
SCHEDULING,
TIMETABLING AND ROUTING

Lecture 5

Mathematical Programming, Exercises

Marco Chiarandini

Outline

Outline

How to solve mathematical programs

- Use a mathematical workbench like MATLAB, MATHEMATICA, MAPLE, R.
- Use a modeling language to convert the theoretical model to a computer usable representation and employ an out-of-the-box general solver to find solutions.
- Use a framework that already has many general algorithms available and only implement problem specific parts, e. g., separators or upper bounding.
- Develop everything yourself, maybe making use of libraries that provide high-performance implementations of specific algorithms.

Thorsten Koch
"Rapid Mathematical Programming"
Technische Universität, Berlin, Dissertation, 2004

How to solve mathematical programs

- Use a mathematical workbench like MATLAB, MATHEMATICA, MAPLE, R.

Advantages: easy if familiar with the workbench

Disadvantages: restricted, not state-of-the-art

How to solve mathematical programs

- Use a modeling language to convert the theoretical model to a computer usable representation and employ an out-of-the-box general solver to find solutions.

Advantages: flexible on modeling side, easy to use, immediate results, easy to test different models, possible to switch between different state-of-the-art solvers

Disadvantages: algoritmical restrictions in the solution process, no upper bounding possible

How to solve mathematical programs

- Use a framework that already has many general algorithms available and only implement problem specific parts, e.g., separators or upper bounding.

Advantages: allow to implement sophisticated solvers, high performance bricks are available, flexible

Disadvantages: view imposed by designers, vendor specific hence no transferability,

How to solve mathematical programs

- Develop everything yourself, maybe making use of libraries that provide high-performance implementations of specific algorithms.

Advantages: specific implementations and max flexibility

Disadvantages: for extremely large problems, bounding procedures are more crucial than branching

Modeling Languages

Name		URL	Solver	State
AIMMS	Advanced Integrated Multi-dimensional Modeling Software	www.aimms.com	open	commercial
AMPL	A Modeling Language for Mathematical Programming	www.ampl.com	open	commercial
GAMS	General Algebraic Modeling System	www.gams.com	open	commercial
LINGO	Lingo	www.lindo.com	fixed	commercial
LPL	(Linear|Logical|Literate) Programming Language	www.virtual-optima.com	open	commercial
MINOPT	Mixed Integer Non-linear Optimizer	titan.princeton.edu/MINOPT	open	mixed
MOSEL	Mosel	www.dashoptimization.com	fixed	commercial
MPL	Mathematical Programming Language	www.maximalsoftware.com	open	commercial
OMNI	Omni	www.haverly.com	open	commercial
OPL	Optimization Programming Language	www.ilog.com	fixed	commercial
GNU-MP	GNU Mathematical Programming Language	www.gnu.org/software/glpk	fixed	free
ZIMPL	Zuse Institute Mathematical Programming Language	www.zib.de/koch/zimpl	open	free

Thorsten Koch
"Rapid Mathematical Programming"
Technische Universität, Berlin, Dissertation, 2004

LP-Solvers

CPLEX	`http://www.ilog.com/products/cplex`
XPRESS-MP	`http://www.dashoptimization.com`
SOPLEX	`http://www.zib.de/Optimization/Software/Soplex`
COIN CLP	`http://www.coin-or.org`
GLPK	`http://www.gnu.org/software/glpk`
LP_SOLVE	`http://lpsolve.sourceforge.net/`

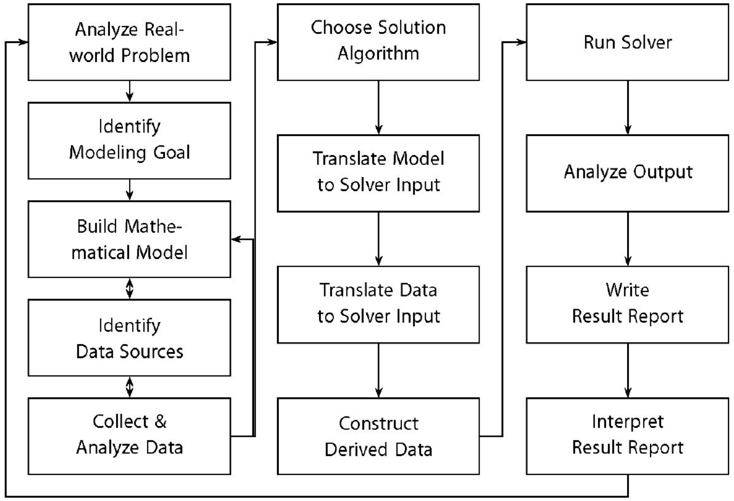
"Software Survey: Linear Programming" by Robert Fourer
`http://www.lionhrtpub.com/orms/orms-6-05/frsurvey.html`

Outline

ZIBOpt

- Zimpl is a little algebraic Modeling language to translate the mathematical model of a problem into a linear or (mixed-) integer mathematical program expressed in .lp or .mps file format which can be read and (hopefully) solved by a LP or MIP solver.
- Scip is an IP-Solver. It solves Integer Programs and Constraint Programs: the problem is successively divided into smaller subproblems (branching) that are solved recursively. Integer Programming uses LP relaxations and cutting planes to provide strong dual bounds, while Constraint Programming can handle arbitrary (non-linear) constraints and uses propagation to tighten domains of variables.
- SoPlex is an LP-Solver. It implements the revised simplex algorithm. It features primal and dual solving routines for linear programs and is implemented as a C++ class library that can be used with other programs (like SCIP). It can solve standalone linear programs given in MPS or LP-Format.

Modeling Cycle



H. Schichl. "Models and the history of modeling".
In Kallrath, ed., Modeling Languages in Mathematical Optimization, Kluwer, 2004.

Some commands

```
$ zimpl -t lp sudoku.zpl
$ scip -f sudoku.lp
```

```
scip> help
scip> read sudoku.lp
scip> display display
scip> display problem
scip> set display width 120
scip> display statistics
scip> display parameters
scip> set default
scip> set load settings/*/*.set
scip> set load /home/marco/ZIBopt/ziboptsuite-1.00/scip-1.00/settings/
    emphasis/cpsolver.set
```


Callable libraries
How to construct a problem instance in SCIP

```
SCIPcreate(), // create a SCIP object
SCIPcreateProb() // build the problem
SCIPcreateVar() // create variables
SCIPAddVar() // add them to the problem
// Constraints: For example, if you want to
// fill in the rows of a general MIP, you have to call
SCIPcreateConsLinear(),
SCIPAddConsLinear()
SCIPreleaseCons() // after finishing.
SCIPsolve()
SCIPreleaseVar() releas variable poiinters
```

```
SCIP_CALL() // exception handling
```

```
SCIPsetIntParam(scip, "display/memused/status", 0) == set display \
memused status 0
SCIPprintStatistics() == display statistics
```


Sudoku into Exact Hitting Set

Exact Covering: Set partitioning with $\vec{c} = \vec{1}$

- A = 1, 4, 7;
- B = 1, 4;
- C = 4, 5, 7;
- D = 3, 5, 6;
- E = 2, 3, 6, 7; and
- F = 2, 7.

$$\min \sum_{j=1}^{n} y_j \qquad \sum_{j=1}^{n} a_{ij} y_j = 1 \quad \forall i \qquad y_j \in \{0,1\}$$

	A	B	C	D	E	F
1	1	1	0	0	0	0
2	0	0	0	0	1	1
3	0	0	0	1	1	0
4	1	1	1	0	0	0
5	0	0	1	1	0	0
6	0	0	0	1	1	0
7	1	0	1	0	1	1

The dual of Exact Covering is the Exact Hitting Set

- A = 1, 2
- B = 5, 6
- C = 4, 5
- D = 1, 2, 3
- E = 3, 4
- F = 4, 5
- G = 1, 3, 5, 6

$$\max \sum_{j=1}^{n} x_j \qquad \sum_{j=1}^{n} a_{ij} x_j = 1 \quad \forall i \qquad x_j \in \{0,1\}$$

	A	B	C	D	E	F	G
1	1	0	0	1	0	0	1
2	1	0	0	1	0	0	0
3	0	0	0	1	1	0	1
4	0	0	1	0	1	1	0
5	0	1	1	0	0	1	1
6	0	1	0	0	0	0	1