SCIP Optimization Suite
–
Installation and Testing

Matthias Miltenberger

Zuse Institute Berlin

30th September 2014

Contents of the USB stick

- tarball of the SCIP Optimization Suite 3.1
- precompiled binaries for Windows (32bit and 64bit)
- installer for Cygwin
  - Cygwin emulates a Linux system on Windows
- additional test problems for the afternoon session

- Please copy everything to your machine!

Installing Cygwin

- enter directory `cygwin`
- run `setup-x86.exe`
- choose local installation
- point installer to `cygwin` directory
- select all available packages to be installed
- wait a few minutes...

Installing Cygwin

- enter directory `cygwin`
- run `setup-x86.exe`
- choose local installation
- point installer to `cygwin` directory
- select all available packages to be installed
- wait a few minutes...

- run Cygwin shell from the desktop
- copy SCIP package into Cygwin file system to have access to it

Installing SCIP

basic requirements:

- LP solver
- C/C++ compiler

- everything you need for a basic installation is contained in the SCIP Optimization Suite
- extract the package and type make:

```
tar xvf scipoptsuite-3.1.0.tgz
make
```

Installing SCIP

basic requirements:

- LP solver
- C/C++ compiler

- everything you need for a basic installation is contained in the SCIP Optimization Suite
- extract the package and type make:

```
tar xvf scipoptsuite-3.1.0.tgz
make
```

- will most likely not work on a standard Linux installation...
- additional packages/third-party software is required to fully enjoy SCIP
- carefully read the error messages to see what is missing

Third party software

- ZLIB
  - necessary to load compressed problem files
- Readline
  - comfortable interactive shell (tab-completion, history of commands, ...)
- GMP
  - Gnu Multiple Precision library
  - allows for higher precision arithmetic
- ncurses
  - necessary for output formatting
- ...

- You need the respective developer versions, indicated by `-devel` or `-dev` in your favorite package manager!

Alternative

What if you cannot install a package?

- most features can be disabled to solve installation problems:
- `make ZLIB=false`
- `make READLINE=false`
- `make GMP=false`

- ZIMPL has additional requirements: *bison*, *flex*
- in case you cannot install ZIMPL:
  `make ZIMPL=false`

... even more options

- SCIP supports several different LP solvers:
  - IBM CPLEX: `make LPS=cpx`
  - FICO Xpress: `make LPS=xprs`
  - Gurobi: `make LPS=grb`
  - CoinOR CLP: `make LPS=clp`
  - Mosek: `make LPS=msk`
  - QSopt: `make LPS=qso`
  - improved SoPlex interface: `make LPS=spx2`

- for MINLP we recommend you install and use Ipopt
  - `make IPOPT=true`
  - a working installation of Ipopt is required (not covered here)

- give your binary a descriptive name:
  - `make VERSION=awesome-new-feature`
  - defaults to the current version number

Debugging with SCIP

SCIP can be compiled and run in debug mode

- asserts will be checked
- additional checks will be performed
- more warnings may be printed
- debug information is available for gdb
- obviously slower than the optimized mode
- useful when writing new code and on the hunt for bugs

- `make OPT=dbg`

Examples contained in SCIP

- show how SCIP can be used
- available examples:
  - Binpacking
  - CallableLibrary
  - Coloring
  - Eventhdlr
  - Queens
  - Scheduler
  - TSP
  - ...

- go to `examples/`
- every example has its own Makefile (simply type `make`)
- examples usually rely on compiled main SCIP code
- provide starting points for further developments

Beyond SCIP...

- GCG, a generic column generation solver, can be compiled with `make gcg`
  - relies on *bliss* – for graph computations
- UG, the parallelization framework, can be compiled with `make ug`
  - default is to build FiberSCIP, a shared memory parallel SCIP extension

Testing Environment

- SCIP package provides many scripts to make testing easy
- general setup:
  - set of instances put together in a test set:
    `check/testset/miplib.test`
  - set of solution values to check found solution:
    `check/testset/miplib.solu` (optional)
  - run SCIP on the entire test set:
    `make test TEST=miplib`
  - test non-default parameters:
    `make test TEST=miplib SETTINGS=heuristics-off`
  - use the interactive shell to create a new settings file
  - settings need to be placed in `settings/`

Default values

```
    TEST = short
     OPT = opt
SETTINGS = default
 VERSION = 3.1.0
     LPS = spx
     GMP = true
   ZIMPL = true
READLINE = true
    ZLIB = true
   IPOPT = false
    TIME = 3600 (sec)
     MEM = 6144 (MB)
```

Output of a single test run

- `.out` file:
  concatenation of all logfiles
- `.err` file:
  concatenation of all occured errors/warnings
- `.set` file:
  copy of the used settings
- `.res` file:
  concise summary of test run
- `.tex` file:
  LaTeX table of results
- `.pav` file:
  machine readable table of results

- all files are stored in `check/results/`:

`check.miplib.scip-3.1.0.linux.x86_64.gnu.dbg.spx.opt49.default.out`

Compare multiple runs

- multiple runs over the same test set can be compared
- use `check/allcmpres.sh`

- go to check/
- run `allcmpres.sh results/check.1.res results/check.2.res`
- more than 2 result files can be compared
- usual usecase:
  - compare different SCIP settings
  - compare different SCIP versions

Additional info about SoPlex

- SoPlex provides very similar setup
- many Makefile options are the same as in SCIP
- test run also produces a `JSON` file that contains all data and is easily parseable
- comparison evaluations are done using Python scripts