SCIP

Stefan Vigerske, Humboldt University Berlin, Germany

Contents

1 Introduction

SCIP (**S**olving **C**onstraint **I**nteger **P**rograms) is developed at the Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB). The SCIP main developer had been Tobias Achterberg, current developers are Timo Berthold, Gerald Gamrath, Stefan Heinz, Gregor Hendel, Thorsten Koch, Stefan Vigerske, Robert Waniek, Michael Winkler, and Kati Wolter. Since SCIP is distributed under the ZIB Academic License, it is only available for users with a **GAMS academic license**.

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. SCIP can also be used as a pure MIP solver or as a framework for branch-cut-and-price. Within GAMS, the MIP and MIQCP solving facilities of SCIP are available.

For more detailed information, we refer to [1, 2, 3, 4, 5, 7] and the SCIP web site `http://scip.zib.de`, especially the list of papers listed at `http://scip.zib.de/related.shtml`.

GAMS/SCIP uses the linear solver SoPlex [8] as LP solver and the COIN-OR Interior Point Optimizer Ipopt [6] as nonlinear solver.

2 Model requirements

SCIP supports continuous, binary, integer, semi-continuous, and semi-integer variables, special ordered sets, and branching priorities. Indicator constraints are not supported by the interface yet.

3 Usage

The following statement can be used inside your GAMS program to specify using SCIP

```
Option MIP = SCIP;      { or QCP or RMIQCP or MIQCP }
```

The above statement should appear before the Solve statement. If SCIP was specified as the default solver during GAMS installation, the above statement is not necessary.

If a continuous linear program (LP or RMIP) is given to GAMS/SCIP, the linear programming solver is called directly. In this case, it is not possible to provide an option file.

GAMS/SCIP currently does not support the GAMS Branch-and-Cut-and-Heuristic (BCH) Facility. If you need to use GAMS/SCIP with BCH, please consider to use a GAMS system of version $\leq$ 23.3, available at http://www.gams.com/download/download_old.htm.

Specification of SCIP Options

GAMS/SCIP currently supports the GAMS parameters `reslim`, `iterlim`, `nodlim`, `optcr`, and `optca`. Further, under Linux and Windows, the option `threads` can be used to control the number of threads used in the linear algebra routines of IPOPT.

Options can be specified by a SCIP options file. A SCIP options file consists of one option or comment per line. A pound sign (#) at the beginning of a line causes the entire line to be ignored. Otherwise, the line will be interpreted as an option name and value separated by an equal sign (=) and any amount of white space (blanks or tabs). Further, string values have to be enclosed in quotation marks.

A small example for a scip.opt file is:

```
presolving/probing/maxrounds = 0
separating/maxrounds         = 0
separating/maxroundsroot     = 0
```

It causes GAMS/SCIP to disable probing during presolve and to turn off all cut generators.

4 Detailed Options Description

SCIP supports a large set of options. Sample option files can be obtained from http://www.gams.com/~svigerske/scip2.0.

In the following we give a detailed list of some SCIP options. A list of all SCIP options can be obtained from http://scip.zib.de/doc/html/PARAMETERS.html.

GAMS interface specific options

`gams/names` (`boolean`) `FALSE`
This option causes GAMS names for the variables and equations to be loaded into SCIP. These names will then be used for error messages, log entries, and so forth. Turning names off may help if memory is very tight.

`gams/mipstart` (`boolean`) `TRUE`
This option controls the use of advanced starting values for mixed integer programs. A setting of TRUE indicates that the variable level values should be checked to see if they provide an integer feasible solution before starting optimization.

`gams/print_statistics` (`boolean`) `FALSE`
This option controls the printing of solve statistics after a MIP solve. Turning on this option indicates that statistics like the number of generated cuts of each type or the calls of heuristics are printed after the MIP solve.

`gams/interactive` (`boolean`) `FALSE`
whether a SCIP shell should be opened instead of issuing a solve command (this option is not available in demo mode)

Branching

`branching/preferbinary` (`boolean`) `FALSE`
should branching on binary variables be preferred?

`branching/clamp` ($0 \leq$ real ≤ 0.5) 0.2
minimal relative distance of branching point to bounds when branching on a continuous variable

`branching/allfullstrong/priority` ($-536870912 \leq$ integer ≤ 536870911) -1000
priority of branching rule <allfullstrong>

`branching/allfullstrong/maxdepth` ($-1 \leq$ integer) -1
maximal depth level, up to which branching rule <allfullstrong> should be used (-1 for no limit)

`branching/allfullstrong/maxbounddist` ($0 \leq$ real ≤ 1) 1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying branching rule (0.0: only on current best node, 1.0: on all nodes)

`branching/fullstrong/priority` ($-536870912 \leq$ integer ≤ 536870911) 0
priority of branching rule <fullstrong>

`branching/fullstrong/maxdepth` ($-1 \leq$ integer) -1
maximal depth level, up to which branching rule <fullstrong> should be used (-1 for no limit)

`branching/fullstrong/maxbounddist` ($0 \leq$ real ≤ 1) 1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying branching rule (0.0: only on current best node, 1.0: on all nodes)

`branching/inference/priority` ($-536870912 \leq$ integer ≤ 536870911) 1000
priority of branching rule <inference>

`branching/inference/maxdepth` ($-1 \leq$ integer) -1
maximal depth level, up to which branching rule <inference> should be used (-1 for no limit)

`branching/inference/maxbounddist` ($0 \leq$ real ≤ 1) 1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying branching rule (0.0: only on current best node, 1.0: on all nodes)

`branching/inference/useweightedsum` (`boolean`) `TRUE`
should a weighted sum of inference, conflict and cutoff weigths be used?

`branching/mostinf/priority` ($-536870912 \leq$ integer ≤ 536870911) 100
priority of branching rule <mostinf>

`branching/mostinf/maxdepth` ($-1 \leq$ integer) -1
maximal depth level, up to which branching rule <mostinf> should be used (-1 for no limit)

`branching/mostinf/maxbounddist` ($0 \leq$ real ≤ 1) 1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying branching rule (0.0: only on current best node, 1.0: on all nodes)

`branching/leastinf/priority` ($-536870912 \leq$ integer ≤ 536870911) 50
priority of branching rule <leastinf>

`branching/leastinf/maxdepth` ($-1 \leq$ integer) -1
maximal depth level, up to which branching rule <leastinf> should be used (-1 for no limit)

`branching/leastinf/maxbounddist` ($0 \leq$ real ≤ 1) 1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying branching rule (0.0: only on current best node, 1.0: on all nodes)

`branching/pscost/priority` ($-536870912 \leq$ integer ≤ 536870911) 2000
priority of branching rule <pscost>

`branching/pscost/maxdepth` ($-1 \leq$ integer) -1
maximal depth level, up to which branching rule <pscost> should be used (-1 for no limit)

`branching/pscost/maxbounddist` ($0 \leq$ real ≤ 1) 1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying branching rule (0.0: only on current best node, 1.0: on all nodes)

`branching/pscost/strategy` (`character`) `r`
strategy for computing score of external branching candidates (b: rb-int-br, r: rb-int-br-rev, i: rb-inf)

`branching/random/priority` ($-536870912 \leq$ integer ≤ 536870911) -100000
priority of branching rule <random>

`branching/random/maxdepth` ($-1 \leq$ integer) -1
maximal depth level, up to which branching rule <random> should be used (-1 for no limit)

`branching/random/maxbounddist` ($0 \leq$ real ≤ 1) 1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying branching rule (0.0: only on current best node, 1.0: on all nodes)

`branching/random/seed` ($0 \leq$ integer) 0
initial random seed value

`branching/relpscost/priority` ($-536870912 \leq$ integer ≤ 536870911) 10000
priority of branching rule <relpscost>

`branching/relpscost/maxdepth` ($-1 \leq$ integer) -1
maximal depth level, up to which branching rule <relpscost> should be used (-1 for no limit)

`branching/relpscost/maxbounddist` ($0 \leq$ real ≤ 1) 1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying branching rule (0.0: only on current best node, 1.0: on all nodes)

`branching/relpscost/sbiterquot` ($0 \leq$ real) 0.5
maximal fraction of strong branching LP iterations compared to node relaxation LP iterations

`branching/relpscost/sbiterofs` ($0 \leq$ integer) 100000
additional number of allowed strong branching LP iterations

`branching/relpscost/initcand` ($0 \leq$ integer) 100
maximal number of candidates initialized with strong branching per node

`branching/relpscost/inititer` ($0 \leq$ integer) 0
iteration limit for strong branching initializations of pseudo cost entries (0: auto)

Conflict analysis

`conflict/enable` (`boolean`) `TRUE`
should conflict analysis be enabled?

`conflict/useprop` (`boolean`) `TRUE`
should propagation conflict analysis be used?

`conflict/useinflp` (`boolean`) `TRUE`
should infeasible LP conflict analysis be used?

`conflict/useboundlp` (`boolean`) `FALSE`
should bound exceeding LP conflict analysis be used?

`conflict/usesb` (`boolean`) `FALSE`
should infeasible/bound exceeding strong branching conflict analysis be used?

`conflict/usepseudo` (`boolean`) `TRUE`
should pseudo solution conflict analysis be used?

`conflict/preferbinary` (`boolean`) `FALSE`
should binary conflicts be preferred?

`conflict/restartnum` ($0 \leq$ integer) 0
number of successful conflict analysis calls that trigger a restart (0: disable conflict restarts)

`conflict/restartfac` ($0 \leq$ real) 1.5
factor to increase restartnum with after each restart

Constraints

`constraints/linear/sepafreq` ($-1 \leq$ integer) 0
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/linear/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/linear/maxrounds` ($-1 \leq$ integer) 5
maximal number of separation rounds per node (-1: unlimited)

`constraints/linear/maxroundsroot` ($-1 \leq$ integer) -1
maximal number of separation rounds per node in the root node (-1: unlimited)

`constraints/linear/maxsepacuts` ($0 \leq$ integer) 50
maximal number of cuts separated per separation round

`constraints/linear/maxsepacutsroot` ($0 \leq$ integer) 200
maximal number of cuts separated per separation round in the root node

`constraints/linear/separateall` (`boolean`) `FALSE`
should all constraints be subject to cardinality cut generation instead of only the ones with non-zero dual value?

`constraints/and/sepafreq` ($-1 \leq$ integer) 1
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/and/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/bounddisjunction/sepafreq` ($-1 \leq$ integer) -1
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/bounddisjunction/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/conjunction/sepafreq` ($-1 \leq$ integer) -1
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/conjunction/propfreq` ($-1 \leq$ integer) -1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/countsols/sepafreq` ($-1 \leq$ integer) -1
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/countsols/propfreq` ($-1 \leq$ integer) -1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/countsols/active` (`boolean`) `FALSE`
is the constraint handler active?

`constraints/countsols/sparsetest` (`boolean`) `TRUE`
should the sparse solution test be turned on?

`constraints/countsols/discardsols` (`boolean`) `TRUE`
is it allowed to discard solutions?

`constraints/countsols/collect` (`boolean`) `FALSE`
should the solutions be collected?

`constraints/countsols/sollimit` ($-1 \leq$ integer ≤ -1) -1
counting stops, if the given number of solutions were found (-1: no limit)

`constraints/cumulative/sepafreq` ($-1 \leq$ integer) 1
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/cumulative/propfreq` ($-1 \leq$ integer) 5
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/cumulative/usebinvars` (`boolean`) `FALSE`
should the binary representation be used?

`constraints/cumulative/usecoretimes` (`boolean`) `TRUE`
should coretimes be propagated?

`constraints/cumulative/usecoretimesholes` (`boolean`) `FALSE`
should coretimes be propagated to detect holes?

`constraints/cumulative/localcuts` (`boolean`) `FALSE`
should cuts be added only locally?

`constraints/cumulative/usecovercuts` (`boolean`) `TRUE`
should covering cuts be added every node?

`constraints/cumulative/useedgefinding` (`boolean`) `FALSE`
should edge finding be used?

`constraints/cumulative/useenergeticreasoning` (`boolean`) `FALSE`
should energetic reasoning be used?

`constraints/cumulative/cutsasconss` (`boolean`) `TRUE`
should the cumulative constraint create cuts as knapsack constraints?

`constraints/indicator/sepafreq` ($-1 \leq$ integer) 10
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/indicator/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/integral/sepafreq` ($-1 \leq$ integer) -1
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/integral/propfreq` ($-1 \leq$ integer) -1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/knapsack/sepafreq` ($-1 \leq$ integer) 0
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/knapsack/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/linear/upgrade/knapsack` (`boolean`) `TRUE`
enable linear upgrading for constraint handler <knapsack>

`constraints/knapsack/maxrounds` ($-1 \leq$ integer) 5
maximal number of separation rounds per node (-1: unlimited)

`constraints/knapsack/maxroundsroot` ($-1 \leq$ integer) -1
maximal number of separation rounds per node in the root node (-1: unlimited)

`constraints/knapsack/maxsepacuts` ($0 \leq$ integer) 50
maximal number of cuts separated per separation round

`constraints/knapsack/maxsepacutsroot` ($0 \leq$ integer) 200
maximal number of cuts separated per separation round in the root node

`constraints/linking/sepafreq` ($-1 \leq$ integer) 1
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/linking/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/linking/linearize` (`boolean`) `FALSE`
this constraint will not propagate or separate, linear and setppc are used?

`constraints/logicor/sepafreq` ($-1 \leq$ integer) 0
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/logicor/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/linear/upgrade/logicor` (`boolean`) `TRUE`
enable linear upgrading for constraint handler <logicor>

`constraints/or/sepafreq` ($-1 \leq$ integer) 0
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/or/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/orbitope/sepafreq` ($-1 \leq$ integer) 5
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/orbitope/propfreq` ($-1 \leq$ integer) -1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/quadratic/sepafreq` ($-1 \leq$ integer) 2
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/quadratic/propfreq` ($-1 \leq$ integer) 10
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/quadratic/replacebinaryprod` ($0 \leq$ integer) ∞
max. length of linear term which when multiplied with a binary variables is replaced by an auxiliary variable and a linear reformulation (0 to turn off)

`constraints/quadratic/empathy4and` ($0 \leq$ integer ≤ 2) 0
empathy level for using the AND constraint handler: 0 always avoid using AND; 1 use AND sometimes; 2 use AND as often as possible

`constraints/quadratic/minefficacysepa` ($0 \leq$ real) 0.0001
minimal efficacy for a cut to be added to the LP during separation; overwrites separating/efficacy

`constraints/quadratic/minefficacyenfo` ($0 \leq$ real) $2 \cdot 10^{-6}$
minimal target efficacy of a cut in order to add it to relaxation during enforcement (may be ignored)

`constraints/quadratic/scaling` (`boolean`) `TRUE`
whether a quadratic constraint should be scaled w.r.t. the current gradient norm when checking for feasibility

`constraints/quadratic/cutmaxrange` ($0 \leq$ real) 10^{10}
maximal range of a cut (maximal coefficient divided by minimal coefficient) in order to be added to LP relaxation

`constraints/quadratic/linearizenlpsol` (`boolean`) `TRUE`
whether convex quadratic constraints should be linearized in a solution found by the NLP or RENS heuristic

`constraints/quadratic/checkcurvature` (`boolean`) `TRUE`
whether multivariate quadratic functions should be checked for convexity/concavity

`constraints/quadratic/linfeasshift` (`boolean`) `TRUE`
whether to try to make solutions in check function feasible by shifting a linear variable (esp. useful if constraint was actually objective function)

`constraints/setppc/sepafreq` ($-1 \leq$ integer) 0
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/setppc/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/linear/upgrade/setppc` (`boolean`) `TRUE`
enable linear upgrading for constraint handler <setppc>

`constraints/SOS1/sepafreq` ($-1 \leq$ integer) 0
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/SOS1/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/SOS1/branchSOS` (`boolean`) `TRUE`
Use SOS1 branching in enforcing (otherwise leave decision to branching rules)?

`constraints/SOS1/branchNonzeros` (`boolean`) `FALSE`
Branch on SOS constraint with most number of nonzeros?

`constraints/SOS1/branchWeight` (`boolean`) `FALSE`
Branch on SOS cons. with highest nonzero-variable weight for branching (needs branchNonzeros = false)?

`constraints/SOS2/sepafreq` ($-1 \leq$ integer) 0
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/SOS2/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/soc/sepafreq` ($-1 \leq$ integer) 1
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/soc/propfreq` ($-1 \leq$ integer) 20
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/quadratic/upgrade/soc` (`boolean`) `TRUE`
enable quadratic upgrading for constraint handler <soc>

`constraints/soc/nauxvars` ($0 \leq$ integer) 0
number of auxiliary variables to use when creating a linear outer approx. of a SOC3 constraint; 0 to turn off

`constraints/soc/glineur` (`boolean`) `TRUE`
whether the Glineur Outer Approximation should be used instead of Ben-Tal Nemirovski

`constraints/soc/linearizenlpsol` (`boolean`) `TRUE`
whether SOC constraints should be linearized in a solution found by the NLP or RENS heuristic

`constraints/soc/minefficacy` ($0 \leq$ real) 0.0001
minimal efficacy of a cut to be added to LP in separation

`constraints/soc/linfeasshift` (`boolean`) `TRUE`
whether to try to make solutions feasible in check by shifting the variable on the right hand side

`constraints/soc/nlpform` (`character`) `a`
which formulation to use when adding a SOC constraint to the NLP (a: automatic, q: nonconvex quadratic form, s: convex sqrt form, e: convex exponential-sqrt form)

`constraints/varbound/sepafreq` ($-1 \leq$ integer) 0
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/varbound/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

`constraints/linear/upgrade/varbound` (`boolean`) `TRUE`
enable linear upgrading for constraint handler <varbound>

`constraints/xor/sepafreq` ($-1 \leq$ integer) 0
frequency for separating cuts (-1: never, 0: only in root node)

`constraints/xor/propfreq` ($-1 \leq$ integer) 1
frequency for propagating domains (-1: never, 0: only in root node)

Output

`display/verblevel` ($0 \leq$ integer ≤ 5) 4
verbosity level of output

`display/width` ($0 \leq$ integer) 80
maximal number of characters in a node information line

`display/freq` ($-1 \leq$ integer) 100
frequency for displaying node information lines

`display/headerfreq` ($-1 \leq$ integer) 15
frequency for displaying header lines (every n'th node information line)

`display/lpinfo` (`boolean`) `FALSE`
should the LP solver display status messages?

`display/sols/active` (0 ≤ integer ≤ 2) 0
display activation status of display column <sols> (0: off, 1: auto, 2:on)

`display/feasST/active` (0 ≤ integer ≤ 2) 0
display activation status of display column <feasST> (0: off, 1: auto, 2:on)

`display/solfound/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <solfound> (0: off, 1: auto, 2:on)

`display/time/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <time> (0: off, 1: auto, 2:on)

`display/nnodes/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <nnodes> (0: off, 1: auto, 2:on)

`display/nodesleft/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <nodesleft> (0: off, 1: auto, 2:on)

`display/lpiterations/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <lpiterations> (0: off, 1: auto, 2:on)

`display/lpavgiterations/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <lpavgiterations> (0: off, 1: auto, 2:on)

`display/memused/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <memused> (0: off, 1: auto, 2:on)

`display/depth/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <depth> (0: off, 1: auto, 2:on)

`display/maxdepth/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <maxdepth> (0: off, 1: auto, 2:on)

`display/plungedepth/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <plungedepth> (0: off, 1: auto, 2:on)

`display/nfrac/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <nfrac> (0: off, 1: auto, 2:on)

`display/nexternbranchcands/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <nexternbranchcands> (0: off, 1: auto, 2:on)

`display/vars/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <vars> (0: off, 1: auto, 2:on)

`display/conss/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <conss> (0: off, 1: auto, 2:on)

`display/curconss/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <curconss> (0: off, 1: auto, 2:on)

`display/curcols/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <curcols> (0: off, 1: auto, 2:on)

`display/currows/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <currows> (0: off, 1: auto, 2:on)

`display/cuts/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <cuts> (0: off, 1: auto, 2:on)

`display/separounds/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <separounds> (0: off, 1: auto, 2:on)

`display/poolsize/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <poolsize> (0: off, 1: auto, 2:on)

`display/conflicts/active` (0 ≤ integer ≤ 2) 1
display activation status of display column <conflicts> (0: off, 1: auto, 2:on)

`display/strongbranchs/active` ($0 \leq$ integer ≤ 2) 1
display activation status of display column <strongbranchs> (0: off, 1: auto, 2:on)

`display/lpobj/active` ($0 \leq$ integer ≤ 2) 1
display activation status of display column <lpobj> (0: off, 1: auto, 2:on)

`display/curdualbound/active` ($0 \leq$ integer ≤ 2) 1
display activation status of display column <curdualbound> (0: off, 1: auto, 2:on)

`display/estimate/active` ($0 \leq$ integer ≤ 2) 1
display activation status of display column <estimate> (0: off, 1: auto, 2:on)

`display/avgdualbound/active` ($0 \leq$ integer ≤ 2) 1
display activation status of display column <avgdualbound> (0: off, 1: auto, 2:on)

`display/dualbound/active` ($0 \leq$ integer ≤ 2) 1
display activation status of display column <dualbound> (0: off, 1: auto, 2:on)

`display/primalbound/active` ($0 \leq$ integer ≤ 2) 1
display activation status of display column <primalbound> (0: off, 1: auto, 2:on)

`display/cutoffbound/active` ($0 \leq$ integer ≤ 2) 1
display activation status of display column <cutoffbound> (0: off, 1: auto, 2:on)

`display/gap/active` ($0 \leq$ integer ≤ 2) 1
display activation status of display column <gap> (0: off, 1: auto, 2:on)

`display/nsols/active` ($0 \leq$ integer ≤ 2) 1
display activation status of display column <nsols> (0: off, 1: auto, 2:on)

Heuristics

`heuristics/actconsdiving/freq` ($-1 \leq$ integer) -1
frequency for calling primal heuristic <actconsdiving> (-1: never, 0: only at depth freqofs)

`heuristics/actconsdiving/freqofs` ($0 \leq$ integer) 5
frequency offset for calling primal heuristic <actconsdiving>

`heuristics/actconsdiving/maxlpiterquot` ($0 \leq$ real) 0.05
maximal fraction of diving LP iterations compared to node LP iterations

`heuristics/actconsdiving/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/actconsdiving/backtrack` (`boolean`) `TRUE`
use one level of backtracking if infeasibility is encountered?

`heuristics/coefdiving/freq` ($-1 \leq$ integer) 10
frequency for calling primal heuristic <coefdiving> (-1: never, 0: only at depth freqofs)

`heuristics/coefdiving/freqofs` ($0 \leq$ integer) 1
frequency offset for calling primal heuristic <coefdiving>

`heuristics/coefdiving/maxlpiterquot` ($0 \leq$ real) 0.05
maximal fraction of diving LP iterations compared to node LP iterations

`heuristics/coefdiving/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/coefdiving/backtrack` (`boolean`) `TRUE`
use one level of backtracking if infeasibility is encountered?

`heuristics/crossover/freq` ($-1 \leq$ integer) 30
frequency for calling primal heuristic <crossover> (-1: never, 0: only at depth freqofs)

`heuristics/crossover/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <crossover>

`heuristics/crossover/nodesofs` ($0 \leq$ integer ≤ -1) 500

number of nodes added to the contingent of the total nodes

`heuristics/crossover/nusedsols` ($2 \leq$ integer) 3
number of solutions to be taken into account

`heuristics/crossover/nodesquot` ($0 \leq$ real ≤ 1) 0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem

`heuristics/crossover/minfixingrate` ($0 \leq$ real ≤ 1) 0.666
minimum percentage of integer variables that have to be fixed

`heuristics/dins/freq` ($-1 \leq$ integer) -1
frequency for calling primal heuristic <dins> (-1: never, 0: only at depth freqofs)

`heuristics/dins/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <dins>

`heuristics/dins/nodesofs` ($0 \leq$ integer ≤ -1) 5000
number of nodes added to the contingent of the total nodes

`heuristics/dins/nodesquot` ($0 \leq$ real ≤ 1) 0.05
contingent of sub problem nodes in relation to the number of nodes of the original problem

`heuristics/dins/minnodes` ($0 \leq$ integer ≤ -1) 500
minimum number of nodes required to start the subproblem

`heuristics/dins/solnum` ($1 \leq$ integer) 5
number of pool-solutions to be checked for flag array update (for hard fixing of binary variables)

`heuristics/dins/neighborhoodsize` ($1 \leq$ integer) 18
radius (using Manhattan metric) of the incumbent's neighborhood to be searched

`heuristics/feaspump/freq` ($-1 \leq$ integer) 20
frequency for calling primal heuristic <feaspump> (-1: never, 0: only at depth freqofs)

`heuristics/feaspump/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <feaspump>

`heuristics/feaspump/maxlpiterquot` ($0 \leq$ real) 0.01
maximal fraction of diving LP iterations compared to node LP iterations

`heuristics/feaspump/objfactor` ($0 \leq$ real ≤ 1) 1
factor by which the regard of the objective is decreased in each round, 1.0 for dynamic

`heuristics/feaspump/alphadiff` ($0 \leq$ real ≤ 1) 1
threshold difference for the convex parameter to perform perturbation

`heuristics/feaspump/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/feaspump/neighborhoodsize` ($1 \leq$ integer) 18
radius (using Manhattan metric) of the neighborhood to be searched in stage 3

`heuristics/feaspump/beforecuts` (`boolean`) `TRUE`
should the feasibility pump be called at root node before cut separation?

`heuristics/feaspump2/usefp20` (`boolean`) `FALSE`
should an iterative round-and-propagate scheme be used to find the integral points?

`heuristics/feaspump2/pertsolfound` (`boolean`) `TRUE`
should a random perturbation be performed if a feasible solution was found?

`heuristics/feaspump2/stage3` (`boolean`) `FALSE`
should we solve a local branching sub-MIP if no solution could be found?

`heuristics/fixandinfer/freq` ($-1 \leq$ integer) -1
frequency for calling primal heuristic <fixandinfer> (-1: never, 0: only at depth freqofs)

`heuristics/fixandinfer/freqofs` ($0 \leq$ integer) 0

frequency offset for calling primal heuristic <fixandinfer>

`heuristics/fracdiving/freq` ($-1 \leq$ integer) 10
frequency for calling primal heuristic <fracdiving> (-1: never, 0: only at depth freqofs)

`heuristics/fracdiving/freqofs` ($0 \leq$ integer) 3
frequency offset for calling primal heuristic <fracdiving>

`heuristics/fracdiving/maxlpiterquot` ($0 \leq$ real) 0.05
maximal fraction of diving LP iterations compared to node LP iterations

`heuristics/fracdiving/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/fracdiving/backtrack` (`boolean`) `TRUE`
use one level of backtracking if infeasibility is encountered?

`heuristics/guideddiving/freq` ($-1 \leq$ integer) 10
frequency for calling primal heuristic <guideddiving> (-1: never, 0: only at depth freqofs)

`heuristics/guideddiving/freqofs` ($0 \leq$ integer) 7
frequency offset for calling primal heuristic <guideddiving>

`heuristics/guideddiving/maxlpiterquot` ($0 \leq$ real) 0.05
maximal fraction of diving LP iterations compared to node LP iterations

`heuristics/guideddiving/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/guideddiving/backtrack` (`boolean`) `TRUE`
use one level of backtracking if infeasibility is encountered?

`heuristics/intdiving/freq` ($-1 \leq$ integer) -1
frequency for calling primal heuristic <intdiving> (-1: never, 0: only at depth freqofs)

`heuristics/intdiving/freqofs` ($0 \leq$ integer) 9
frequency offset for calling primal heuristic <intdiving>

`heuristics/intdiving/maxlpiterquot` ($0 \leq$ real) 0.05
maximal fraction of diving LP iterations compared to node LP iterations

`heuristics/intdiving/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/intdiving/backtrack` (`boolean`) `TRUE`
use one level of backtracking if infeasibility is encountered?

`heuristics/intshifting/freq` ($-1 \leq$ integer) 10
frequency for calling primal heuristic <intshifting> (-1: never, 0: only at depth freqofs)

`heuristics/intshifting/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <intshifting>

`heuristics/linesearchdiving/freq` ($-1 \leq$ integer) 10
frequency for calling primal heuristic <linesearchdiving> (-1: never, 0: only at depth freqofs)

`heuristics/linesearchdiving/freqofs` ($0 \leq$ integer) 6
frequency offset for calling primal heuristic <linesearchdiving>

`heuristics/linesearchdiving/maxlpiterquot` ($0 \leq$ real) 0.05
maximal fraction of diving LP iterations compared to node LP iterations

`heuristics/linesearchdiving/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/linesearchdiving/backtrack` (`boolean`) `TRUE`
use one level of backtracking if infeasibility is encountered?

`heuristics/localbranching/freq` ($-1 \leq$ integer) -1

frequency for calling primal heuristic <localbranching> (-1: never, 0: only at depth freqofs)

`heuristics/localbranching/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <localbranching>

`heuristics/localbranching/nodesofs` ($0 \leq$ integer) 1000
number of nodes added to the contingent of the total nodes

`heuristics/localbranching/neighborhoodsize` ($1 \leq$ integer) 18
radius (using Manhattan metric) of the incumbent's neighborhood to be searched

`heuristics/localbranching/nodesquot` ($0 \leq$ real ≤ 1) 0.05
contingent of sub problem nodes in relation to the number of nodes of the original problem

`heuristics/mutation/freq` ($-1 \leq$ integer) -1
frequency for calling primal heuristic <mutation> (-1: never, 0: only at depth freqofs)

`heuristics/mutation/freqofs` ($0 \leq$ integer) 8
frequency offset for calling primal heuristic <mutation>

`heuristics/mutation/nodesofs` ($0 \leq$ integer) 500
number of nodes added to the contingent of the total nodes

`heuristics/mutation/nodesquot` ($0 \leq$ real ≤ 1) 0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem

`heuristics/mutation/minfixingrate` ($10^{-6} \leq$ real ≤ 0.999999) 0.8
percentage of integer variables that have to be fixed

`heuristics/objpscostdiving/freq` ($-1 \leq$ integer) 20
frequency for calling primal heuristic <objpscostdiving> (-1: never, 0: only at depth freqofs)

`heuristics/objpscostdiving/freqofs` ($0 \leq$ integer) 4
frequency offset for calling primal heuristic <objpscostdiving>

`heuristics/objpscostdiving/maxlpiterquot` ($0 \leq$ real ≤ 1) 0.01
maximal fraction of diving LP iterations compared to total iteration number

`heuristics/objpscostdiving/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/octane/freq` ($-1 \leq$ integer) -1
frequency for calling primal heuristic <octane> (-1: never, 0: only at depth freqofs)

`heuristics/octane/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <octane>

`heuristics/oneopt/freq` ($-1 \leq$ integer) 1
frequency for calling primal heuristic <oneopt> (-1: never, 0: only at depth freqofs)

`heuristics/oneopt/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <oneopt>

`heuristics/pscostdiving/freq` ($-1 \leq$ integer) 10
frequency for calling primal heuristic <pscostdiving> (-1: never, 0: only at depth freqofs)

`heuristics/pscostdiving/freqofs` ($0 \leq$ integer) 2
frequency offset for calling primal heuristic <pscostdiving>

`heuristics/pscostdiving/maxlpiterquot` ($0 \leq$ real) 0.05
maximal fraction of diving LP iterations compared to node LP iterations

`heuristics/pscostdiving/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/pscostdiving/backtrack` (`boolean`) `TRUE`
use one level of backtracking if infeasibility is encountered?

`heuristics/rens/freq` ($-1 \leq$ integer) 0

frequency for calling primal heuristic <rens> (-1: never, 0: only at depth freqofs)

`heuristics/rens/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <rens>

`heuristics/rens/minfixingrate` ($0 \leq$ real ≤ 1) 0.5
minimum percentage of integer variables that have to be fixable

`heuristics/rens/nodesofs` ($0 \leq$ integer ≤ -1) 500
number of nodes added to the contingent of the total nodes

`heuristics/rens/nodesquot` ($0 \leq$ real ≤ 1) 0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem

`heuristics/rins/freq` ($-1 \leq$ integer) -1
frequency for calling primal heuristic <rins> (-1: never, 0: only at depth freqofs)

`heuristics/rins/freqofs` ($0 \leq$ integer) 5
frequency offset for calling primal heuristic <rins>

`heuristics/rins/nodesofs` ($0 \leq$ integer) 500
number of nodes added to the contingent of the total nodes

`heuristics/rins/nodesquot` ($0 \leq$ real ≤ 1) 0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem

`heuristics/rins/minfixingrate` ($0 \leq$ real ≤ 1) 0
minimum percentage of integer variables that have to be fixed

`heuristics/rootsoldiving/freq` ($-1 \leq$ integer) 20
frequency for calling primal heuristic <rootsoldiving> (-1: never, 0: only at depth freqofs)

`heuristics/rootsoldiving/freqofs` ($0 \leq$ integer) 5
frequency offset for calling primal heuristic <rootsoldiving>

`heuristics/rootsoldiving/maxlpiterquot` ($0 \leq$ real) 0.01
maximal fraction of diving LP iterations compared to node LP iterations

`heuristics/rootsoldiving/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/rounding/freq` ($-1 \leq$ integer) 1
frequency for calling primal heuristic <rounding> (-1: never, 0: only at depth freqofs)

`heuristics/rounding/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <rounding>

`heuristics/shiftandpropagate/freq` ($-1 \leq$ integer) 0
frequency for calling primal heuristic <shiftandpropagate> (-1: never, 0: only at depth freqofs)

`heuristics/shiftandpropagate/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <shiftandpropagate>

`heuristics/shifting/freq` ($-1 \leq$ integer) 10
frequency for calling primal heuristic <shifting> (-1: never, 0: only at depth freqofs)

`heuristics/shifting/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <shifting>

`heuristics/simplerounding/freq` ($-1 \leq$ integer) 1
frequency for calling primal heuristic <simplerounding> (-1: never, 0: only at depth freqofs)

`heuristics/simplerounding/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <simplerounding>

`heuristics/subnlp/freq` ($-1 \leq$ integer) 1
frequency for calling primal heuristic <subnlp> (-1: never, 0: only at depth freqofs)

`heuristics/subnlp/freqofs` ($0 \leq$ integer) 0

frequency offset for calling primal heuristic <subnlp>

`heuristics/subnlp/nlpverblevel` ($0 \leq$ integer) 0
verbosity level of NLP solver

`heuristics/subnlp/nlpiterlimit` ($0 \leq$ integer) 0
iteration limit of NLP solver; 0 to use solver default

`heuristics/subnlp/nlptimelimit` ($0 \leq$ real) 0
time limit of NLP solver; 0 to use solver default

`heuristics/subnlp/nlpsolver` (`string`)
name of an NLP solver to use (empty value means to use solver with highest priority)

`heuristics/subnlp/iteroffset` ($0 \leq$ integer) 500
number of iterations added to the contingent of the total number of iterations

`heuristics/subnlp/iterquotient` ($0 \leq$ real) 0.1
contingent of NLP iterations in relation to the number of nodes in SCIP

`heuristics/subnlp/itermin` ($0 \leq$ integer) 300
contingent of NLP iterations in relation to the number of nodes in SCIP

`heuristics/subnlp/runalways` (`boolean`) `FALSE`
whether to run NLP heuristic always if starting point available (does not use iteroffset,iterquot,itermin)

`heuristics/subnlp/forbidfixings` (`boolean`) `TRUE`
whether to add constraints that forbid specific fixings that turned out to be infeasible

`heuristics/trivial/freq` ($-1 \leq$ integer) 0
frequency for calling primal heuristic <trivial> (-1: never, 0: only at depth freqofs)

`heuristics/trivial/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <trivial>

`heuristics/trysol/freq` ($-1 \leq$ integer) 1
frequency for calling primal heuristic <trysol> (-1: never, 0: only at depth freqofs)

`heuristics/trysol/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <trysol>

`heuristics/twoopt/freq` ($-1 \leq$ integer) -1
frequency for calling primal heuristic <twoopt> (-1: never, 0: only at depth freqofs)

`heuristics/twoopt/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <twoopt>

`heuristics/undercover/freq` ($-1 \leq$ integer) 0
frequency for calling primal heuristic <undercover> (-1: never, 0: only at depth freqofs)

`heuristics/undercover/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <undercover>

`heuristics/undercover/fixingalts` (`string`) `li`
prioritized sequence of fixing values used ('l'p relaxation, 'n'lp relaxation, 'i'ncumbent solution)

`heuristics/undercover/nodesofs` ($0 \leq$ integer ≤ -1) 500
number of nodes added to the contingent of the total nodes

`heuristics/undercover/nodesquot` ($0 \leq$ real ≤ 1) 0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem

`heuristics/undercover/onlyconvexify` (`boolean`) `FALSE`
should we only fix variables in order to obtain a convex problem?

`heuristics/undercover/postnlp` (`boolean`) `TRUE`
should the nlp heuristic be called to polish a feasible solution?

`heuristics/veclendiving/freq` ($-1 \leq$ integer) 10

frequency for calling primal heuristic <veclendiving> (-1: never, 0: only at depth freqofs)

`heuristics/veclendiving/freqofs` ($0 \leq$ integer) 4
frequency offset for calling primal heuristic <veclendiving>

`heuristics/veclendiving/maxlpiterquot` ($0 \leq$ real) 0.05
maximal fraction of diving LP iterations compared to node LP iterations

`heuristics/veclendiving/maxlpiterofs` ($0 \leq$ integer) 1000
additional number of allowed LP iterations

`heuristics/veclendiving/backtrack` (`boolean`) `TRUE`
use one level of backtracking if infeasibility is encountered?

`heuristics/zirounding/freq` ($-1 \leq$ integer) 1
frequency for calling primal heuristic <zirounding> (-1: never, 0: only at depth freqofs)

`heuristics/zirounding/freqofs` ($0 \leq$ integer) 0
frequency offset for calling primal heuristic <zirounding>

Limits

`limits/time` ($0 \leq$ real) 1000
maximal time in seconds to run

`limits/nodes` ($-1 \leq$ integer ≤ -1) -1
maximal number of nodes to process (-1: no limit)

`limits/stallnodes` ($-1 \leq$ integer ≤ -1) -1
solving stops, if the given number of nodes was processed since the last improvement of the primal solution value (-1: no limit)

`limits/memory` ($0 \leq$ real) ∞
maximal memory usage in MB; reported memory usage is lower than real memory usage!

`limits/gap` ($0 \leq$ real) 0.1
solving stops, if the relative gap = —(primalbound - dualbound)/dualbound— is below the given value

`limits/absgap` ($0 \leq$ real) 0
solving stops, if the absolute gap = —primalbound - dualbound— is below the given value

`limits/solutions` ($-1 \leq$ integer) -1
solving stops, if the given number of solutions were found (-1: no limit)

`limits/bestsol` ($-1 \leq$ integer) -1
solving stops, if the given number of solution improvements were found (-1: no limit)

`limits/maxsol` ($1 \leq$ integer) 100
maximal number of solutions to store in the solution storage

`limits/restarts` ($-1 \leq$ integer) -1
solving stops, if the given number of restarts was triggered (-1: no limit)

LP

`lp/solvefreq` ($-1 \leq$ integer) 1
frequency for solving LP at the nodes (-1: never; 0: only root LP)

`lp/solvedepth` ($-1 \leq$ integer) -1
maximal depth for solving LP at the nodes (-1: no depth limit)

`lp/initalgorithm` (`character`) `s`
LP algorithm for solving initial LP relaxations (automatic 's'implex, 'p'rimal simplex, 'd'ual simplex, 'b'arrier, barrier with 'c'rossover)

`lp/resolvealgorithm` (`character`) `s`
LP algorithm for resolving LP relaxations if a starting basis exists (automatic 's'implex, 'p'rimal simplex, 'd'ual simplex, 'b'arrier, barrier with 'c'rossover)

`lp/pricing` (`character`) `l`
LP pricing strategy ('l'pi default, 'a'uto, 'f'ull pricing, 'p'artial, 's'teepest edge pricing, 'q'uickstart steepest edge pricing, 'd'evex pricing)

Memory

`memory/savefac` ($0 \leq$ real ≤ 1) 0.8
fraction of maximal memory usage resulting in switch to memory saving mode

Micellaneous

`misc/catchctrlc` (`boolean`) `TRUE`
should the CTRL-C interrupt be caught by SCIP?

`misc/usevartable` (`boolean`) `TRUE`
should a hashtable be used to map from variable names to variables?

`misc/useconstable` (`boolean`) `TRUE`
should a hashtable be used to map from constraint names to constraints?

`misc/usesmalltables` (`boolean`) `FALSE`
should smaller hashtables be used? yields better performance for small problems with about 100 variables

`misc/permutationseed` ($-1 \leq$ integer) -1
seed value for permuting the problem after the problem was transformed (-1: no permutation)

Node Selection

`nodeselection/childsel` (`character`) `h`
child selection rule ('d'own, 'u'p, 'p'seudo costs, 'i'nference, 'l'p value, 'r'oot LP value difference, 'h'brid inference/root LP value difference)

`nodeselection/bfs/stdpriority` ($-536870912 \leq$ integer ≤ 536870911) 100000
priority of node selection rule <bfs> in standard mode

`nodeselection/dfs/stdpriority` ($-536870912 \leq$ integer ≤ 536870911) 0
priority of node selection rule <dfs> in standard mode

`nodeselection/estimate/stdpriority` ($-536870912 \leq$ integer ≤ 536870911) 200000
priority of node selection rule <estimate> in standard mode

`nodeselection/estimate/bestnodefreq` ($0 \leq$ integer) 10
frequency at which the best node instead of the best estimate is selected (0: never)

`nodeselection/hybridestim/stdpriority` ($-536870912 \leq$ integer ≤ 536870911) 50000
priority of node selection rule <hybridestim> in standard mode

`nodeselection/hybridestim/bestnodefreq` ($0 \leq$ integer) 1000
frequency at which the best node instead of the hybrid best estimate / best bound is selected (0: never)

`nodeselection/restartdfs/stdpriority` ($-536870912 \leq$ integer ≤ 536870911) 10000
priority of node selection rule <restartdfs> in standard mode

`nodeselection/restartdfs/selectbestfreq` ($0 \leq$ integer) 0
frequency for selecting the best node instead of the deepest one

Tolerances

`numerics/epsilon` ($10^{-20} \leq$ real ≤ 0.001) 10^{-9}
absolute values smaller than this are considered zero

`numerics/sumepsilon` ($10^{-17} \leq$ real ≤ 0.001) 10^{-6}
absolute values of sums smaller than this are considered zero

`numerics/feastol` ($10^{-17} \leq$ real ≤ 0.001) 10^{-6}
feasibility tolerance for constraints

`numerics/dualfeastol` ($10^{-17} \leq$ real ≤ 0.001) 10^{-9}
feasibility tolerance for reduced costs in LP solution

Presolving

`presolving/maxrounds` ($-1 \leq$ integer) -1
maximal number of presolving rounds (-1: unlimited, 0: off)

`presolving/maxrestarts` ($-1 \leq$ integer) -1
maximal number of restarts (-1: unlimited)

`presolving/boundshift/maxrounds` ($-1 \leq$ integer) 0
maximal number of presolving rounds the presolver participates in (-1: no limit)

`presolving/dualfix/maxrounds` ($-1 \leq$ integer) -1
maximal number of presolving rounds the presolver participates in (-1: no limit)

`presolving/implics/maxrounds` ($-1 \leq$ integer) -1
maximal number of presolving rounds the presolver participates in (-1: no limit)

`presolving/inttobinary/maxrounds` ($-1 \leq$ integer) -1
maximal number of presolving rounds the presolver participates in (-1: no limit)

`presolving/probing/maxrounds` ($-1 \leq$ integer) -1
maximal number of presolving rounds the presolver participates in (-1: no limit)

`presolving/probing/maxruns` ($-1 \leq$ integer) 1
maximal number of runs, probing participates in (-1: no limit)

`presolving/trivial/maxrounds` ($-1 \leq$ integer) -1
maximal number of presolving rounds the presolver participates in (-1: no limit)

Domain Propagation

`propagating/maxrounds` ($-1 \leq$ integer) 100
maximal number of propagation rounds per node (-1: unlimited)

`propagating/maxroundsroot` ($-1 \leq$ integer) 1000
maximal number of propagation rounds in the root node (-1: unlimited)

`propagating/abortoncutoff` (`boolean`) `TRUE`
should propagation be aborted immediately? setting this to FALSE could help conflict analysis to produce more conflict constraints

`propagating/pseudoobj/freq` ($-1 \leq$ integer) 1
frequency for calling propagator <pseudoobj> (-1: never, 0: only in root node)

`propagating/rootredcost/freq` ($-1 \leq$ integer) 1
frequency for calling propagator <rootredcost> (-1: never, 0: only in root node)

`propagating/vbounds/freq` ($-1 \leq$ integer) 1
frequency for calling propagator <vbounds> (-1: never, 0: only in root node)

Separation

`separating/maxbounddist` ($0 \leq$ real ≤ 1) 1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separation (0.0: only on current best node, 1.0: on all nodes)

`separating/minefficacy` ($0 \leq$ real) 0.05
minimal efficacy for a cut to enter the LP

`separating/minefficacyroot` ($0 \leq$ real) 0.01
minimal efficacy for a cut to enter the LP in the root node

`separating/minortho` ($0 \leq$ real ≤ 1) 0.5
minimal orthogonality for a cut to enter the LP

`separating/minorthoroot` ($0 \leq$ real ≤ 1) 0.5
minimal orthogonality for a cut to enter the LP in the root node

`separating/maxrounds` ($-1 \leq$ integer) 5

maximal number of separation rounds per node (-1: unlimited)

`separating/maxroundsroot` ($-1 \leq$ integer) -1
maximal number of separation rounds in the root node (-1: unlimited)

`separating/maxstallrounds` ($-1 \leq$ integer) 5
maximal number of consecutive separation rounds without objective or integrality improvement (-1: no additional restriction)

`separating/maxcuts` ($0 \leq$ integer) 100
maximal number of cuts separated per separation round (0: disable local separation)

`separating/maxcutsroot` ($0 \leq$ integer) 2000
maximal number of separated cuts at the root node (0: disable root node separation)

`separating/poolfreq` ($-1 \leq$ integer) 0
separation frequency for the global cut pool (-1: disable global cut pool, 0: only separate pool at the root)

`separating/clique/freq` ($-1 \leq$ integer) 0
frequency for calling separator <clique> (-1: never, 0: only in root node)

`separating/clique/maxsepacuts` ($-1 \leq$ integer) 10
maximal number of clique cuts separated per separation round (-1: no limit)

`separating/cgmip/freq` ($-1 \leq$ integer) -1
frequency for calling separator <cgmip> (-1: never, 0: only in root node)

`separating/cgmip/maxrounds` ($-1 \leq$ integer) 0
maximal number of cgmip separation rounds per node (-1: unlimited)

`separating/cgmip/maxroundsroot` ($-1 \leq$ integer) 50
maximal number of cgmip separation rounds in the root node (-1: unlimited)

`separating/cgmip/dynamiccuts` (`boolean`) `TRUE`
should generated cuts be removed from the LP if they are no longer tight?

`separating/cgmip/nodelimit` ($-1 \leq$ integer ≤ -1) 10000
node limit for sub-MIP (-1: unlimited)

`separating/cgmip/usecmir` (`boolean`) `TRUE`
use CMIR-generator (otherwise add cut directly)?

`separating/cgmip/cmirownbounds` (`boolean`) `FALSE`
tell CMIR-generator which bounds to used in rounding?

`separating/cgmip/allowlocal` (`boolean`) `FALSE`
allow to generate local cuts?

`separating/cgmip/onlyintvars` (`boolean`) `FALSE`
generate cuts for problems with only integer variables?

`separating/cgmip/onlyactiverows` (`boolean`) `TRUE`
use only active rows to generate cuts?

`separating/cgmip/usecutpool` (`boolean`) `TRUE`
use cutpool to store CG-cuts even if the are not efficient?

`separating/cgmip/primalseparation` (`boolean`) `TRUE`
only separate cuts that are tight for the best feasible solution?

`separating/cgmip/onlyrankone` (`boolean`) `FALSE`
whether only rank 1 inequalities should be separated

`separating/cgmip/earlyterm` (`boolean`) `TRUE`
terminate separation if a violated (but possibly sub-optimal) cut has been found?

`separating/cgmip/addviolationcons` (`boolean`) `TRUE`
add constraint to subscip that only allows violated cuts?

`separating/cgmip/addviolconshdlr` (`boolean`) `FALSE`
add constraint handler to filter out violated cuts?

`separating/cgmip/conshdlrusenorm` (`boolean`) `TRUE`
should the violation constraint handler use the norm of a cut to check for feasibility?

`separating/cgmip/objlone` (`boolean`) `FALSE`
should the objective of the sub-MIP minimize the l1-norm of the multipliers?

`separating/cmir/freq` ($-1 \leq$ integer) 0
frequency for calling separator <cmir> (-1: never, 0: only in root node)

`separating/cmir/maxrounds` ($-1 \leq$ integer) 3
maximal number of cmir separation rounds per node (-1: unlimited)

`separating/cmir/maxroundsroot` ($-1 \leq$ integer) 10
maximal number of cmir separation rounds in the root node (-1: unlimited)

`separating/cmir/maxsepacuts` ($0 \leq$ integer) 100
maximal number of cmir cuts separated per separation round

`separating/cmir/maxsepacutsroot` ($0 \leq$ integer) 500
maximal number of cmir cuts separated per separation round in the root node

`separating/cmir/dynamiccuts` (`boolean`) `TRUE`
should generated cuts be removed from the LP if they are no longer tight?

`separating/flowcover/freq` ($-1 \leq$ integer) 0
frequency for calling separator <flowcover> (-1: never, 0: only in root node)

`separating/flowcover/maxrounds` ($-1 \leq$ integer) 5
maximal number of separation rounds per node (-1: unlimited)

`separating/flowcover/maxroundsroot` ($-1 \leq$ integer) 10
maximal number of separation rounds in the root node (-1: unlimited)

`separating/flowcover/maxsepacuts` ($0 \leq$ integer) 100
maximal number of flow cover cuts separated per separation round

`separating/flowcover/maxsepacutsroot` ($0 \leq$ integer) 200
maximal number of flow cover cuts separated per separation round in the root

`separating/flowcover/dynamiccuts` (`boolean`) `TRUE`
should generated cuts be removed from the LP if they are no longer tight?

`separating/gomory/freq` ($-1 \leq$ integer) 0
frequency for calling separator <gomory> (-1: never, 0: only in root node)

`separating/gomory/maxrounds` ($-1 \leq$ integer) 5
maximal number of gomory separation rounds per node (-1: unlimited)

`separating/gomory/maxroundsroot` ($-1 \leq$ integer) -1
maximal number of gomory separation rounds in the root node (-1: unlimited)

`separating/gomory/maxsepacuts` ($0 \leq$ integer) 50
maximal number of gomory cuts separated per separation round

`separating/gomory/maxsepacutsroot` ($0 \leq$ integer) 500
maximal number of gomory cuts separated per separation round in the root node

`separating/gomory/dynamiccuts` (`boolean`) `TRUE`
should generated cuts be removed from the LP if they are no longer tight?

`separating/impliedbounds/freq` ($-1 \leq$ integer) 0
frequency for calling separator <impliedbounds> (-1: never, 0: only in root node)

`separating/intobj/freq` ($-1 \leq$ integer) -1
frequency for calling separator <intobj> (-1: never, 0: only in root node)

`separating/mcf/freq` ($-1 \leq$ integer) 0
frequency for calling separator <mcf> (-1: never, 0: only in root node)

`separating/mcf/dynamiccuts` (`boolean`) `TRUE`
should generated cuts be removed from the LP if they are no longer tight?

`separating/mcf/maxsepacuts` ($-1 \leq$ integer) 100
maximal number of mcf cuts separated per separation round

`separating/mcf/maxsepacutsroot` ($-1 \leq$ integer) 200
maximal number of mcf cuts separated per separation round in the root node – default separation

`separating/oddcycle/freq` ($-1 \leq$ integer) -1
frequency for calling separator <oddcycle> (-1: never, 0: only in root node)

`separating/oddcycle/useclassical` (`boolean`) `TRUE`
should classical search method by Groetschel, Lovasz, Schrijver be used? Otherwise use levelgraph method by Hoffman, Padberg.

`separating/oddcycle/liftoddcycles` (`boolean`) `FALSE`
should odd cycle cuts be lifted?

`separating/oddcycle/maxsepacuts` ($0 \leq$ integer) 5000
maximal number of oddcycle cuts separated per separation round

`separating/oddcycle/maxsepacutsroot` ($0 \leq$ integer) 5000
maximal number of oddcycle cuts separated per separation round in the root node

`separating/oddcycle/maxrounds` ($-1 \leq$ integer) 10
maximal number of oddcycle separation rounds per node (-1: unlimited)

`separating/oddcycle/maxroundsroot` ($-1 \leq$ integer) 10
maximal number of oddcycle separation rounds in the root node (-1: unlimited)

`separating/rapidlearning/freq` ($-1 \leq$ integer) -1
frequency for calling separator <rapidlearning> (-1: never, 0: only in root node)

`separating/redcost/freq` ($-1 \leq$ integer) 1
frequency for calling separator <redcost> (-1: never, 0: only in root node)

`separating/redcost/continuous` (`boolean`) `FALSE`
should reduced cost fixing be also applied to continuous variables?

`separating/strongcg/freq` ($-1 \leq$ integer) 0
frequency for calling separator <strongcg> (-1: never, 0: only in root node)

`separating/strongcg/maxrounds` ($-1 \leq$ integer) 5
maximal number of strong CG separation rounds per node (-1: unlimited)

`separating/strongcg/maxroundsroot` ($-1 \leq$ integer) 20
maximal number of strong CG separation rounds in the root node (-1: unlimited)

`separating/strongcg/maxsepacuts` ($0 \leq$ integer) 50
maximal number of strong CG cuts separated per separation round

`separating/strongcg/maxsepacutsroot` ($0 \leq$ integer) 500
maximal number of strong CG cuts separated per separation round in the root node

`separating/strongcg/dynamiccuts` (`boolean`) `TRUE`
should generated cuts be removed from the LP if they are no longer tight?

`separating/zerohalf/freq` ($-1 \leq$ integer) -1
frequency for calling separator <zerohalf> (-1: never, 0: only in root node)

`separating/zerohalf/maxrounds` ($-1 \leq$ integer) 5
maximal number of zerohalf separation rounds per node (-1: unlimited)

`separating/zerohalf/maxroundsroot` ($-1 \leq$ integer) 10
maximal number of zerohalf separation rounds in the root node (-1: unlimited)

`separating/zerohalf/maxsepacuts` ($0 \leq$ integer) 50
maximal number of 0,1/2-cuts separated per separation round

`separating/zerohalf/maxsepacutsroot` ($0 \leq$ integer) 500
maximal number of 0,1/2-cuts separated per separation round in the root node

`separating/zerohalf/dynamiccuts` (`boolean`) `TRUE`
should generated cuts be removed from the LP if they are no longer tight?

`separating/zerohalf/preprocessing/decomposeproblem` (`boolean`) `FALSE`
should problem be decomposed into subproblems (if possible) before applying preprocessing?

`separating/zerohalf/preprocessing/delta` ($0 \leq$ real ≤ 1) 0.5
value of delta parameter used in preprocessing method 'd'

`separating/zerohalf/preprocessing/ppmethods` (`string`) `CXGXIM`
preprocessing methods and ordering:
'd' columns with small LP solution,
'G' modified Gaussian elimination,
'i' identical columns,
'I' identical rows,
'L' large slack rows,
'M' large slack rows (minslack),
's' column singletons,
'X' add trivial zerohalf cuts,
'z' zero columns,
'Z' zero rows,
'C' fast 'z','s',
'R' fast 'Z','L','I'
#
'-' no preprocessing

`separating/zerohalf/separating/forcecutstolp` (`boolean`) `FALSE`
should the cuts be forced to enter the LP?

`separating/zerohalf/separating/forcecutstosepastore` (`boolean`) `FALSE`
should the cuts be forced to enter SCIP's sepastore?

`separating/zerohalf/separating/minviolation` ($0.001 \leq$ real ≤ 0.5) 0.3
minimal violation of a 0,1/2-cut to be separated

`separating/zerohalf/separating/sepamethods` (`string`) `2g`
separating methods and ordering:
'!' stop further processing if a cut was found,
'2' exact polynomial time algorithm (only if matrix has max 2 odd entries per row),
'e' enumeration heuristics (k=1: try all preprocessed rows),
'E' enumeration heuristics (k=2: try all combinations of up to two preprocessed rows),
'g' Extended Gaussian elimination heuristics,
's' auxiliary IP heuristics (i.e. number of solved nodes is limited)
'S' auxiliary IP exact (i.e. unlimited number of nodes)
#
'-' no processing

`separating/zerohalf/separating/auxip/settingsfile` (`string`) `-`
optional settings file of the auxiliary IP (-: none)

`separating/zerohalf/separating/auxip/sollimit` ($-1 \leq$ integer) -1
limits/solutions setting of the auxiliary IP

`separating/zerohalf/separating/auxip/penaltyfactor` ($0 \leq$ real ≤ 1) 0.001
penalty factor used with objective function 'p' of auxiliary IP

`separating/zerohalf/separating/auxip/useallsols` (`boolean`) `TRUE`
should all (proper) solutions of the auxiliary IP be used to generate cuts instead of using only the best?

`separating/zerohalf/separating/auxip/objective` (`character`) `v`
auxiliary IP objective:
'v' maximize cut violation,
'u' minimize number of aggregated rows in cut,
'w' minimize number of aggregated rows in cut
weighted by the number of rows in the aggregation,
'p' maximize cut violation and penalize a high number
of aggregated rows in the cut weighted by the number
# of rows in the aggregation and the penalty factor p

Timing

`timing/clocktype` ($1 \leq$ integer ≤ 2) 1
default clock type (1: CPU user seconds, 2: wall clock time)

`timing/enabled` (`boolean`) `TRUE`
is timing enabled?

SCIP References

[1] Tobias Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007.

[2] Tobias Achterberg. SCIP: Solving Constraint Integer Programs. *Mathematical Programming Computations*, 1(1):1–41, 2009.

[3] Tobias Achterberg, Timo Berthold, Thorsten Koch, and Kati Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In L. Perron and M.A. Trick, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 5th International Conference, CPAIOR 2008*, volume 5015 of *LNCS*, pages 6–20. Springer, 2008.

[4] Timo Berthold. Primal heuristics for mixed integer programs. Diploma thesis, Technische Universität Berlin, 2006.

[5] Timo Berthold, Stefan Heinz, and Stefan Vigerske. Extending a CIP framework to solve MIQCPs. In Jon Lee and Sven Leyffer, editors, *Mixed-integer nonlinear optimization: Algorithmic advances and applications*, IMA volumes in Mathematics and its Applications. Springer, 2009. to appear.

[6] Andreas Wächter and Lorenz T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. http://projects.coin-or.org/Ipopt.

[7] Kati Wolter. Implementation of cutting plane separators for mixed integer programs. Diploma thesis, Technische Universität Berlin, 2006.

[8] Roland Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus.* PhD thesis, Technische Universität Berlin, 1996. http://www.zib.de/Publications/abstracts/TR-96-09, http://soplex.zib.de.