

SCIP

Stefan Vigerske, Humboldt University Berlin, Germany

Contents

1	Introduction	1
2	Model requirements	1
3	Usage	2
3.1	Specification of SCIP Options	2
3.2	Specification of Indicators	2
4	Special Features	3
4.1	SCIP interactive shell	3
4.2	Emphasis Settings	3
4.3	Solution Pool	3
4.4	Branch-and-Bound tracing	3
4.5	Notes on solving MINLPs with SCIP	4
5	Detailed Options Description	4

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 main developers are Timo Berthold, Gerald Gamrath, Stefan Heinz, Marc Pfetsch, Stefan Vigerske, 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 or MINLP solver or as a framework for branch-cut-and-price. Within GAMS, the MIP and MINLP solving facilities of SCIP are available.

For more detailed information, we refer to [1, 2, 3, 4, 5, 6, 8] 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 SOPLEXP [9] as LP solver, the COIN-OR Interior Point Optimizer IPOPT [7] as nonlinear solver, and CPPAD to compute derivatives of nonlinear functions.

2 Model requirements

SCIP supports continuous, binary, integer, semi-continuous, semi-integer, and indicator variables, special ordered sets, and branching priorities for discrete variables.

3 Usage

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

```
Option MIP = SCIP;      { or QCP or NLP or MIQCP or MINLP or ... }
```

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.

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 ≤ 23.3 , available at http://www.gams.com/download/download_old.htm.

3.1 Specification of SCIP Options

GAMS/SCIP 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:

```
propagating/probing/maxprerounds = 0
separating/maxrounds             = 0
separating/maxroundsroot         = 0
```

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

3.2 Specification of Indicators

Indicators are a modeling tool to specify that certain equations in a model must only be satisfied if certain binary variables take a specified value. Indicators are not supported by the GAMS language, but can be passed to SCIP via a separate file. The name of that file is specified via the option `gams/indicatorfile` in a SCIP option file.

The indicator specification file declares for some equations, for which value of which binary variables the equation is “switched on”. The syntax is

```
indic equation$variable onval
```

where `equation` is the name of the equation, `variable` is the name of the binary variable, and `onval` is either 0 or 1. The line specifies that `equation` has to hold whenever variable `variable` takes value `onval`.

For example, assume a GAMS model contains a set of equations of the form

```
equ1(i,j,k)$ (ord(i)<ord(j)).. lhs =l= rhs;
```

To specify that they only have to be satisfied if a binary variable

```
bin1(i,k)
```

takes the value 1, the indicator specification file should contain the line

```
indic equ1(i,j,k)$bin1(i,k) 1
```

More documentation can be found at <http://www.gams.com/solvers/cpxindic.htm>. In difference to the GAMS/CPLEX interface, the indicator specifications need to be in a separate file for SCIP.

Currently, indicators can only be used for linear equations.

4 Special Features

4.1 SCIP interactive shell

The interactive shell in SCIP is a powerful tool that allows the user to display various information (e.g., branching statistics, presolved model), load emphasis settings, interrupt a solve to change parameters or trigger a restart, write the model in various file formats, start SCIPs solution counter, and many more things.

When setting the option `gams/interactive` to TRUE, the GAMS/SCIP interface opens the interactive shell of SCIP after having load the GAMS problem and parameters. The command `help` prints a list of available commands.

A tutorial on using the SCIP shell is available at <http://scip.zib.de/doc/html/SHELL.html>.

4.2 Emphasis Settings

SCIP includes various emphasis settings, which are predefined values for a set of SCIP parameters. Such predefined settings are available for setting the effort that SCIP should spend for, e.g., presolving, separation, or heuristics.

The emphasis settings are not available as single parameters, but can be set via SCIPs interactive shell. E.g., writing `set heuristics emphasis` in the shell displays the available emphasis settings for heuristics (aggressive, fast, off) and expects the user to input which setting to use. Further, general emphasis settings are available in the `set emphasis` menu, some of them giving predefined settings similar to the CPLEX option `mipemphasis`.

Further, settings file that specify all available emphasis settings are available at <http://www.gams.com/~svigerske/scip2.1>.

4.3 Solution Pool

When SCIP solves a problem, it may find several solutions, whereof only the best one is available to the GAMS user via the variable level values in the GAMS model. If the option `gams/dumpsolutions` is specified, then all alternative solutions found by SCIP are writing into GDX files and an index file with the name given by the `dumpsolutions` option is written.

The GAMS testlib model `dumpsol` shows an example use for this option via GUROBI. It can easily be adapted to be used with SCIP.

4.4 Branch-and-Bound tracing

The option `gams/miptrace/file` can be used to specify the name of a file where information about the progress of the tree search in SCIP are stored. The file is created and updated during the solution process, so it may also be used to monitor the progress of SCIP while it still solves the model.

New entries are written periodically, depending on how many nodes have been processed or how much time has been elapsed since the last entry was written. Each entry contains information on the current primal and dual bound.

4.5 Notes on solving MINLPs with SCIP

SCIP includes capabilities to handle nonlinear equations specified that are specified via algebraic expressions. Thus, external functions are not supported. Also not all GAMS operands are supported yet, including trigonometric functions (sin, cos, ...).

Note, that the support of MINLPs in SCIP is still a very new feature and thus in a **BETA** stage. Especially for badly formulated models, SCIP may produce no or wrong solutions.

Nonconvex MINLPs are solved by SCIP via a spatial branch-and-bound algorithm using convex relaxations. The tightness of a convex relaxation depends heavily on the variable bounds, thus tight bounds for the nonlinear variables are crucial for SCIP.

Special options for convex MINLPs

Convex MINLPs are much easier to solve for SCIP, provided it recognizes the convexity of the model. So far, only a simple convexity check is implemented in SCIP, which may not give a conclusive answer in all cases. However, the option `constraints/nonlinear/assumeconvex = TRUE` can be used to tell SCIP that it should assume all nonlinear constraints to be of convex type. This may help to improve solving times for convex MINLPs considerably.

Another useful feature especially for convex MINLPs is to enable the generation of cuts in the solution of the NLP relaxation in the root node and to consider using these cuts during the whole solution process. This is achieved by the parameters

```
constraints/quadratic/sepanlpmincont = 0
constraints/soc/sepanlpmincont = 0
constraints/nonlinear/sepanlpmincont = 0
constraints/abspower/sepanlpmincont = 0
separating/poolfreq = 1
```

5 Detailed Options Description

SCIP supports a large set of options. Sample option files can be obtained from <http://www.gams.com/~svigerske/scip2.1>.

In the following we give a detailed list of most SCIP options.

GAMS interface specific options

<code>gams/dumpsolutions</code> (string)	
name of solutions index gdx file for writing all solutions	
<code>gams/indicatorfile</code> (string)	
name of GAMS options file that contains definitions on indicators	
<code>gams/interactive</code> (boolean)	FALSE
whether a SCIP shell should be opened instead of issuing a solve command	
<code>gams/mipstart</code> (boolean)	TRUE
whether to try GAMS variable level values as initial primal solution	
<code>gams/miptrace/file</code> (string)	
name of file where to write branch-and-bound trace information too	
<code>gams/miptrace/nodefreq</code> ($0 \leq \text{integer}$)	100
frequency in number of nodes when to write branch-and-bound trace information, 0 to disable	
<code>gams/miptrace/timefreq</code> ($0 \leq \text{real}$)	5
frequency in seconds when to write branch-and-bound trace information, 0.0 to disable	

`gams/printstatistics` (boolean) FALSE
whether to print statistics on a MIP solve

`gams/resolvenlp` (boolean) TRUE
whether to resolve MINLP with fixed discrete variables if best solution violates some constraints

Branching

`branching/allfullstrong/maxbounddist` ($0 \leq \text{real} \leq 1$) 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/allfullstrong/maxdepth` ($-1 \leq \text{integer}$) -1
maximal depth level, up to which branching rule <allfullstrong> should be used (-1 for no limit)

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

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

`branching/delaypscostupdate` (boolean) TRUE
should updating pseudo costs for continuous variables be delayed to the time after separation?

`branching/fullstrong/maxbounddist` ($0 \leq \text{real} \leq 1$) 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/maxdepth` ($-1 \leq \text{integer}$) -1
maximal depth level, up to which branching rule <fullstrong> should be used (-1 for no limit)

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

`branching/inference/maxbounddist` ($0 \leq \text{real} \leq 1$) 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/maxdepth` ($-1 \leq \text{integer}$) -1
maximal depth level, up to which branching rule <inference> should be used (-1 for no limit)

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

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

`branching/leastinf/maxbounddist` ($0 \leq \text{real} \leq 1$) 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/maxdepth` ($-1 \leq \text{integer}$) -1
maximal depth level, up to which branching rule <leastinf> should be used (-1 for no limit)

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

`branching/lpgainnormalize` (character) s
strategy for normalization of LP gain when updating pseudocosts of continuous variables (divide by movement of 'lp value, reduction in 'd'omain width, or reduction in domain width of 's'ibling)

`branching/mostinf/maxbounddist` ($0 \leq \text{real} \leq 1$) 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/mostinf/maxdepth` ($-1 \leq \text{integer}$) -1
maximal depth level, up to which branching rule <mostinf> should be used (-1 for no limit)

<code>branching/mostinf/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	100
priority of branching rule <mostinf>	
<code>branching/preferbinary</code> (boolean)	FALSE
should branching on binary variables be preferred?	
<code>branching/pscost/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	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)	
<code>branching/pscost/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level, up to which branching rule <pscost> should be used (-1 for no limit)	
<code>branching/pscost/narymaxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth where to do n-ary branching, -1 to turn off	
<code>branching/pscost/naryminwidth</code> ($0 \leq \text{real} \leq 1$)	0.001
minimal domain width in children when doing n-ary branching, relative to global bounds	
<code>branching/pscost/narywidthfactor</code> ($1 \leq \text{real}$)	2
factor of domain width in n-ary branching when creating nodes with increasing distance from branching value	
<code>branching/pscost/nchildren</code> ($2 \leq \text{integer}$)	2
number of children to create in n-ary branching	
<code>branching/pscost/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	2000
priority of branching rule <pscost>	
<code>branching/pscost/strategy</code> (character)	u
strategy for utilizing pseudo-costs of external branching candidates (multiply as in pseudo costs 'u'pdate rule, or by 'd'omain reduction, or by domain reduction of 's'ibling, or by 'v'ariable score)	
<code>branching/random/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	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)	
<code>branching/random/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level, up to which branching rule <random> should be used (-1 for no limit)	
<code>branching/random/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-100000
priority of branching rule <random>	
<code>branching/random/seed</code> ($0 \leq \text{integer}$)	0
initial random seed value	
<code>branching/relpscost/initcand</code> ($0 \leq \text{integer}$)	100
maximal number of candidates initialized with strong branching per node	
<code>branching/relpscost/inititer</code> ($0 \leq \text{integer}$)	0
iteration limit for strong branching initializations of pseudo cost entries (0: auto)	
<code>branching/relpscost/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	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)	
<code>branching/relpscost/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level, up to which branching rule <relpscost> should be used (-1 for no limit)	
<code>branching/relpscost/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	10000
priority of branching rule <relpscost>	
<code>branching/relpscost/sbiterofs</code> ($0 \leq \text{integer}$)	100000
additional number of allowed strong branching LP iterations	
<code>branching/relpscost/sbiterquot</code> ($0 \leq \text{real}$)	0.5
maximal fraction of strong branching LP iterations compared to node relaxation LP iterations	

Branching (advanced options)

<code>branching/fullstrong/reevalage</code> ($0 \leq \text{integer}$)	10
number of intermediate LPs solved to trigger reevaluation of strong branching value for a variable that was already evaluated at the current node	
<code>branching/inference/conflictweight</code> ($0 \leq \text{real}$)	1000
weight in score calculations for conflict score	
<code>branching/inference/cutoffweight</code> ($0 \leq \text{real}$)	1
weight in score calculations for cutoff score	
<code>branching/inference/fractionals</code> (boolean)	TRUE
should branching on LP solution be restricted to the fractional variables?	
<code>branching/inference/inferenceweight</code> (real)	1
weight in score calculations for inference score	
<code>branching/pscost/maxscoreweight</code> (real)	1.3
weight for maximum of scores of a branching candidate when building weighted sum of min/max/sum of scores	
<code>branching/pscost/minscoreweight</code> (real)	0.8
weight for minimum of scores of a branching candidate when building weighted sum of min/max/sum of scores	
<code>branching/pscost/sumscoreweight</code> (real)	0.1
weight for sum of scores of a branching candidate when building weighted sum of min/max/sum of scores	
<code>branching/relpscost/conflictlengthweight</code> (real)	0
weight in score calculations for conflict length score	
<code>branching/relpscost/conflictweight</code> (real)	0.01
weight in score calculations for conflict score	
<code>branching/relpscost/cutoffweight</code> (real)	0.0001
weight in score calculations for cutoff score	
<code>branching/relpscost/inferenceweight</code> (real)	0.0001
weight in score calculations for inference score	
<code>branching/relpscost/maxbdchgs</code> ($-1 \leq \text{integer}$)	5
maximal number of bound tightenings before the node is reevaluated (-1: unlimited)	
<code>branching/relpscost/maxlookahead</code> ($1 \leq \text{integer}$)	8
maximal number of further variables evaluated without better score	
<code>branching/relpscost/maxreliable</code> ($0 \leq \text{real}$)	8
maximal value for minimum pseudo cost size to regard pseudo cost value as reliable	
<code>branching/relpscost/minreliable</code> ($0 \leq \text{real}$)	1
minimal value for minimum pseudo cost size to regard pseudo cost value as reliable	
<code>branching/relpscost/pscostweight</code> (real)	1
weight in score calculations for pseudo cost score	
<code>branching/scorefac</code> ($0 \leq \text{real} \leq 1$)	0.167
branching score factor to weigh downward and upward gain prediction in sum score function	
<code>branching/scorefunc</code> (character)	p
branching score function ('s'um, 'p'roduct)	
Conflict analysis	
<code>conflict/bounddisjunction/continuousfrac</code> ($0 \leq \text{real} \leq 1$)	0.4
maximal percentage of continuous variables within a conflict	
<code>conflict/enable</code> (boolean)	TRUE
should conflict analysis be enabled?	
<code>conflict/preferbinary</code> (boolean)	FALSE
should binary conflicts be preferred?	

<code>conflict/restartfac</code> ($0 \leq \text{real}$)	1.5
factor to increase restartnum with after each restart	
<code>conflict/restartnum</code> ($0 \leq \text{integer}$)	0
number of successful conflict analysis calls that trigger a restart (0: disable conflict restarts)	
<code>conflict/useboundlp</code> (<code>boolean</code>)	FALSE
should bound exceeding LP conflict analysis be used?	
<code>conflict/useinflp</code> (<code>boolean</code>)	TRUE
should infeasible LP conflict analysis be used?	
<code>conflict/useprop</code> (<code>boolean</code>)	TRUE
should propagation conflict analysis be used?	
<code>conflict/usepseudo</code> (<code>boolean</code>)	TRUE
should pseudo solution conflict analysis be used?	
<code>conflict/usesb</code> (<code>boolean</code>)	FALSE
should infeasible/bound exceeding strong branching conflict analysis be used?	
Conflict analysis (advanced options)	
<code>conflict/allowlocal</code> (<code>boolean</code>)	TRUE
should conflict constraints be generated that are only valid locally?	
<code>conflict/bounddisjunction/priority</code> (<code>integer</code>)	-3000000
priority of conflict handler <bounddisjunction>	
<code>conflict/depthscorefac</code> (<code>real</code>)	1
score factor for depth level in bound relaxation heuristic of LP analysis	
<code>conflict/dynamic</code> (<code>boolean</code>)	TRUE
should the conflict constraints be subject to aging?	
<code>conflict/fuiplevels</code> ($-1 \leq \text{integer}$)	-1
number of depth levels up to which first UIP's are used in conflict analysis (-1: use All-FirstUIP rule)	
<code>conflict/indicatorconflict/priority</code> (<code>integer</code>)	200000
priority of conflict handler <indicatorconflict>	
<code>conflict/interconss</code> ($-1 \leq \text{integer}$)	-1
maximal number of intermediate conflict constraints generated in conflict graph (-1: use every intermediate constraint)	
<code>conflict/keepreprop</code> (<code>boolean</code>)	TRUE
should constraints be kept for repropagation even if they are too long?	
<code>conflict/linear/priority</code> (<code>integer</code>)	-1000000
priority of conflict handler <linear>	
<code>conflict/logicor/priority</code> (<code>integer</code>)	800000
priority of conflict handler <logicor>	
<code>conflict/lpiterations</code> ($-1 \leq \text{integer}$)	10
maximal number of LP iterations in each LP resolving loop (-1: no limit)	
<code>conflict/maxconss</code> ($-1 \leq \text{integer}$)	10
maximal number of conflict constraints accepted at an infeasible node (-1: use all generated conflict constraints)	
<code>conflict/maxlploops</code> ($-1 \leq \text{integer}$)	2
maximal number of LP resolving loops during conflict analysis (-1: no limit)	
<code>conflict/maxvarsfac</code> ($0 \leq \text{real}$)	0.1
maximal fraction of variables involved in a conflict constraint	
<code>conflict/minmaxvars</code> ($0 \leq \text{integer}$)	30
minimal absolute maximum of variables involved in a conflict constraint	

<code>conflict/reconvlevels</code> ($-1 \leq \text{integer}$)	-1
number of depth levels up to which UIP reconvergence constraints are generated (-1: generate reconvergence constraints in all depth levels)	
<code>conflict/removable</code> (boolean)	TRUE
should the conflict's relaxations be subject to LP aging and cleanup?	
<code>conflict/repropagate</code> (boolean)	TRUE
should earlier nodes be repropagated in order to replace branching decisions by deductions?	
<code>conflict/scorefac</code> ($10^{-6} \leq \text{real} \leq 1$)	0.98
factor to decrease importance of variables' earlier conflict scores	
<code>conflict/separate</code> (boolean)	TRUE
should the conflict constraints be separated?	
<code>conflict/setppc/priority</code> (integer)	700000
priority of conflict handler <setppc>	
<code>conflict/settlelocal</code> (boolean)	FALSE
should conflict constraints be attached only to the local subtree where they can be useful?	

Constraints

<code>constraints/abspower/addvarbounds</code> (boolean)	TRUE
should variable bounds be added to the cutpool?	
<code>constraints/abspower/branchminconverror</code> (boolean)	FALSE
whether to compute branching point such that the convexification error is minimized (after branching on 0.0)	
<code>constraints/abspower/cutmaxrange</code> ($0 \leq \text{real}$)	10^7
maximal coef range of a cut (maximal coefficient divided by minimal coefficient) in order to be added to LP relaxation	
<code>constraints/abspower/linfeasshift</code> (boolean)	TRUE
whether to try to make solutions in check function feasible by shifting the linear variable z	
<code>constraints/abspower/minefficacyenfofac</code> ($1 \leq \text{real}$)	2
minimal target efficacy of a cut in order to add it to relaxation during enforcement as factor of feasibility tolerance (may be ignored)	
<code>constraints/abspower/minefficacysepa</code> ($0 \leq \text{real}$)	0.0001
minimal efficacy for a cut to be added to the LP during separation; overwrites separating/efficacy	
<code>constraints/abspower/preferzerobranch</code> ($0 \leq \text{integer} \leq 3$)	1
how much to prefer branching on 0.0 when sign of variable is not fixed yet: 0 no preference, 1 prefer if LP solution will be cutoff in both child nodes, 2 prefer always, 3 ensure always	
<code>constraints/abspower/projectrefpoint</code> (boolean)	TRUE
whether to project the reference point when linearizing an absolute power constraint in a convex region	
<code>constraints/abspower/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/abspower/sepafreq</code> ($-1 \leq \text{integer}$)	1
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/abspower/sepainboundsonly</code> (boolean)	FALSE
whether to separate linearization cuts only in the variable bounds (does not affect enforcement)	
<code>constraints/abspower/sepanlpmincont</code> ($0 \leq \text{real} \leq 2$)	1
minimal required fraction of continuous variables in problem to use solution of NLP relaxation in root for separation	
<code>constraints/and/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/and/sepafreq</code> ($-1 \leq \text{integer}$)	1

frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/bivariate/linfeasshift</code> (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)	
<code>constraints/bivariate/maxpropounds</code> ($0 \leq$ integer)	1
limit on number of propagation rounds for a single constraint within one round of SCIP propagation	
<code>constraints/bivariate/minefficacyenfo</code> ($0 \leq$ real)	$2 \cdot 10^{-6}$
minimal target efficacy of a cut in order to add it to relaxation during enforcement (may be ignored)	
<code>constraints/bivariate/minefficacysepa</code> ($0 \leq$ real)	0.0001
minimal efficacy for a cut to be added to the LP during separation; overwrites separating/efficacy	
<code>constraints/bivariate/ninitlprefpoints</code> ($0 \leq$ integer)	3
number of reference points in each direction where to compute linear support for envelope in LP initialization	
<code>constraints/bivariate/propfreq</code> ($-1 \leq$ integer)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/bivariate/sepafreq</code> ($-1 \leq$ integer)	1
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/bounddisjunction/propfreq</code> ($-1 \leq$ integer)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/bounddisjunction/sepafreq</code> ($-1 \leq$ integer)	-1
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/indicator/propfreq</code> ($-1 \leq$ integer)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/indicator/sepafreq</code> ($-1 \leq$ integer)	10
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/integral/propfreq</code> ($-1 \leq$ integer)	-1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/integral/sepafreq</code> ($-1 \leq$ integer)	-1
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/knapsack/maxrounds</code> ($-1 \leq$ integer)	5
maximal number of separation rounds per node (-1: unlimited)	
<code>constraints/knapsack/maxroundsroot</code> ($-1 \leq$ integer)	-1
maximal number of separation rounds per node in the root node (-1: unlimited)	
<code>constraints/knapsack/maxsepacuts</code> ($0 \leq$ integer)	50
maximal number of cuts separated per separation round	
<code>constraints/knapsack/maxsepacutsroot</code> ($0 \leq$ integer)	200
maximal number of cuts separated per separation round in the root node	
<code>constraints/knapsack/propfreq</code> ($-1 \leq$ integer)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/knapsack/sepafreq</code> ($-1 \leq$ integer)	0
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/linear/maxrounds</code> ($-1 \leq$ integer)	5
maximal number of separation rounds per node (-1: unlimited)	
<code>constraints/linear/maxroundsroot</code> ($-1 \leq$ integer)	-1
maximal number of separation rounds per node in the root node (-1: unlimited)	
<code>constraints/linear/maxsepacuts</code> ($0 \leq$ integer)	50
maximal number of cuts separated per separation round	

<code>constraints/linear/maxsepacutsroot</code> ($0 \leq \text{integer}$)	200
maximal number of cuts separated per separation round in the root node	
<code>constraints/linear/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/linear/sepafreq</code> ($-1 \leq \text{integer}$)	0
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/linear/separateall</code> (boolean)	FALSE
should all constraints be subject to cardinality cut generation instead of only the ones with non-zero dual value?	
<code>constraints/linear/upgrade/knapsack</code> (boolean)	TRUE
enable linear upgrading for constraint handler <knapsack>	
<code>constraints/linear/upgrade/logicor</code> (boolean)	TRUE
enable linear upgrading for constraint handler <logicor>	
<code>constraints/linear/upgrade/setppc</code> (boolean)	TRUE
enable linear upgrading for constraint handler <setppc>	
<code>constraints/linear/upgrade/varbound</code> (boolean)	TRUE
enable linear upgrading for constraint handler <varbound>	
<code>constraints/logicor/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/logicor/sepafreq</code> ($-1 \leq \text{integer}$)	0
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/nonlinear/cutmaxrange</code> ($0 \leq \text{real}$)	10^7
maximal coef range of a cut (maximal coefficient divided by minimal coefficient) in order to be added to LP relaxation	
<code>constraints/nonlinear/linfeasshift</code> (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)	
<code>constraints/nonlinear/maxpropounds</code> ($0 \leq \text{integer}$)	1
limit on number of propagation rounds for a single constraint within one round of SCIP propagation	
<code>constraints/nonlinear/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/nonlinear/reformulate</code> (boolean)	TRUE
whether to reformulate expression graph	
<code>constraints/nonlinear/sepafreq</code> ($-1 \leq \text{integer}$)	1
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/nonlinear/sepanlpmincont</code> ($0 \leq \text{real} \leq 2$)	1
minimal required fraction of continuous variables in problem to use solution of NLP relaxation in root for separation	
<code>constraints/nonlinear/upgrade/abspower</code> (boolean)	TRUE
enable nonlinear upgrading for constraint handler <abspower>	
<code>constraints/nonlinear/upgrade/and</code> (boolean)	TRUE
enable nonlinear upgrading for constraint handler <and>	
<code>constraints/nonlinear/upgrade/bivariate</code> (boolean)	FALSE
enable nonlinear upgrading for constraint handler <bivariate>	
<code>constraints/nonlinear/upgrade/linear</code> (boolean)	TRUE
enable nonlinear upgrading for constraint handler <linear>	
<code>constraints/nonlinear/upgrade/quadratic</code> (boolean)	TRUE
enable nonlinear upgrading for constraint handler <quadratic>	

<code>constraints/quadratic/checkcurvature</code> (boolean)	TRUE
whether multivariate quadratic functions should be checked for convexity/concavity	
<code>constraints/quadratic/empathy4and</code> ($0 \leq \text{integer} \leq 2$)	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	
<code>constraints/quadratic/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/quadratic/replacebinaryprod</code> ($0 \leq \text{integer}$)	∞
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)	
<code>constraints/quadratic/sepaftereq</code> ($-1 \leq \text{integer}$)	1
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/quadratic/sepanlpmincont</code> ($0 \leq \text{real} \leq 2$)	1
minimal required fraction of continuous variables in problem to use solution of NLP relaxation in root for separation	
<code>constraints/quadratic/upgrade/abspower</code> (boolean)	TRUE
enable quadratic upgrading for constraint handler <abspower>	
<code>constraints/quadratic/upgrade/bivariate</code> (boolean)	FALSE
enable quadratic upgrading for constraint handler <bivariate>	
<code>constraints/quadratic/upgrade/bounddisjunction</code> (boolean)	TRUE
enable quadratic upgrading for constraint handler <bounddisjunction>	
<code>constraints/quadratic/upgrade/linear</code> (boolean)	TRUE
enable quadratic upgrading for constraint handler <linear>	
<code>constraints/quadratic/upgrade/soc</code> (boolean)	TRUE
enable quadratic upgrading for constraint handler <soc>	
<code>constraints/setppc/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/setppc/sepaftereq</code> ($-1 \leq \text{integer}$)	0
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/soc/glineur</code> (boolean)	TRUE
whether the Glineur Outer Approximation should be used instead of Ben-Tal Nemirovski	
<code>constraints/soc/linfeasshift</code> (boolean)	TRUE
whether to try to make solutions feasible in check by shifting the variable on the right hand side	
<code>constraints/soc/minefficacy</code> ($0 \leq \text{real}$)	0.0001
minimal efficacy of a cut to be added to LP in separation	
<code>constraints/soc/nauxvars</code> ($0 \leq \text{integer}$)	0
number of auxiliary variables to use when creating a linear outer approx. of a SOC3 constraint; 0 to turn off	
<code>constraints/soc/nlpform</code> (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, d: convex division form)	
<code>constraints/soc/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/soc/sepaftereq</code> ($-1 \leq \text{integer}$)	1
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/soc/sepanlpmincont</code> ($0 \leq \text{real} \leq 2$)	1
minimal required fraction of continuous variables in problem to use solution of NLP relaxation in root for separation	

<code>constraints/SOS1/branchnonzeros</code> (boolean)	FALSE
Branch on SOS constraint with most number of nonzeros?	
<code>constraints/SOS1/branchsos</code> (boolean)	TRUE
Use SOS1 branching in enforcing (otherwise leave decision to branching rules)?	
<code>constraints/SOS1/branchweight</code> (boolean)	FALSE
Branch on SOS cons. with highest nonzero-variable weight for branching (needs <code>branchnonzeros = false</code>)?	
<code>constraints/SOS1/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/SOS1/sepafreq</code> ($-1 \leq \text{integer}$)	0
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/SOS2/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/SOS2/sepafreq</code> ($-1 \leq \text{integer}$)	0
frequency for separating cuts (-1: never, 0: only in root node)	
<code>constraints/varbound/propfreq</code> ($-1 \leq \text{integer}$)	1
frequency for propagating domains (-1: never, 0: only in root node)	
<code>constraints/varbound/sepafreq</code> ($-1 \leq \text{integer}$)	0
frequency for separating cuts (-1: never, 0: only in root node)	
Constraints (advanced options)	
<code>constraints/abspower/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/abspower/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/abspower/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/abspower/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/abspower/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/agelimit</code> ($-1 \leq \text{integer}$)	0
maximum age an unnecessary constraint can reach before it is deleted (0: dynamic, -1: keep all constraints)	
<code>constraints/and/aggrlinearization</code> (boolean)	FALSE
should an aggregated linearization be used?	
<code>constraints/and/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/and/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/and/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/and/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/and/enforcecuts</code> (boolean)	TRUE
should cuts be separated during LP enforcing?	
<code>constraints/and/linearize</code> (boolean)	FALSE
should the "and" constraint get linearized and removed (in presolving)?	

<code>constraints/and/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/and/presolpairwise</code> (boolean)	TRUE
should pairwise constraint comparison be performed in presolving?	
<code>constraints/and/presolusehashing</code> (boolean)	TRUE
should hash table be used for detecting redundant constraints in advance	
<code>constraints/bivariate/cutmaxrange</code> ($0 \leq \text{real}$)	10^7
maximal coef range of a cut (maximal coefficient divided by minimal coefficient) in order to be added to LP relaxation	
<code>constraints/bivariate/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/bivariate/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/bivariate/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/bivariate/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/bivariate/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/bounddisjunction/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/bounddisjunction/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/bounddisjunction/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/bounddisjunction/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/bounddisjunction/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/disableenfops</code> (boolean)	FALSE
should enforcement of pseudo solution be disabled?	
<code>constraints/indicator/addcoupling</code> (boolean)	TRUE
add initial coupling inequalities	
<code>constraints/indicator/addcouplingcons</code> (boolean)	FALSE
add initial coupling inequalities as linear constraints, if 'addcoupling' is true	
<code>constraints/indicator/branchindicators</code> (boolean)	FALSE
Branch on indicator constraints in enforcing?	
<code>constraints/indicator/conflictsupgrade</code> (boolean)	FALSE
Try to upgrade bounddisjunction conflicts by replacing slack variables?	
<code>constraints/indicator/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/indicator/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/indicator/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	

<code>constraints/indicator/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/indicator/enforcecuts</code> (boolean)	FALSE
in enforcing try to generate cuts (only if <code>sepaalternativelp</code> is true)	
<code>constraints/indicator/forcerestart</code> (boolean)	FALSE
force restart if we have a max FS instance and gap is 1?	
<code>constraints/indicator/generatebilinear</code> (boolean)	FALSE
do not generate indicator constraint, but a bilinear constraint instead	
<code>constraints/indicator/genlogicor</code> (boolean)	FALSE
Generate logicor constraints instead of cuts?	
<code>constraints/indicator/maxconditionaltlp</code> ($0 \leq \text{real}$)	0
maximum estimated condition of the solution basis matrix of the alternative LP to be trustworthy (0.0 to disable check)	
<code>constraints/indicator/maxcouplingvalue</code> ($0 \leq \text{real} \leq 10^9$)	10000
maximum coefficient for binary variable in coupling constraint	
<code>constraints/indicator/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/indicator/nolinconscont</code> (boolean)	FALSE
decompose problem - do not generate linear constraint if all variables are continuous	
<code>constraints/indicator/removeindicators</code> (boolean)	FALSE
remove indicator constraint if corresponding variable bound constraint has been added?	
<code>constraints/indicator/restartfrac</code> ($0 \leq \text{real} \leq 1$)	0.9
fraction of binary variables that need to be fixed before restart occurs (in <code>forcerestart</code>)	
<code>constraints/indicator/sepaalternativelp</code> (boolean)	FALSE
Separate using the alternative LP?	
<code>constraints/indicator/trysolutions</code> (boolean)	TRUE
Try to make solutions feasible by setting indicator variables?	
<code>constraints/indicator/updatebounds</code> (boolean)	FALSE
Update bounds of original variables for separation?	
<code>constraints/integral/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/integral/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/integral/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/integral/eagerfreq</code> ($-1 \leq \text{integer}$)	-1
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/integral/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/knapsack/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/knapsack/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/knapsack/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	

<code>constraints/knapsack/disaggregation</code> (boolean)	TRUE
should disaggregation of knapsack constraints be allowed in preprocessing?	
<code>constraints/knapsack/dualpresolving</code> (boolean)	TRUE
should dual presolving steps be performed?	
<code>constraints/knapsack/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/knapsack/maxcardbounddist</code> ($0 \leq \text{real} \leq 1$)	0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for separating knapsack cuts	
<code>constraints/knapsack/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/knapsack/negatedclique</code> (boolean)	TRUE
should negated clique information be used in solving process	
<code>constraints/knapsack/presolpairwise</code> (boolean)	TRUE
should pairwise constraint comparison be performed in presolving?	
<code>constraints/knapsack/presolusehashing</code> (boolean)	TRUE
should hash table be used for detecting redundant constraints in advance	
<code>constraints/knapsack/sepacardfreq</code> ($-1 \leq \text{integer}$)	1
multiplier on separation frequency, how often knapsack cuts are separated (-1: never, 0: only at root)	
<code>constraints/knapsack/simplifyinequalities</code> (boolean)	FALSE
should presolving try to simplify knapsacks	
<code>constraints/linear/aggregatevariables</code> (boolean)	TRUE
should presolving search for aggregations in equations	
<code>constraints/linear/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/linear/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/linear/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/linear/dualpresolving</code> (boolean)	TRUE
should dual presolving steps be performed?	
<code>constraints/linear/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/linear/maxaggrnormscale</code> ($0 \leq \text{real}$)	0
maximal allowed relative gain in maximum norm for constraint aggregation (0.0: disable constraint aggregation)	
<code>constraints/linear/maxcardbounddist</code> ($0 \leq \text{real} \leq 1$)	0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for separating knapsack cardinality cuts	
<code>constraints/linear/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/linear/mingainpernmincomparisons</code> ($0 \leq \text{real}$)	10^{-6}
minimal gain per minimal pairwise presolve comparisons to repeat pairwise comparison round	
<code>constraints/linear/nmincomparisons</code> ($1 \leq \text{integer}$)	200000
number for minimal pairwise presolve comparisons	
<code>constraints/linear/presolpairwise</code> (boolean)	TRUE

should pairwise constraint comparison be performed in presolving?	
<code>constraints/linear/presolusehashing</code> (boolean)	TRUE
should hash table be used for detecting redundant constraints in advance	
<code>constraints/linear/simplifyinequalities</code> (boolean)	FALSE
should presolving try to simplify inequalities	
<code>constraints/linear/sortvars</code> (boolean)	TRUE
apply binaries sorting in decr. order of coeff abs value?	
<code>constraints/linear/tightenboundsfreq</code> ($-1 \leq$ integer)	1
multiplier on propagation frequency, how often the bounds are tightened (-1: never, 0: only at root)	
<code>constraints/logicor/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/logicor/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/logicor/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/logicor/dualpresolving</code> (boolean)	TRUE
should dual presolving steps be performed?	
<code>constraints/logicor/eagerfreq</code> ($-1 \leq$ integer)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/logicor/maxprerounds</code> ($-1 \leq$ integer)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/logicor/negatedclique</code> (boolean)	FALSE
should negated clique information be used in presolving	
<code>constraints/logicor/presolpairwise</code> (boolean)	TRUE
should pairwise constraint comparison be performed in presolving?	
<code>constraints/logicor/presolusehashing</code> (boolean)	TRUE
should hash table be used for detecting redundant constraints in advance	
<code>constraints/nonlinear/assumeconvex</code> (boolean)	FALSE
whether to assume that nonlinear functions in inequalities (\leq) are convex (disables reformulation)	
<code>constraints/nonlinear/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/nonlinear/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/nonlinear/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/nonlinear/eagerfreq</code> ($-1 \leq$ integer)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/nonlinear/maxexpansionexponent</code> ($1 \leq$ integer)	2
maximal exponent where still expanding non-monomial polynomials in expression simplification	
<code>constraints/nonlinear/maxprerounds</code> ($-1 \leq$ integer)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/nonlinear/minefficacyenfofac</code> ($1 \leq$ real)	2
minimal target efficacy of a cut in order to add it to relaxation during enforcement as a factor of the feasibility tolerance (may be ignored)	
<code>constraints/nonlinear/minefficacysepa</code> ($0 \leq$ real)	0.0001

minimal efficacy for a cut to be added to the LP during separation; overwrites separating/efficacy	
<code>constraints/obsoleteage</code> ($-1 \leq \text{integer}$)	-1
age of a constraint after which it is marked obsolete (0: dynamic, -1 do not mark constraints obsolete)	
<code>constraints/quadratic/binreforminitial</code> (boolean)	FALSE
whether to make non-varbound linear constraints added due to replacing products with binary variables initial	
<code>constraints/quadratic/checkfactorable</code> (boolean)	TRUE
whether constraint functions should be checked to be factorable	
<code>constraints/quadratic/cutmaxrange</code> ($0 \leq \text{real}$)	10^7
maximal coef range of a cut (maximal coefficient divided by minimal coefficient) in order to be added to LP relaxation	
<code>constraints/quadratic/defaultbound</code> ($0 \leq \text{real}$)	∞
a default bound to impose on unbounded variables in quadratic terms (-defaultbound is used for missing lower bounds)	
<code>constraints/quadratic/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/quadratic/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/quadratic/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/quadratic/disaggregate</code> (boolean)	FALSE
whether to disaggregate quadratic parts that decompose into a sum of non-overlapping quadratic terms	
<code>constraints/quadratic/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/quadratic/linearizeheursol</code> (boolean)	TRUE
whether linearizations of convex quadratic constraints should be added to cutpool in a solution found by some heuristic	
<code>constraints/quadratic/linfeasshift</code> (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)	
<code>constraints/quadratic/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/quadratic/maxpropounds</code> ($0 \leq \text{integer}$)	1
limit on number of propagation rounds for a single constraint within one round of SCIP propagation during solve	
<code>constraints/quadratic/maxpropoundspresolve</code> ($0 \leq \text{integer}$)	10
limit on number of propagation rounds for a single constraint within one round of SCIP presolve	
<code>constraints/quadratic/minefficacyenfofac</code> ($1 \leq \text{real}$)	2
minimal target efficacy of a cut in order to add it to relaxation during enforcement as a factor of the feasibility tolerance (may be ignored)	
<code>constraints/quadratic/minefficacysepa</code> ($0 \leq \text{real}$)	0.0001
minimal efficacy for a cut to be added to the LP during separation; overwrites separating/efficacy	
<code>constraints/quadratic/scaling</code> (boolean)	TRUE
whether a quadratic constraint should be scaled w.r.t. the current gradient norm when checking for feasibility	
<code>constraints/setppc/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/setppc/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	

<code>constraints/setppc/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/setppc/dualpresolving</code> (boolean)	TRUE
should dual presolving steps be performed?	
<code>constraints/setppc/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/setppc/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/setppc/npseudobranches</code> ($0 \leq \text{integer}$)	2
number of children created in pseudo branching (0: disable pseudo branching)	
<code>constraints/setppc/presolpairwise</code> (boolean)	TRUE
should pairwise constraint comparison be performed in presolving?	
<code>constraints/setppc/presolusehashing</code> (boolean)	TRUE
should hash table be used for detecting redundant constraints in advance	
<code>constraints/soc/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/soc/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/soc/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/soc/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/soc/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/soc/projectpoint</code> (boolean)	FALSE
whether the reference point of a cut should be projected onto the feasible set of the SOC constraint	
<code>constraints/soc/scaling</code> (boolean)	TRUE
whether a constraint should be scaled w.r.t. the current gradient norm when checking for feasibility	
<code>constraints/soc/sparsify</code> (boolean)	FALSE
whether to sparsify cuts	
<code>constraints/soc/sparsifymaxloss</code> ($0 \leq \text{real} \leq 1$)	0.2
maximal loss in cut efficacy by sparsification	
<code>constraints/soc/sparsifynzgrowth</code> ($1 \leq \text{real}$)	1.3
growth rate of maximal allowed nonzeros in cuts in sparsification	
<code>constraints/SOS1/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/SOS1/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/SOS1/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/SOS1/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/SOS1/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	

<code>constraints/SOS2/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/SOS2/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/SOS2/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/SOS2/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/SOS2/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/varbound/delaypresol</code> (boolean)	FALSE
should presolving method be delayed, if other presolvers found reductions?	
<code>constraints/varbound/delayprop</code> (boolean)	FALSE
should propagation method be delayed, if other propagators found reductions?	
<code>constraints/varbound/delaysepa</code> (boolean)	FALSE
should separation method be delayed, if other separators found cuts?	
<code>constraints/varbound/eagerfreq</code> ($-1 \leq \text{integer}$)	100
frequency for using all instead of only the useful constraints in separation, propagation and enforcement (-1: never, 0: only in first evaluation)	
<code>constraints/varbound/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the constraint handler participates in (-1: no limit)	
<code>constraints/varbound/presolpairwise</code> (boolean)	TRUE
should pairwise constraint comparison be performed in presolving?	

Output

<code>display/avgdualbound/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <avgdualbound> (0: off, 1: auto, 2:on)	
<code>display/conflicts/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <conflicts> (0: off, 1: auto, 2:on)	
<code>display/conss/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <conss> (0: off, 1: auto, 2:on)	
<code>display/curcols/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <curcols> (0: off, 1: auto, 2:on)	
<code>display/curconss/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <curconss> (0: off, 1: auto, 2:on)	
<code>display/curdualbound/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <curdualbound> (0: off, 1: auto, 2:on)	
<code>display/currows/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <currows> (0: off, 1: auto, 2:on)	
<code>display/cutoffbound/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <cutoffbound> (0: off, 1: auto, 2:on)	
<code>display/cuts/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <cuts> (0: off, 1: auto, 2:on)	
<code>display/depth/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <depth> (0: off, 1: auto, 2:on)	
<code>display/dualbound/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <dualbound> (0: off, 1: auto, 2:on)	

<code>display/estimate/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <estimate> (0: off, 1: auto, 2:on)	
<code>display/feasST/active</code> ($0 \leq \text{integer} \leq 2$)	0
display activation status of display column <feasST> (0: off, 1: auto, 2:on)	
<code>display/freq</code> ($-1 \leq \text{integer}$)	100
frequency for displaying node information lines	
<code>display/gap/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <gap> (0: off, 1: auto, 2:on)	
<code>display/headerfreq</code> ($-1 \leq \text{integer}$)	15
frequency for displaying header lines (every n'th node information line)	
<code>display/lpavgiterations/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <lpavgiterations> (0: off, 1: auto, 2:on)	
<code>display/lpcond/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <lpcond> (0: off, 1: auto, 2:on)	
<code>display/lpinfo</code> (boolean)	FALSE
should the LP solver display status messages?	
<code>display/lpiterations/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <lpiterations> (0: off, 1: auto, 2:on)	
<code>display/lpobj/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <lpobj> (0: off, 1: auto, 2:on)	
<code>display/maxdepth/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <maxdepth> (0: off, 1: auto, 2:on)	
<code>display/memused/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <memused> (0: off, 1: auto, 2:on)	
<code>display/nextnbranchcands/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <nextnbranchcands> (0: off, 1: auto, 2:on)	
<code>display/nfrac/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <nfrac> (0: off, 1: auto, 2:on)	
<code>display/nnodes/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <nnodes> (0: off, 1: auto, 2:on)	
<code>display/nodesleft/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <nodesleft> (0: off, 1: auto, 2:on)	
<code>display/nsols/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <nsols> (0: off, 1: auto, 2:on)	
<code>display/plungedepth/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <plungedepth> (0: off, 1: auto, 2:on)	
<code>display/poolsize/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <poolsize> (0: off, 1: auto, 2:on)	
<code>display/primalbound/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <primalbound> (0: off, 1: auto, 2:on)	
<code>display/primalgap/active</code> ($0 \leq \text{integer} \leq 2$)	0
display activation status of display column <primalgap> (0: off, 1: auto, 2:on)	
<code>display/separounds/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <separounds> (0: off, 1: auto, 2:on)	
<code>display/solfound/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <solfound> (0: off, 1: auto, 2:on)	

<code>display/sols/active</code> ($0 \leq \text{integer} \leq 2$)	0
display activation status of display column <code><sols></code> (0: off, 1: auto, 2: on)	
<code>display/strongbranches/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <code><strongbranches></code> (0: off, 1: auto, 2: on)	
<code>display/time/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <code><time></code> (0: off, 1: auto, 2: on)	
<code>display/vars/active</code> ($0 \leq \text{integer} \leq 2$)	1
display activation status of display column <code><vars></code> (0: off, 1: auto, 2: on)	
<code>display/verblevel</code> ($0 \leq \text{integer} \leq 5$)	4
verbosity level of output	
<code>display/width</code> ($0 \leq \text{integer}$)	139
maximal number of characters in a node information line	
Heuristics	
<code>heuristics/actconsdiving/backtrack</code> (boolean)	TRUE
use one level of backtracking if infeasibility is encountered?	
<code>heuristics/actconsdiving/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling primal heuristic <code><actconsdiving></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/actconsdiving/freqofs</code> ($0 \leq \text{integer}$)	5
frequency offset for calling primal heuristic <code><actconsdiving></code>	
<code>heuristics/actconsdiving/maxlpiterofs</code> ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
<code>heuristics/actconsdiving/maxlpiterquot</code> ($0 \leq \text{real}$)	0.05
maximal fraction of diving LP iterations compared to node LP iterations	
<code>heuristics/cliquote/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling primal heuristic <code><cliquote></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/cliquote/freqofs</code> ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <code><cliquote></code>	
<code>heuristics/cliquote/minfixingrate</code> ($0 \leq \text{real} \leq 1$)	0.5
minimum percentage of integer variables that have to be fixable	
<code>heuristics/cliquote/nodesofs</code> ($0 \leq \text{integer}$)	500
number of nodes added to the contingent of the total nodes	
<code>heuristics/cliquote/nodesquot</code> ($0 \leq \text{real} \leq 1$)	0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem	
<code>heuristics/coefdiving/backtrack</code> (boolean)	TRUE
use one level of backtracking if infeasibility is encountered?	
<code>heuristics/coefdiving/freq</code> ($-1 \leq \text{integer}$)	10
frequency for calling primal heuristic <code><coefdiving></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/coefdiving/freqofs</code> ($0 \leq \text{integer}$)	1
frequency offset for calling primal heuristic <code><coefdiving></code>	
<code>heuristics/coefdiving/maxlpiterofs</code> ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
<code>heuristics/coefdiving/maxlpiterquot</code> ($0 \leq \text{real}$)	0.05
maximal fraction of diving LP iterations compared to node LP iterations	
<code>heuristics/crossover/freq</code> ($-1 \leq \text{integer}$)	30
frequency for calling primal heuristic <code><crossover></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/crossover/freqofs</code> ($0 \leq \text{integer}$)	0

frequency offset for calling primal heuristic <crossover>	
heuristics/crossover/minfixingrate ($0 \leq \text{real} \leq 1$)	0.666
minimum percentage of integer variables that have to be fixed	
heuristics/crossover/nodesofs ($0 \leq \text{integer}$)	500
number of nodes added to the contingent of the total nodes	
heuristics/crossover/nodesquot ($0 \leq \text{real} \leq 1$)	0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem	
heuristics/crossover/nusedsols ($2 \leq \text{integer}$)	3
number of solutions to be taken into account	
heuristics/dins/freq ($-1 \leq \text{integer}$)	-1
frequency for calling primal heuristic <dins> (-1: never, 0: only at depth freqofs)	
heuristics/dins/freqofs ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <dins>	
heuristics/dins/minnodes ($0 \leq \text{integer}$)	500
minimum number of nodes required to start the subproblem	
heuristics/dins/neighborhoodsize ($1 \leq \text{integer}$)	18
radius (using Manhattan metric) of the incumbent's neighborhood to be searched	
heuristics/dins/nodesofs ($0 \leq \text{integer}$)	5000
number of nodes added to the contingent of the total nodes	
heuristics/dins/nodesquot ($0 \leq \text{real} \leq 1$)	0.05
contingent of sub problem nodes in relation to the number of nodes of the original problem	
heuristics/dins/solnum ($1 \leq \text{integer}$)	5
number of pool-solutions to be checked for flag array update (for hard fixing of binary variables)	
heuristics/feaspump/alphadiff ($0 \leq \text{real} \leq 1$)	1
threshold difference for the convex parameter to perform perturbation	
heuristics/feaspump/beforecuts (boolean)	TRUE
should the feasibility pump be called at root node before cut separation?	
heuristics/feaspump/freq ($-1 \leq \text{integer}$)	20
frequency for calling primal heuristic <feaspump> (-1: never, 0: only at depth freqofs)	
heuristics/feaspump/freqofs ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <feaspump>	
heuristics/feaspump/maxlpiterofs ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
heuristics/feaspump/maxlpiterquot ($0 \leq \text{real}$)	0.01
maximal fraction of diving LP iterations compared to node LP iterations	
heuristics/feaspump/neighborhoodsize ($1 \leq \text{integer}$)	18
radius (using Manhattan metric) of the neighborhood to be searched in stage 3	
heuristics/feaspump/objfactor ($0 \leq \text{real} \leq 1$)	1
factor by which the regard of the objective is decreased in each round, 1.0 for dynamic	
heuristics/feaspump/pertsolfound (boolean)	TRUE
should a random perturbation be performed if a feasible solution was found?	
heuristics/feaspump/stage3 (boolean)	FALSE
should we solve a local branching sub-MIP if no solution could be found?	
heuristics/feaspump/usefp20 (boolean)	FALSE
should an iterative round-and-propagate scheme be used to find the integral points?	
heuristics/fixandinfer/freq ($-1 \leq \text{integer}$)	-1

frequency for calling primal heuristic <fixandinfer> (-1: never, 0: only at depth freqofs)	
heuristics/fixandinfer/freqofs ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <fixandinfer>	
heuristics/fracdiving/backtrack (boolean)	TRUE
use one level of backtracking if infeasibility is encountered?	
heuristics/fracdiving/freq ($-1 \leq \text{integer}$)	10
frequency for calling primal heuristic <fracdiving> (-1: never, 0: only at depth freqofs)	
heuristics/fracdiving/freqofs ($0 \leq \text{integer}$)	3
frequency offset for calling primal heuristic <fracdiving>	
heuristics/fracdiving/maxlpiterofs ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
heuristics/fracdiving/maxlpiterquot ($0 \leq \text{real}$)	0.05
maximal fraction of diving LP iterations compared to node LP iterations	
heuristics/guideddiving/backtrack (boolean)	TRUE
use one level of backtracking if infeasibility is encountered?	
heuristics/guideddiving/freq ($-1 \leq \text{integer}$)	10
frequency for calling primal heuristic <guideddiving> (-1: never, 0: only at depth freqofs)	
heuristics/guideddiving/freqofs ($0 \leq \text{integer}$)	7
frequency offset for calling primal heuristic <guideddiving>	
heuristics/guideddiving/maxlpiterofs ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
heuristics/guideddiving/maxlpiterquot ($0 \leq \text{real}$)	0.05
maximal fraction of diving LP iterations compared to node LP iterations	
heuristics/intdiving/backtrack (boolean)	TRUE
use one level of backtracking if infeasibility is encountered?	
heuristics/intdiving/freq ($-1 \leq \text{integer}$)	-1
frequency for calling primal heuristic <intdiving> (-1: never, 0: only at depth freqofs)	
heuristics/intdiving/freqofs ($0 \leq \text{integer}$)	9
frequency offset for calling primal heuristic <intdiving>	
heuristics/intdiving/maxlpiterofs ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
heuristics/intdiving/maxlpiterquot ($0 \leq \text{real}$)	0.05
maximal fraction of diving LP iterations compared to node LP iterations	
heuristics/intshifting/freq ($-1 \leq \text{integer}$)	10
frequency for calling primal heuristic <intshifting> (-1: never, 0: only at depth freqofs)	
heuristics/intshifting/freqofs ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <intshifting>	
heuristics/linesearchdiving/backtrack (boolean)	TRUE
use one level of backtracking if infeasibility is encountered?	
heuristics/linesearchdiving/freq ($-1 \leq \text{integer}$)	10
frequency for calling primal heuristic <linesearchdiving> (-1: never, 0: only at depth freqofs)	
heuristics/linesearchdiving/freqofs ($0 \leq \text{integer}$)	6
frequency offset for calling primal heuristic <linesearchdiving>	
heuristics/linesearchdiving/maxlpiterofs ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
heuristics/linesearchdiving/maxlpiterquot ($0 \leq \text{real}$)	0.05

maximal fraction of diving LP iterations compared to node LP iterations	
<code>heuristics/localbranching/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling primal heuristic <localbranching> (-1: never, 0: only at depth freqofs)	
<code>heuristics/localbranching/freqofs</code> ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <localbranching>	
<code>heuristics/localbranching/neighborhoodsize</code> ($1 \leq \text{integer}$)	18
radius (using Manhattan metric) of the incumbent's neighborhood to be searched	
<code>heuristics/localbranching/nodesofs</code> ($0 \leq \text{integer}$)	1000
number of nodes added to the contingent of the total nodes	
<code>heuristics/localbranching/nodesquot</code> ($0 \leq \text{real} \leq 1$)	0.05
contingent of sub problem nodes in relation to the number of nodes of the original problem	
<code>heuristics/mutation/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling primal heuristic <mutation> (-1: never, 0: only at depth freqofs)	
<code>heuristics/mutation/freqofs</code> ($0 \leq \text{integer}$)	8
frequency offset for calling primal heuristic <mutation>	
<code>heuristics/mutation/minfixingrate</code> ($10^{-6} \leq \text{real} \leq 0.999999$)	0.8
percentage of integer variables that have to be fixed	
<code>heuristics/mutation/nodesofs</code> ($0 \leq \text{integer}$)	500
number of nodes added to the contingent of the total nodes	
<code>heuristics/mutation/nodesquot</code> ($0 \leq \text{real} \leq 1$)	0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem	
<code>heuristics/objpscostdiving/freq</code> ($-1 \leq \text{integer}$)	20
frequency for calling primal heuristic <objpscostdiving> (-1: never, 0: only at depth freqofs)	
<code>heuristics/objpscostdiving/freqofs</code> ($0 \leq \text{integer}$)	4
frequency offset for calling primal heuristic <objpscostdiving>	
<code>heuristics/objpscostdiving/maxlpiterofs</code> ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
<code>heuristics/objpscostdiving/maxlpiterquot</code> ($0 \leq \text{real} \leq 1$)	0.01
maximal fraction of diving LP iterations compared to total iteration number	
<code>heuristics/octane/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling primal heuristic <octane> (-1: never, 0: only at depth freqofs)	
<code>heuristics/octane/freqofs</code> ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <octane>	
<code>heuristics/oneopt/freq</code> ($-1 \leq \text{integer}$)	1
frequency for calling primal heuristic <oneopt> (-1: never, 0: only at depth freqofs)	
<code>heuristics/oneopt/freqofs</code> ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <oneopt>	
<code>heuristics/pscostdiving/backtrack</code> (boolean)	TRUE
use one level of backtracking if infeasibility is encountered?	
<code>heuristics/pscostdiving/freq</code> ($-1 \leq \text{integer}$)	10
frequency for calling primal heuristic <pscostdiving> (-1: never, 0: only at depth freqofs)	
<code>heuristics/pscostdiving/freqofs</code> ($0 \leq \text{integer}$)	2
frequency offset for calling primal heuristic <pscostdiving>	
<code>heuristics/pscostdiving/maxlpiterofs</code> ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
<code>heuristics/pscostdiving/maxlpiterquot</code> ($0 \leq \text{real}$)	0.05

maximal fraction of diving LP iterations compared to node LP iterations	
<code>heuristics/rens/freq</code> ($-1 \leq \text{integer}$)	0
frequency for calling primal heuristic <code><rens></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/rens/freqofs</code> ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <code><rens></code>	
<code>heuristics/rens/minfixingrate</code> ($0 \leq \text{real} \leq 1$)	0.5
minimum percentage of integer variables that have to be fixable	
<code>heuristics/rens/nodesofs</code> ($0 \leq \text{integer}$)	500
number of nodes added to the contingent of the total nodes	
<code>heuristics/rens/nodesquot</code> ($0 \leq \text{real} \leq 1$)	0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem	
<code>heuristics/rens/startsol</code> (character)	1
solution that is used for fixing values ('lp relaxation, 'n'lp relaxation)	
<code>heuristics/rins/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling primal heuristic <code><rins></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/rins/freqofs</code> ($0 \leq \text{integer}$)	5
frequency offset for calling primal heuristic <code><rins></code>	
<code>heuristics/rins/minfixingrate</code> ($0 \leq \text{real} \leq 1$)	0
minimum percentage of integer variables that have to be fixed	
<code>heuristics/rins/nodesofs</code> ($0 \leq \text{integer}$)	500
number of nodes added to the contingent of the total nodes	
<code>heuristics/rins/nodesquot</code> ($0 \leq \text{real} \leq 1$)	0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem	
<code>heuristics/rootsoldiving/freq</code> ($-1 \leq \text{integer}$)	20
frequency for calling primal heuristic <code><rootsoldiving></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/rootsoldiving/freqofs</code> ($0 \leq \text{integer}$)	5
frequency offset for calling primal heuristic <code><rootsoldiving></code>	
<code>heuristics/rootsoldiving/maxlpiterofs</code> ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
<code>heuristics/rootsoldiving/maxlpiterquot</code> ($0 \leq \text{real}$)	0.01
maximal fraction of diving LP iterations compared to node LP iterations	
<code>heuristics/rounding/freq</code> ($-1 \leq \text{integer}$)	1
frequency for calling primal heuristic <code><rounding></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/rounding/freqofs</code> ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <code><rounding></code>	
<code>heuristics/shiftandpropagate/freq</code> ($-1 \leq \text{integer}$)	0
frequency for calling primal heuristic <code><shiftandpropagate></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/shiftandpropagate/freqofs</code> ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <code><shiftandpropagate></code>	
<code>heuristics/shifting/freq</code> ($-1 \leq \text{integer}$)	10
frequency for calling primal heuristic <code><shifting></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/shifting/freqofs</code> ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <code><shifting></code>	
<code>heuristics/simplerounding/freq</code> ($-1 \leq \text{integer}$)	1
frequency for calling primal heuristic <code><simplerounding></code> (-1: never, 0: only at depth freqofs)	
<code>heuristics/simplerounding/freqofs</code> ($0 \leq \text{integer}$)	0

frequency offset for calling primal heuristic <simplerounding>	
heuristics/subnlp/forbidfixings (boolean)	TRUE
whether to add constraints that forbid specific fixings that turned out to be infeasible	
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/itermin ($0 \leq$ integer)	300
contingent of NLP iterations in relation to the number of nodes in SCIP	
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/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/nlpverblevel ($0 \leq$ integer)	0
verbosity level of NLP solver	
heuristics/subnlp/runalways (boolean)	FALSE
whether to run NLP heuristic always if starting point available (does not use iteroffset,iterquot,itermin)	
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/fixingalts (string)	li
prioritized sequence of fixing values used ('p relaxation, 'nlp relaxation, 'incumbent solution)	
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/nodesofs ($0 \leq$ integer)	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/vbounds/freq ($-1 \leq \text{integer}$)	-1
frequency for calling primal heuristic <vbounds> (-1: never, 0: only at depth freqofs)	
heuristics/vbounds/freqofs ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <vbounds>	
heuristics/vbounds/minfixingrate ($0 \leq \text{real} \leq 1$)	0.5
minimum percentage of integer variables that have to be fixable	
heuristics/vbounds/nodesofs ($0 \leq \text{integer}$)	500
number of nodes added to the contingent of the total nodes	
heuristics/vbounds/nodesquot ($0 \leq \text{real} \leq 1$)	0.1
contingent of sub problem nodes in relation to the number of nodes of the original problem	
heuristics/veclending/backtrack (boolean)	TRUE
use one level of backtracking if infeasibility is encountered?	
heuristics/veclending/freq ($-1 \leq \text{integer}$)	10
frequency for calling primal heuristic <veclending> (-1: never, 0: only at depth freqofs)	
heuristics/veclending/freqofs ($0 \leq \text{integer}$)	4
frequency offset for calling primal heuristic <veclending>	
heuristics/veclending/maxlpiterofs ($0 \leq \text{integer}$)	1000
additional number of allowed LP iterations	
heuristics/veclending/maxlpiterquot ($0 \leq \text{real}$)	0.05
maximal fraction of diving LP iterations compared to node LP iterations	
heuristics/zirounding/freq ($-1 \leq \text{integer}$)	1
frequency for calling primal heuristic <zirounding> (-1: never, 0: only at depth freqofs)	
heuristics/zirounding/freqofs ($0 \leq \text{integer}$)	0
frequency offset for calling primal heuristic <zirounding>	
Heuristics (advanced options)	
heuristics/actconsdiving/maxdepth ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <actconsdiving> (-1: no limit)	
heuristics/actconsdiving/maxdiveavgquot ($0 \leq \text{real}$)	0
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{avglowerbound} - \text{lowerbound})$ where diving is performed (0.0: no limit)	
heuristics/actconsdiving/maxdiveavgquotnosol ($0 \leq \text{real}$)	0
maximal AVGQUOT when no solution was found yet (0.0: no limit)	
heuristics/actconsdiving/maxdiveubquot ($0 \leq \text{real} \leq 1$)	0.8
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{cutoffbound} - \text{lowerbound})$ where diving is performed (0.0: no limit)	
heuristics/actconsdiving/maxdiveubquotnosol ($0 \leq \text{real} \leq 1$)	0.1
maximal UBQUOT when no solution was found yet (0.0: no limit)	
heuristics/actconsdiving/maxreldepth ($0 \leq \text{real} \leq 1$)	1
maximal relative depth to start diving	
heuristics/actconsdiving/minreldepth ($0 \leq \text{real} \leq 1$)	0
minimal relative depth to start diving	
heuristics/actconsdiving/priority ($-536870912 \leq \text{integer} \leq 536870911$)	-1003700
priority of heuristic <actconsdiving>	
heuristics/cliقة/copycuts (boolean)	TRUE
should all active cuts from cutpool be copied to constraints in subproblem?	

heuristics/cliq/initialseed (0 ≤ integer)	0
initial random seed value to permute variables	
heuristics/cliq/maxdepth (-1 ≤ integer)	-1
maximal depth level to call primal heuristic <cliq> (-1: no limit)	
heuristics/cliq/maxnodes (0 ≤ integer)	5000
maximum number of nodes to regard in the subproblem	
heuristics/cliq/maxpropounds (-1 ≤ integer ≤ 536870911)	2
maximum number of propagation rounds during probing (-1 infinity)	
heuristics/cliq/minimprove (0 ≤ real ≤ 1)	0.01
factor by which clique heuristic should at least improve the incumbent	
heuristics/cliq/minnodes (0 ≤ integer)	500
minimum number of nodes required to start the subproblem	
heuristics/cliq/multiplier (0 ≤ real)	1.1
value to increase nodenum to determine the next run	
heuristics/cliq/priority (-536870912 ≤ integer ≤ 536870911)	-1000500
priority of heuristic <cliq>	
heuristics/coefdiving/maxdepth (-1 ≤ integer)	-1
maximal depth level to call primal heuristic <coefdiving> (-1: no limit)	
heuristics/coefdiving/maxdiveavgquot (0 ≤ real)	0
maximal quotient (curlowerbound - lowerbound)/(avglowerbound - lowerbound) where diving is performed (0.0: no limit)	
heuristics/coefdiving/maxdiveavgquotnosol (0 ≤ real)	0
maximal AVGQUOT when no solution was found yet (0.0: no limit)	
heuristics/coefdiving/maxdiveubquot (0 ≤ real ≤ 1)	0.8
maximal quotient (curlowerbound - lowerbound)/(cutoffbound - lowerbound) where diving is performed (0.0: no limit)	
heuristics/coefdiving/maxdiveubquotnosol (0 ≤ real ≤ 1)	0.1
maximal UBQUOT when no solution was found yet (0.0: no limit)	
heuristics/coefdiving/maxreldepth (0 ≤ real ≤ 1)	1
maximal relative depth to start diving	
heuristics/coefdiving/minreldepth (0 ≤ real ≤ 1)	0
minimal relative depth to start diving	
heuristics/coefdiving/priority (-536870912 ≤ integer ≤ 536870911)	-1001000
priority of heuristic <coefdiving>	
heuristics/crossover/copycuts (boolean)	TRUE
if uselprows == FALSE, should all active cuts from cutpool be copied to constraints in subproblem?	
heuristics/crossover/dontwaitatroot (boolean)	FALSE
should the nwaitingnodes parameter be ignored at the root node?	
heuristics/crossover/maxdepth (-1 ≤ integer)	-1
maximal depth level to call primal heuristic <crossover> (-1: no limit)	
heuristics/crossover/maxnodes (0 ≤ integer)	5000
maximum number of nodes to regard in the subproblem	
heuristics/crossover/minimprove (0 ≤ real ≤ 1)	0.01
factor by which Crossover should at least improve the incumbent	
heuristics/crossover/minnodes (0 ≤ integer)	500
minimum number of nodes required to start the subproblem	
heuristics/crossover/nwaitingnodes (0 ≤ integer)	200

number of nodes without incumbent change that heuristic should wait	
heuristics/crossover/priority ($-536870912 \leq \text{integer} \leq 536870911$) priority of heuristic <crossover>	-1104000
heuristics/crossover/randomization (boolean) should the choice which sols to take be randomized?	TRUE
heuristics/crossover/uselprows (boolean) should subproblem be created out of the rows in the LP rows?	FALSE
heuristics/dins/copycuts (boolean) if uselprows == FALSE, should all active cuts from cutpool be copied to constraints in subproblem?	TRUE
heuristics/dins/maxdepth ($-1 \leq \text{integer}$) maximal depth level to call primal heuristic <dins> (-1: no limit)	-1
heuristics/dins/maxnodes ($0 \leq \text{integer}$) maximum number of nodes to regard in the subproblem	5000
heuristics/dins/minimprove ($0 \leq \text{real} \leq 1$) factor by which dins should at least improve the incumbent	0.01
heuristics/dins/nwaitingnodes ($0 \leq \text{integer}$) number of nodes without incumbent change that heuristic should wait	0
heuristics/dins/priority ($-536870912 \leq \text{integer} \leq 536870911$) priority of heuristic <dins>	-1105000
heuristics/dins/uselprows (boolean) should subproblem be created out of the rows in the LP rows?	FALSE
heuristics/feaspump/copycuts (boolean) should all active cuts from cutpool be copied to constraints in subproblem?	TRUE
heuristics/feaspump/cyclelength ($1 \leq \text{integer} \leq 100$) maximum length of cycles to be checked explicitly in each round	3
heuristics/feaspump/maxdepth ($-1 \leq \text{integer}$) maximal depth level to call primal heuristic <feaspump> (-1: no limit)	-1
heuristics/feaspump/maxloops ($-1 \leq \text{integer}$) maximal number of pumping loops (-1: no limit)	10000
heuristics/feaspump/maxsols ($-1 \leq \text{integer}$) total number of feasible solutions found up to which heuristic is called (-1: no limit)	2
heuristics/feaspump/maxstallloops ($-1 \leq \text{integer}$) maximal number of pumping rounds without fractionality improvement (-1: no limit)	10
heuristics/feaspump/minflips ($1 \leq \text{integer}$) minimum number of random variables to flip, if a 1-cycle is encountered	10
heuristics/feaspump/perturbfreq ($1 \leq \text{integer}$) number of iterations until a random perturbation is forced	100
heuristics/feaspump/priority ($-536870912 \leq \text{integer} \leq 536870911$) priority of heuristic <feaspump>	-1000000
heuristics/fixandinfer/maxdepth ($-1 \leq \text{integer}$) maximal depth level to call primal heuristic <fixandinfer> (-1: no limit)	-1
heuristics/fixandinfer/minfixings ($0 \leq \text{integer}$) minimal number of fixings to apply before dive may be aborted	100
heuristics/fixandinfer/priority ($-536870912 \leq \text{integer} \leq 536870911$) priority of heuristic <fixandinfer>	-500000
heuristics/fixandinfer/proprounds ($-1 \leq \text{integer}$)	0

maximal number of propagation rounds in probing subproblems (-1: no limit, 0: auto)	
<code>heuristics/fracdiving/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <fracdiving> (-1: no limit)	
<code>heuristics/fracdiving/maxdiveavgquot</code> ($0 \leq \text{real}$)	0
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{avglowerbound} - \text{lowerbound})$ where diving is performed (0.0: no limit)	
<code>heuristics/fracdiving/maxdiveavgquotnosol</code> ($0 \leq \text{real}$)	0
maximal AVGQUOT when no solution was found yet (0.0: no limit)	
<code>heuristics/fracdiving/maxdiveubquot</code> ($0 \leq \text{real} \leq 1$)	0.8
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{cutoffbound} - \text{lowerbound})$ where diving is performed (0.0: no limit)	
<code>heuristics/fracdiving/maxdiveubquotnosol</code> ($0 \leq \text{real} \leq 1$)	0.1
maximal UBQUOT when no solution was found yet (0.0: no limit)	
<code>heuristics/fracdiving/maxreldepth</code> ($0 \leq \text{real} \leq 1$)	1
maximal relative depth to start diving	
<code>heuristics/fracdiving/minreldepth</code> ($0 \leq \text{real} \leq 1$)	0
minimal relative depth to start diving	
<code>heuristics/fracdiving/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1003000
priority of heuristic <fracdiving>	
<code>heuristics/guideddiving/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <guideddiving> (-1: no limit)	
<code>heuristics/guideddiving/maxdiveavgquot</code> ($0 \leq \text{real}$)	0
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{avglowerbound} - \text{lowerbound})$ where diving is performed (0.0: no limit)	
<code>heuristics/guideddiving/maxdiveubquot</code> ($0 \leq \text{real} \leq 1$)	0.8
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{cutoffbound} - \text{lowerbound})$ where diving is performed (0.0: no limit)	
<code>heuristics/guideddiving/maxreldepth</code> ($0 \leq \text{real} \leq 1$)	1
maximal relative depth to start diving	
<code>heuristics/guideddiving/minreldepth</code> ($0 \leq \text{real} \leq 1$)	0
minimal relative depth to start diving	
<code>heuristics/guideddiving/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1007000
priority of heuristic <guideddiving>	
<code>heuristics/intdiving/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <intdiving> (-1: no limit)	
<code>heuristics/intdiving/maxdiveavgquot</code> ($0 \leq \text{real}$)	0
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{avglowerbound} - \text{lowerbound})$ where diving is performed (0.0: no limit)	
<code>heuristics/intdiving/maxdiveavgquotnosol</code> ($0 \leq \text{real}$)	0
maximal AVGQUOT when no solution was found yet (0.0: no limit)	
<code>heuristics/intdiving/maxdiveubquot</code> ($0 \leq \text{real} \leq 1$)	0.8
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{cutoffbound} - \text{lowerbound})$ where diving is performed (0.0: no limit)	
<code>heuristics/intdiving/maxdiveubquotnosol</code> ($0 \leq \text{real} \leq 1$)	0.1
maximal UBQUOT when no solution was found yet (0.0: no limit)	
<code>heuristics/intdiving/maxreldepth</code> ($0 \leq \text{real} \leq 1$)	1
maximal relative depth to start diving	

<code>heuristics/intdiving/minreldepth</code> ($0 \leq \text{real} \leq 1$)	0
minimal relative depth to start diving	
<code>heuristics/intdiving/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1003500
priority of heuristic <intdiving>	
<code>heuristics/intshifting/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <intshifting> (-1: no limit)	
<code>heuristics/intshifting/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-10000
priority of heuristic <intshifting>	
<code>heuristics/linesearchdiving/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <linesearchdiving> (-1: no limit)	
<code>heuristics/linesearchdiving/maxdiveavquot</code> ($0 \leq \text{real}$)	0
maximal quotient (curlowerbound - lowerbound)/(avglowerbound - lowerbound) where diving is performed (0.0: no limit)	
<code>heuristics/linesearchdiving/maxdiveavquotnosol</code> ($0 \leq \text{real}$)	0
maximal AVQUOT when no solution was found yet (0.0: no limit)	
<code>heuristics/linesearchdiving/maxdiveubquot</code> ($0 \leq \text{real} \leq 1$)	0.8
maximal quotient (curlowerbound - lowerbound)/(cutoffbound - lowerbound) where diving is performed (0.0: no limit)	
<code>heuristics/linesearchdiving/maxdiveubquotnosol</code> ($0 \leq \text{real} \leq 1$)	0.1
maximal UBQUOT when no solution was found yet (0.0: no limit)	
<code>heuristics/linesearchdiving/maxreldepth</code> ($0 \leq \text{real} \leq 1$)	1
maximal relative depth to start diving	
<code>heuristics/linesearchdiving/minreldepth</code> ($0 \leq \text{real} \leq 1$)	0
minimal relative depth to start diving	
<code>heuristics/linesearchdiving/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1006000
priority of heuristic <linesearchdiving>	
<code>heuristics/localbranching/copycuts</code> (boolean)	TRUE
if <code>uselprows == FALSE</code> , should all active cuts from cutpool be copied to constraints in subproblem?	
<code>heuristics/localbranching/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <localbranching> (-1: no limit)	
<code>heuristics/localbranching/maxnodes</code> ($0 \leq \text{integer}$)	10000
maximum number of nodes to regard in the subproblem	
<code>heuristics/localbranching/minimprove</code> ($0 \leq \text{real} \leq 1$)	0.01
factor by which localbranching should at least improve the incumbent	
<code>heuristics/localbranching/minnodes</code> ($0 \leq \text{integer}$)	1000
minimum number of nodes required to start the subproblem	
<code>heuristics/localbranching/nwaitingnodes</code> ($0 \leq \text{integer}$)	200
number of nodes without incumbent change that heuristic should wait	
<code>heuristics/localbranching/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1102000
priority of heuristic <localbranching>	
<code>heuristics/localbranching/uselprows</code> (boolean)	FALSE
should subproblem be created out of the rows in the LP rows?	
<code>heuristics/mutation/copycuts</code> (boolean)	TRUE
if <code>uselprows == FALSE</code> , should all active cuts from cutpool be copied to constraints in subproblem?	
<code>heuristics/mutation/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <mutation> (-1: no limit)	
<code>heuristics/mutation/maxnodes</code> ($0 \leq \text{integer}$)	5000

maximum number of nodes to regard in the subproblem	
<code>heuristics/mutation/minimprove</code> ($0 \leq \text{real} \leq 1$)	0.01
factor by which mutation should at least improve the incumbent	
<code>heuristics/mutation/minnodes</code> ($0 \leq \text{integer}$)	500
minimum number of nodes required to start the subproblem	
<code>heuristics/mutation/nwaitingnodes</code> ($0 \leq \text{integer}$)	200
number of nodes without incumbent change that heuristic should wait	
<code>heuristics/mutation/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1103000
priority of heuristic <mutation>	
<code>heuristics/mutation/uselprows</code> (boolean)	FALSE
should subproblem be created out of the rows in the LP rows?	
<code>heuristics/objpscostdiving/depthfac</code> ($0 \leq \text{real}$)	0.5
maximal diving depth: number of binary/integer variables times depthfac	
<code>heuristics/objpscostdiving/depthfacnosol</code> ($0 \leq \text{real}$)	2
maximal diving depth factor if no feasible solution was found yet	
<code>heuristics/objpscostdiving/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <objpscostdiving> (-1: no limit)	
<code>heuristics/objpscostdiving/maxreldepth</code> ($0 \leq \text{real} \leq 1$)	1
maximal relative depth to start diving	
<code>heuristics/objpscostdiving/maxsols</code> ($-1 \leq \text{integer}$)	-1
total number of feasible solutions found up to which heuristic is called (-1: no limit)	
<code>heuristics/objpscostdiving/minreldepth</code> ($0 \leq \text{real} \leq 1$)	0
minimal relative depth to start diving	
<code>heuristics/objpscostdiving/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1004000
priority of heuristic <objpscostdiving>	
<code>heuristics/octane/ffirst</code> ($1 \leq \text{integer}$)	10
number of 0-1-points to be tested at first whether they violate a common row	
<code>heuristics/octane/fmax</code> ($1 \leq \text{integer}$)	100
number of 0-1-points to be tested as possible solutions by OCTANE	
<code>heuristics/octane/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <octane> (-1: no limit)	
<code>heuristics/octane/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1008000
priority of heuristic <octane>	
<code>heuristics/octane/useavgnbray</code> (boolean)	TRUE
should the weighted average of the nonbasic cone be used as one ray direction?	
<code>heuristics/octane/useavgray</code> (boolean)	TRUE
should the average of the basic cone be used as one ray direction?	
<code>heuristics/octane/useavwgtray</code> (boolean)	TRUE
should the weighted average of the basic cone be used as one ray direction?	
<code>heuristics/octane/usediffgray</code> (boolean)	FALSE
should the difference between the root solution and the current LP solution be used as one ray direction?	
<code>heuristics/octane/usefracspace</code> (boolean)	TRUE
execute OCTANE only in the space of fractional variables (TRUE) or in the full space?	
<code>heuristics/octane/useobjray</code> (boolean)	TRUE
should the inner normal of the objective be used as one ray direction?	
<code>heuristics/oneopt/duringroot</code> (boolean)	TRUE

should the heuristic be called before and during the root node?	
heuristics/oneopt/maxdepth ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <oneopt> (-1: no limit)	
heuristics/oneopt/priority ($-536870912 \leq \text{integer} \leq 536870911$)	-20000
priority of heuristic <oneopt>	
heuristics/oneopt/weightedobj (boolean)	TRUE
should the objective be weighted with the potential shifting value when sorting the shifting candidates?	
heuristics/pscostdiving/maxdepth ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <pscostdiving> (-1: no limit)	
heuristics/pscostdiving/maxdiveavgquot ($0 \leq \text{real}$)	0
maximal quotient (curlowerbound - lowerbound)/(avglowerbound - lowerbound) where diving is performed (0.0: no limit)	
heuristics/pscostdiving/maxdiveavgquotnosol ($0 \leq \text{real}$)	0
maximal AVGQUOT when no solution was found yet (0.0: no limit)	
heuristics/pscostdiving/maxdiveubquot ($0 \leq \text{real} \leq 1$)	0.8
maximal quotient (curlowerbound - lowerbound)/(cutoffbound - lowerbound) where diving is performed (0.0: no limit)	
heuristics/pscostdiving/maxdiveubquotnosol ($0 \leq \text{real} \leq 1$)	0.1
maximal UBQUOT when no solution was found yet (0.0: no limit)	
heuristics/pscostdiving/maxreldepth ($0 \leq \text{real} \leq 1$)	1
maximal relative depth to start diving	
heuristics/pscostdiving/minreldepth ($0 \leq \text{real} \leq 1$)	0
minimal relative depth to start diving	
heuristics/pscostdiving/priority ($-536870912 \leq \text{integer} \leq 536870911$)	-1002000
priority of heuristic <pscostdiving>	
heuristics/rens/binarybounds (boolean)	TRUE
should general integers get binary bounds [floor(.),ceil(.)] ?	
heuristics/rens/copycuts (boolean)	TRUE
if uselprows == FALSE, should all active cuts from cutpool be copied to constraints in subproblem?	
heuristics/rens/maxdepth ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <rens> (-1: no limit)	
heuristics/rens/maxnodes ($0 \leq \text{integer}$)	5000
maximum number of nodes to regard in the subproblem	
heuristics/rens/minimprove ($0 \leq \text{real} \leq 1$)	0.01
factor by which RENS should at least improve the incumbent	
heuristics/rens/minnodes ($0 \leq \text{integer}$)	500
minimum number of nodes required to start the subproblem	
heuristics/rens/priority ($-536870912 \leq \text{integer} \leq 536870911$)	-1100000
priority of heuristic <rens>	
heuristics/rens/uselprows (boolean)	FALSE
should subproblem be created out of the rows in the LP rows?	
heuristics/rins/copycuts (boolean)	TRUE
if uselprows == FALSE, should all active cuts from cutpool be copied to constraints in subproblem?	
heuristics/rins/maxdepth ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <rins> (-1: no limit)	
heuristics/rins/maxnodes ($0 \leq \text{integer}$)	5000
maximum number of nodes to regard in the subproblem	

<code>heuristics/rins/minimprove</code> ($0 \leq \text{real} \leq 1$)	0.01
factor by which rins should at least improve the incumbent	
<code>heuristics/rins/minnodes</code> ($0 \leq \text{integer}$)	500
minimum number of nodes required to start the subproblem	
<code>heuristics/rins/nwaitingnodes</code> ($0 \leq \text{integer}$)	200
number of nodes without incumbent change that heuristic should wait	
<code>heuristics/rins/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1101000
priority of heuristic <rins>	
<code>heuristics/rins/uselprows</code> (boolean)	FALSE
should subproblem be created out of the rows in the LP rows?	
<code>heuristics/rootsoldiving/alpha</code> ($0 \leq \text{real} \leq 1$)	0.9
soft rounding factor to fade out objective coefficients	
<code>heuristics/rootsoldiving/depthfac</code> ($0 \leq \text{real}$)	0.5
maximal diving depth: number of binary/integer variables times depthfac	
<code>heuristics/rootsoldiving/depthfacnosol</code> ($0 \leq \text{real}$)	2
maximal diving depth factor if no feasible solution was found yet	
<code>heuristics/rootsoldiving/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <rootsoldiving> (-1: no limit)	
<code>heuristics/rootsoldiving/maxreldepth</code> ($0 \leq \text{real} \leq 1$)	1
maximal relative depth to start diving	
<code>heuristics/rootsoldiving/maxsols</code> ($-1 \leq \text{integer}$)	-1
total number of feasible solutions found up to which heuristic is called (-1: no limit)	
<code>heuristics/rootsoldiving/minreldepth</code> ($0 \leq \text{real} \leq 1$)	0
minimal relative depth to start diving	
<code>heuristics/rootsoldiving/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1005000
priority of heuristic <rootsoldiving>	
<code>heuristics/rounding/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <rounding> (-1: no limit)	
<code>heuristics/rounding/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1000
priority of heuristic <rounding>	
<code>heuristics/rounding/successfactor</code> ($-1 \leq \text{integer}$)	100
number of calls per found solution that are considered as standard success, a higher factor causes the heuristic to be called more often	
<code>heuristics/shiftandpropagate/cutoffbreaker</code> ($-1 \leq \text{integer} \leq 1000000$)	15
The number of cutoffs before heuristic stops	
<code>heuristics/shiftandpropagate/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <shiftandpropagate> (-1: no limit)	
<code>heuristics/shiftandpropagate/npropounds</code> ($-1 \leq \text{integer} \leq 1000$)	10
The number of propagation rounds used for each propagation	
<code>heuristics/shiftandpropagate/onlywithoutsol</code> (boolean)	TRUE
Should heuristic only be executed if no primal solution was found, yet?	
<code>heuristics/shiftandpropagate/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	1000
priority of heuristic <shiftandpropagate>	
<code>heuristics/shiftandpropagate/probing</code> (boolean)	TRUE
Should domains be reduced by probing?	
<code>heuristics/shiftandpropagate/relax</code> (boolean)	TRUE
Should continuous variables be relaxed?	

<code>heuristics/shiftandpropagate/sortkey</code> (character)	u
the key for variable sorting: (n)orms or (r)andom	
<code>heuristics/shiftandpropagate/sortvars</code> (boolean)	TRUE
Should variables be sorted for the heuristic?	
<code>heuristics/shifting/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <shifting> (-1: no limit)	
<code>heuristics/shifting/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-5000
priority of heuristic <shifting>	
<code>heuristics/simplerounding/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <simplerounding> (-1: no limit)	
<code>heuristics/simplerounding/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	0
priority of heuristic <simplerounding>	
<code>heuristics/subnlp/keepcopy</code> (boolean)	TRUE
whether to keep SCIP copy or to create new copy each time heuristic is applied	
<code>heuristics/subnlp/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <subnlp> (-1: no limit)	
<code>heuristics/subnlp/maxpresolverounds</code> ($-1 \leq \text{integer}$)	-1
limit on number of presolve rounds in sub-SCIP (-1 for unlimited, 0 for no presolve)	
<code>heuristics/subnlp/minimprove</code> ($0 \leq \text{real} \leq 1$)	0.01
factor by which NLP heuristic should at least improve the incumbent	
<code>heuristics/subnlp/nlpoptfile</code> (string)	
name of an NLP solver specific options file	
<code>heuristics/subnlp/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-2000000
priority of heuristic <subnlp>	
<code>heuristics/subnlp/resolvefromscratch</code> (boolean)	TRUE
should the NLP resolve be started from the original starting point or the infeasible solution?	
<code>heuristics/subnlp/resolvetolfactor</code> ($0 \leq \text{real} \leq 1$)	0.001
if SCIP does not accept a NLP feasible solution, resolve NLP with feas. tolerance reduced by this factor (set to 1.0 to turn off resolve)	
<code>heuristics/trivial/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <trivial> (-1: no limit)	
<code>heuristics/trivial/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	10000
priority of heuristic <trivial>	
<code>heuristics/trysol/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <trysol> (-1: no limit)	
<code>heuristics/trysol/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-3000000
priority of heuristic <trysol>	
<code>heuristics/twoopt/intopt</code> (boolean)	FALSE
Should Integer-2-Optimization be applied or not?	
<code>heuristics/twoopt/matchingrate</code> ($0 \leq \text{real} \leq 1$)	0.5
parameter to determine the percentage of rows two variables have to share before they are considered equal	
<code>heuristics/twoopt/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <twoopt> (-1: no limit)	
<code>heuristics/twoopt/maxnslaves</code> ($-1 \leq \text{integer} \leq 1000000$)	199
maximum number of slaves for one master variable	
<code>heuristics/twoopt/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-20100
priority of heuristic <twoopt>	

<code>heuristics/twoopt/waitingnodes</code> ($0 \leq \text{integer} \leq 10000$)	0
user parameter to determine number of nodes to wait after last best solution before calling heuristic	
<code>heuristics/undercover/beforecuts</code> (boolean)	TRUE
should the heuristic be called at root node before cut separation?	
<code>heuristics/undercover/conflictweight</code> (real)	1000
weight for conflict score in fixing order	
<code>heuristics/undercover/copycuts</code> (boolean)	TRUE
should all active cuts from cutpool be copied to constraints in subproblem?	
<code>heuristics/undercover/coveringobj</code> (character)	u
objective function of the covering problem ('b' ranching status, influenced nonlinear 'c' onstraints/'t' erms, 'd' omain size, 'l' ocks, 'm' in of up/down locks, 'u' nit penalties, constraint 'v' iolation)	
<code>heuristics/undercover/cutoffweight</code> ($0 \leq \text{real}$)	1
weight for cutoff score in fixing order	
<code>heuristics/undercover/fixintfirst</code> (boolean)	FALSE
should integer variables in the cover be fixed first?	
<code>heuristics/undercover/inferenceweight</code> (real)	1
weight for inference score in fixing order	
<code>heuristics/undercover/locksrounding</code> (boolean)	TRUE
shall LP values for integer vars be rounded according to locks?	
<code>heuristics/undercover/maxbacktracks</code> ($0 \leq \text{integer}$)	6
maximum number of backtracks in fix-and-propagate	
<code>heuristics/undercover/maxcoversize</code> ($0 \leq \text{real} \leq 1$)	1
maximum coversize (as fraction of total number of variables)	
<code>heuristics/undercover/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <undercover> (-1: no limit)	
<code>heuristics/undercover/maxnodes</code> ($0 \leq \text{integer}$)	500
maximum number of nodes to regard in the subproblem	
<code>heuristics/undercover/maxrecovers</code> ($0 \leq \text{integer}$)	0
maximum number of recoverings	
<code>heuristics/undercover/maxreorders</code> ($0 \leq \text{integer}$)	1
maximum number of reorderings of the fixing order	
<code>heuristics/undercover/minimprove</code> ($-1 \leq \text{real} \leq 1$)	0
factor by which the heuristic should at least improve the incumbent	
<code>heuristics/undercover/minnodes</code> ($0 \leq \text{integer}$)	500
minimum number of nodes required to start the subproblem	
<code>heuristics/undercover/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1110000
priority of heuristic <undercover>	
<code>heuristics/undercover/recoverdiv</code> ($0 \leq \text{real} \leq 1$)	0.9
fraction of covering variables in the last cover which need to change their value when recovering	
<code>heuristics/vbounds/copycuts</code> (boolean)	TRUE
should all active cuts from cutpool be copied to constraints in subproblem?	
<code>heuristics/vbounds/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <vbounds> (-1: no limit)	
<code>heuristics/vbounds/maxnodes</code> ($0 \leq \text{integer}$)	5000
maximum number of nodes to regard in the subproblem	
<code>heuristics/vbounds/maxpropounds</code> ($-1 \leq \text{integer} \leq 536870911$)	2
maximum number of propagation rounds during probing (-1 infinity)	

<code>heuristics/vbounds/minimprove</code> ($0 \leq \text{real} \leq 1$)	0.01
factor by which vbounds heuristic should at least improve the incumbent	
<code>heuristics/vbounds/minnodes</code> ($0 \leq \text{integer}$)	500
minimum number of nodes required to start the subproblem	
<code>heuristics/vbounds/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1106000
priority of heuristic <vbounds>	
<code>heuristics/veclending/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <veclending> (-1: no limit)	
<code>heuristics/veclending/maxdiveavgquot</code> ($0 \leq \text{real}$)	0
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{avglowerbound} - \text{lowerbound})$ where diving is performed (0.0: no limit)	
<code>heuristics/veclending/maxdiveavgquotnosol</code> ($0 \leq \text{real}$)	0
maximal AVGQUOT when no solution was found yet (0.0: no limit)	
<code>heuristics/veclending/maxdiveubquot</code> ($0 \leq \text{real} \leq 1$)	0.8
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{cutoffbound} - \text{lowerbound})$ where diving is performed (0.0: no limit)	
<code>heuristics/veclending/maxdiveubquotnosol</code> ($0 \leq \text{real} \leq 1$)	0.1
maximal UBQUOT when no solution was found yet (0.0: no limit)	
<code>heuristics/veclending/maxreldepth</code> ($0 \leq \text{real} \leq 1$)	1
maximal relative depth to start diving	
<code>heuristics/veclending/minreldepth</code> ($0 \leq \text{real} \leq 1$)	0
minimal relative depth to start diving	
<code>heuristics/veclending/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1003100
priority of heuristic <veclending>	
<code>heuristics/zirounding/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth level to call primal heuristic <zirounding> (-1: no limit)	
<code>heuristics/zirounding/maxroundingloops</code> ($-1 \leq \text{integer}$)	2
determines maximum number of rounding loops	
<code>heuristics/zirounding/minstopncalls</code> ($1 \leq \text{integer}$)	1000
determines the minimum number of calls before percentage-based deactivation of Zirounding is applied	
<code>heuristics/zirounding/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-500
priority of heuristic <zirounding>	
<code>heuristics/zirounding/stoppercentage</code> ($0 \leq \text{real} \leq 1$)	0.02
if percentage of found solutions falls below this parameter, Zirounding will be deactivated	
<code>heuristics/zirounding/stopziround</code> (boolean)	TRUE
flag to determine if Zirounding is deactivated after a certain percentage of unsuccessful calls	

Limits

<code>limits/absgap</code> ($0 \leq \text{real}$)	0
solving stops, if the absolute gap = $ \text{primalbound} - \text{dualbound} $ is below the given value	
<code>limits/bestsol</code> ($-1 \leq \text{integer}$)	-1
solving stops, if the given number of solution improvements were found (-1: no limit)	
<code>limits/gap</code> ($0 \leq \text{real}$)	0.1
solving stops, if the relative gap = $ \text{primal} - \text{dual} / \text{MIN}(\text{dual} , \text{primal})$ is below the given value	
<code>limits/maxorigsol</code> ($0 \leq \text{integer}$)	10
maximal number of solutions candidates to store in the solution storage of the original problem	
<code>limits/maxsol</code> ($1 \leq \text{integer}$)	100
maximal number of solutions to store in the solution storage	

<code>limits/memory</code> ($0 \leq \text{real}$)	∞
maximal memory usage in MB; reported memory usage is lower than real memory usage!	
<code>limits/nodes</code> ($-1 \leq \text{integer}$)	-1
maximal number of nodes to process (-1: no limit)	
<code>limits/restarts</code> ($-1 \leq \text{integer}$)	-1
solving stops, if the given number of restarts was triggered (-1: no limit)	
<code>limits/solutions</code> ($-1 \leq \text{integer}$)	-1
solving stops, if the given number of solutions were found (-1: no limit)	
<code>limits/stallnodes</code> ($-1 \leq \text{integer}$)	-1
solving stops, if the given number of nodes was processed since the last improvement of the primal solution value (-1: no limit)	
<code>limits/time</code> ($0 \leq \text{real}$)	1000
maximal time in seconds to run	

LP

<code>lp/initialalgorithm</code> (<code>character</code>)	s
LP algorithm for solving initial LP relaxations (automatic 's'implex, 'p'rimal simplex, 'd'ual simplex, 'b'arrier, barrier with 'c'rossover)	
<code>lp/pricing</code> (<code>character</code>)	1
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)	
<code>lp/resolvealgorithm</code> (<code>character</code>)	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)	
<code>lp/solvedepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth for solving LP at the nodes (-1: no depth limit)	
<code>lp/solvefreq</code> ($-1 \leq \text{integer}$)	1
frequency for solving LP at the nodes (-1: never; 0: only root LP)	

LP (advanced options)

<code>lp/checkfeas</code> (<code>boolean</code>)	TRUE
should LP solutions be checked, resolving LP when numerical troubles occur?	
<code>lp/checkstability</code> (<code>boolean</code>)	TRUE
should LP solver's return status be checked for stability?	
<code>lp/cleanupcols</code> (<code>boolean</code>)	FALSE
should new non-basic columns be removed after LP solving?	
<code>lp/cleanupcolsroot</code> (<code>boolean</code>)	FALSE
should new non-basic columns be removed after root LP solving?	
<code>lp/cleanuprows</code> (<code>boolean</code>)	TRUE
should new basic rows be removed after LP solving?	
<code>lp/cleanuprowsroot</code> (<code>boolean</code>)	TRUE
should new basic rows be removed after root LP solving?	
<code>lp/clearinitialprobinglp</code> (<code>boolean</code>)	TRUE
should lp state be cleared at the end of probing mode when lp was initially unsolved, e.g., when called right after presolving?	
<code>lp/colagelimit</code> ($-1 \leq \text{integer}$)	10
maximum age a dynamic column can reach before it is deleted from the LP (-1: don't delete columns due to aging)	
<code>lp/fastmip</code> ($0 \leq \text{integer} \leq 1$)	1

which FASTMIP setting of LP solver should be used? 0: off, 1: low

<code>lp/freesolvalbuffers</code> (boolean)	FALSE
should the buffers for storing LP solution values during diving be freed at end of diving?	
<code>lp/lexdualalgo</code> (boolean)	FALSE
should the lexicographic dual algorithm be used?	
<code>lp/lexdualbasic</code> (boolean)	FALSE
choose fractional basic variables in lexicographic dual algorithm?	
<code>lp/lexdualmaxrounds</code> ($-1 \leq \text{integer}$)	2
maximum number of rounds in the lexicographic dual algorithm (-1: unbounded)	
<code>lp/lexdualrootonly</code> (boolean)	TRUE
should the lexicographic dual algorithm be applied only at the root node	
<code>lp/lexdualstalling</code> (boolean)	TRUE
turn on the lex dual algorithm only when stalling?	
<code>lp/presolving</code> (boolean)	TRUE
should presolving of LP solver be used?	
<code>lp/resolveiterfac</code> ($-1 \leq \text{real}$)	-1
factor of average LP iterations that is used as LP iteration limit for LP resolve (-1: unlimited)	
<code>lp/resolveitermin</code> ($1 \leq \text{integer}$)	1000
minimum number of iterations that are allowed for LP resolve	
<code>lp/resolverestore</code> (boolean)	FALSE
should the LP be resolved to restore the state at start of diving (if FALSE we buffer the solution values)?	
<code>lp/rowagelimit</code> ($-1 \leq \text{integer}$)	10
maximum age a dynamic row can reach before it is deleted from the LP (-1: don't delete rows due to aging)	
<code>lp/rowrepswitch</code> ($0 \leq \text{real}$)	∞
simplex algorithm shall use row representation of the basis if number of rows divided by number of columns exceeds this value	
<code>lp/scaling</code> (boolean)	TRUE
should scaling of LP solver be used?	
<code>lp/threads</code> ($0 \leq \text{integer} \leq 64$)	0
number of threads used for solving the LP (0: automatic)	

Memory

<code>memory/savefac</code> ($0 \leq \text{real} \leq 1$)	0.8
fraction of maximal memory usage resulting in switch to memory saving mode	

Memory (advanced options)

<code>memory/arraygrowfac</code> ($1 \leq \text{real} \leq 10$)	1.2
memory growing factor for dynamically allocated arrays	
<code>memory/arraygrowinit</code> ($0 \leq \text{integer}$)	4
initial size of dynamically allocated arrays	
<code>memory/pathgrowfac</code> ($1 \leq \text{real} \leq 10$)	2
memory growing factor for path array	
<code>memory/pathgrowinit</code> ($0 \leq \text{integer}$)	256
initial size of path array	
<code>memory/treegrowfac</code> ($1 \leq \text{real} \leq 10$)	2
memory growing factor for tree array	
<code>memory/treegrowinit</code> ($0 \leq \text{integer}$)	65536
initial size of tree array	

Micellaneous

<code>misc/catchctrlc</code> (boolean)	TRUE
should the CTRL-C interrupt be caught by SCIP?	
<code>misc/improvingsols</code> (boolean)	FALSE
should only solutions be checked which improve the primal bound	
<code>misc/permutationseed</code> ($-1 \leq \text{integer}$)	-1
seed value for permuting the problem after the problem was transformed (-1: no permutation)	
<code>misc/resetstat</code> (boolean)	TRUE
should the statistics be reseted if the transformed problem is freed (in case of a benders decomposition this parameter should be set to FALSE)	
<code>misc/useconstable</code> (boolean)	TRUE
should a hashtable be used to map from constraint names to constraints?	
<code>misc/usesmalltables</code> (boolean)	FALSE
should smaller hashtables be used? yields better performance for small problems with about 100 variables	
<code>misc/usevartable</code> (boolean)	TRUE
should a hashtable be used to map from variable names to variables?	

Node Selection

<code>nodeselection/bfs/stdpriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	100000
priority of node selection rule <bfs> in standard mode	
<code>nodeselection/childsel</code> (character)	h
child selection rule ('d'own, 'u'p, 'p'seudo costs, 'i'nference, 'l'p value, 'r'oot LP value difference, 'h'ybrid inference/root LP value difference)	
<code>nodeselection/dfs/stdpriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	0
priority of node selection rule <dfs> in standard mode	
<code>nodeselection/estimate/bestnodefreq</code> ($0 \leq \text{integer}$)	10
frequency at which the best node instead of the best estimate is selected (0: never)	
<code>nodeselection/estimate/stdpriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	200000
priority of node selection rule <estimate> in standard mode	
<code>nodeselection/hybridestim/bestnodefreq</code> ($0 \leq \text{integer}$)	1000
frequency at which the best node instead of the hybrid best estimate / best bound is selected (0: never)	
<code>nodeselection/hybridestim/stdpriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	50000
priority of node selection rule <hybridestim> in standard mode	
<code>nodeselection/restartdfs/countonlyleaves</code> (boolean)	TRUE
count only leaf nodes (otherwise all nodes)?	
<code>nodeselection/restartdfs/selectbestfreq</code> ($0 \leq \text{integer}$)	100
frequency for selecting the best node instead of the deepest one	
<code>nodeselection/restartdfs/stdpriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	10000
priority of node selection rule <restartdfs> in standard mode	

Node Selection (advanced options)

<code>nodeselection/bfs/maxplungedepth</code> ($-1 \leq \text{integer}$)	-1
maximal plunging depth, before new best node is forced to be selected (-1 for dynamic setting)	
<code>nodeselection/bfs/maxplungequot</code> ($0 \leq \text{real}$)	0.25
maximal quotient $(\text{curlowerbound} - \text{lowerbound}) / (\text{cutoffbound} - \text{lowerbound})$ where plunging is performed	
<code>nodeselection/bfs/memsavepriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	0
priority of node selection rule <bfs> in memory saving mode	
<code>nodeselection/bfs/minplungedepth</code> ($-1 \leq \text{integer}$)	-1

minimal plunging depth, before new best node may be selected (-1 for dynamic setting)	
<code>nodeselection/dfs/memsavepriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	100000
priority of node selection rule <dfs> in memory saving mode	
<code>nodeselection/estimate/maxplungedepth</code> ($-1 \leq \text{integer}$)	-1
maximal plunging depth, before new best node is forced to be selected (-1 for dynamic setting)	
<code>nodeselection/estimate/maxplungequot</code> ($0 \leq \text{real}$)	0.25
maximal quotient (estimate - lowerbound)/(cutoffbound - lowerbound) where plunging is performed	
<code>nodeselection/estimate/memsavepriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	100
priority of node selection rule <estimate> in memory saving mode	
<code>nodeselection/estimate/minplungedepth</code> ($-1 \leq \text{integer}$)	-1
minimal plunging depth, before new best node may be selected (-1 for dynamic setting)	
<code>nodeselection/hybridestim/estimweight</code> ($0 \leq \text{real} \leq 1$)	0.1
weight of estimate value in node selection score (0: pure best bound search, 1: pure best estimate search)	
<code>nodeselection/hybridestim/maxplungedepth</code> ($-1 \leq \text{integer}$)	-1
maximal plunging depth, before new best node is forced to be selected (-1 for dynamic setting)	
<code>nodeselection/hybridestim/maxplungequot</code> ($0 \leq \text{real}$)	0.25
maximal quotient (estimate - lowerbound)/(cutoffbound - lowerbound) where plunging is performed	
<code>nodeselection/hybridestim/memsavepriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	50
priority of node selection rule <hybridestim> in memory saving mode	
<code>nodeselection/hybridestim/minplungedepth</code> ($-1 \leq \text{integer}$)	-1
minimal plunging depth, before new best node may be selected (-1 for dynamic setting)	
<code>nodeselection/restartdfs/memsavepriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	50000
priority of node selection rule <restartdfs> in memory saving mode	

Tolerances

<code>numerics/dualfeastol</code> ($10^{-17} \leq \text{real} \leq 0.001$)	10^{-9}
feasibility tolerance for reduced costs in LP solution	
<code>numerics/epsilon</code> ($10^{-20} \leq \text{real} \leq 0.001$)	10^{-9}
absolute values smaller than this are considered zero	
<code>numerics/feastol</code> ($10^{-17} \leq \text{real} \leq 0.001$)	10^{-6}
feasibility tolerance for constraints	
<code>numerics/sumepsilon</code> ($10^{-17} \leq \text{real} \leq 0.001$)	10^{-6}
absolute values of sums smaller than this are considered zero	

Tolerances (advanced options)

<code>numerics/barrierconvtol</code> ($10^{-17} \leq \text{real} \leq 0.001$)	10^{-10}
LP convergence tolerance used in barrier algorithm	
<code>numerics/boundstreps</code> ($10^{-17} \leq \text{real}$)	0.05
minimal relative improve for strengthening bounds	
<code>numerics/pseudocostdelta</code> ($0 \leq \text{real}$)	0.0001
minimal objective distance value to use for branching pseudo cost updates	
<code>numerics/pseudocosteps</code> ($10^{-17} \leq \text{real} \leq 1$)	0.1
minimal variable distance value to use for branching pseudo cost updates	

Presolving

<code>presolving/boundshift/maxrounds</code> ($-1 \leq \text{integer}$)	0
maximal number of presolving rounds the presolver participates in (-1: no limit)	
<code>presolving/dualfix/maxrounds</code> ($-1 \leq \text{integer}$)	-1

maximal number of presolving rounds the presolver participates in (-1: no limit)	
<code>presolving/implics/maxrounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the presolver participates in (-1: no limit)	
<code>presolving/inttobinary/maxrounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the presolver participates in (-1: no limit)	
<code>presolving/maxrestarts</code> ($-1 \leq \text{integer}$)	-1
maximal number of restarts (-1: unlimited)	
<code>presolving/maxrounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds (-1: unlimited, 0: off)	
<code>presolving/trivial/maxrounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the presolver participates in (-1: no limit)	
Presolving (advanced options)	
<code>presolving/abortfac</code> ($0 \leq \text{real} \leq 1$)	0.0001
abort presolve, if at most this fraction of the problem was changed in last presolve round	
<code>presolving/boundshift/delay</code> (boolean)	FALSE
should presolver be delayed, if other presolvers found reductions?	
<code>presolving/boundshift/flipping</code> (boolean)	TRUE
is flipping allowed (multiplying with -1)?	
<code>presolving/boundshift/integer</code> (boolean)	TRUE
shift only integer ranges?	
<code>presolving/boundshift/maxshift</code> ($0 \leq \text{integer}$)	∞
absolute value of maximum shift	
<code>presolving/boundshift/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	7900000
priority of presolver <boundshift>	
<code>presolving/donotaggr</code> (boolean)	FALSE
should aggregation of variables be forbidden?	
<code>presolving/donotmultaggr</code> (boolean)	FALSE
should multi-aggregation of variables be forbidden?	
<code>presolving/dualfix/delay</code> (boolean)	FALSE
should presolver be delayed, if other presolvers found reductions?	
<code>presolving/dualfix/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	8000000
priority of presolver <dualfix>	
<code>presolving/immrestartfac</code> ($0 \leq \text{real} \leq 1$)	0.2
fraction of integer variables that were fixed in the root node triggering an immediate restart with preprocessing	
<code>presolving/implics/delay</code> (boolean)	FALSE
should presolver be delayed, if other presolvers found reductions?	
<code>presolving/implics/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-10000
priority of presolver <implics>	
<code>presolving/inttobinary/delay</code> (boolean)	FALSE
should presolver be delayed, if other presolvers found reductions?	
<code>presolving/inttobinary/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	7000000
priority of presolver <inttobinary>	
<code>presolving/restartfac</code> ($0 \leq \text{real} \leq 1$)	0.05
fraction of integer variables that were fixed in the root node triggering a restart with preprocessing after root node evaluation	
<code>presolving/restartminred</code> ($0 \leq \text{real} \leq 1$)	0.1

minimal fraction of integer variables removed after restart to allow for an additional restart

<code>presolving/subrestartfac</code> ($0 \leq \text{real} \leq 1$)	1
fraction of integer variables that were globally fixed during the solving process triggering a restart with preprocessing	
<code>presolving/trivial/delay</code> (boolean)	FALSE
should presolver be delayed, if other presolvers found reductions?	
<code>presolving/trivial/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	9000000
priority of presolver <trivial>	

Domain Propagation

<code>propagating/abortoncutoff</code> (boolean)	TRUE
should propagation be aborted immediately? setting this to FALSE could help conflict analysis to produce more conflict constraints	
<code>propagating/maxrounds</code> ($-1 \leq \text{integer}$)	100
maximal number of propagation rounds per node (-1: unlimited)	
<code>propagating/maxroundsroot</code> ($-1 \leq \text{integer}$)	1000
maximal number of propagation rounds in the root node (-1: unlimited)	
<code>propagating/probing/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling propagator <probing> (-1: never, 0: only in root node)	
<code>propagating/probing/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the propagator participates in (-1: no limit)	
<code>propagating/probing/maxruns</code> ($-1 \leq \text{integer}$)	1
maximal number of runs, probing participates in (-1: no limit)	
<code>propagating/pseudoobj/freq</code> ($-1 \leq \text{integer}$)	1
frequency for calling propagator <pseudoobj> (-1: never, 0: only in root node)	
<code>propagating/pseudoobj/maxprerounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of presolving rounds the propagator participates in (-1: no limit)	
<code>propagating/redcost/continuous</code> (boolean)	FALSE
should reduced cost fixing be also applied to continuous variables?	
<code>propagating/redcost/freq</code> ($-1 \leq \text{integer}$)	1
frequency for calling propagator <redcost> (-1: never, 0: only in root node)	
<code>propagating/redcost/maxprerounds</code> ($-1 \leq \text{integer}$)	0
maximal number of presolving rounds the propagator participates in (-1: no limit)	
<code>propagating/rootredcost/freq</code> ($-1 \leq \text{integer}$)	1
frequency for calling propagator <rootredcost> (-1: never, 0: only in root node)	
<code>propagating/rootredcost/maxprerounds</code> ($-1 \leq \text{integer}$)	0
maximal number of presolving rounds the propagator participates in (-1: no limit)	
<code>propagating/vbounds/freq</code> ($-1 \leq \text{integer}$)	1
frequency for calling propagator <vbounds> (-1: never, 0: only in root node)	
<code>propagating/vbounds/maxprerounds</code> ($-1 \leq \text{integer}$)	0
maximal number of presolving rounds the propagator participates in (-1: no limit)	
<code>propagating/vbounds/usebdwidening</code> (boolean)	TRUE
should bound widening be used to initialize conflict analysis?	
Domain Propagation (advanced options)	
<code>propagating/probing/delay</code> (boolean)	TRUE
should propagator be delayed, if other propagators found reductions?	
<code>propagating/probing/maxfixings</code> ($0 \leq \text{integer}$)	25

maximal number of fixings found, until probing is interrupted (0: don't interrupt)	
<code>propagating/probing/maxsumuseless</code> ($0 \leq \text{integer}$)	0
maximal number of probings without fixings, until probing is aborted (0: don't abort)	
<code>propagating/probing/maxtotaluseless</code> ($0 \leq \text{integer}$)	50
maximal number of successive probings without fixings, bound changes, and implications, until probing is aborted (0: don't abort)	
<code>propagating/probing/maxuseless</code> ($0 \leq \text{integer}$)	1000
maximal number of successive probings without fixings, until probing is aborted (0: don't abort)	
<code>propagating/probing/presoldelay</code> (boolean)	TRUE
should presolver be delayed, if other propagators found reductions?	
<code>propagating/probing/presolpriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-100000
presolving priority of propagator <probing>	
<code>propagating/probing/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-100000
priority of propagator <probing>	
<code>propagating/probing/propounds</code> ($-1 \leq \text{integer}$)	-1
maximal number of propagation rounds in probing subproblems (-1: no limit, 0: auto)	
<code>propagating/pseudoobj/delay</code> (boolean)	FALSE
should propagator be delayed, if other propagators found reductions?	
<code>propagating/pseudoobj/maxcands</code> ($-1 \leq \text{integer}$)	100
maximal number of variables to look at in a single propagation round (-1: process all variables)	
<code>propagating/pseudoobj/presoldelay</code> (boolean)	TRUE
should presolver be delayed, if other propagators found reductions?	
<code>propagating/pseudoobj/presolpriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	6000000
presolving priority of propagator <pseudoobj>	
<code>propagating/pseudoobj/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	0
priority of propagator <pseudoobj>	
<code>propagating/pseudoobj/propcutoffbound</code> (boolean)	FALSE
propagate new cutoff bound directly globally	
<code>propagating/pseudoobj/propfullinroot</code> (boolean)	TRUE
do we want to propagate full if we are propagating the root node, despite the number of maxcand	
<code>propagating/redcost/delay</code> (boolean)	FALSE
should propagator be delayed, if other propagators found reductions?	
<code>propagating/redcost/presoldelay</code> (boolean)	TRUE
should presolver be delayed, if other propagators found reductions?	
<code>propagating/redcost/presolpriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	0
presolving priority of propagator <redcost>	
<code>propagating/redcost/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	10000000
priority of propagator <redcost>	
<code>propagating/rootredcost/delay</code> (boolean)	FALSE
should propagator be delayed, if other propagators found reductions?	
<code>propagating/rootredcost/presoldelay</code> (boolean)	TRUE
should presolver be delayed, if other propagators found reductions?	
<code>propagating/rootredcost/presolpriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	0
presolving priority of propagator <rootredcost>	
<code>propagating/rootredcost/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	1000000
priority of propagator <rootredcost>	

<code>propagating/vbounds/delay</code> (boolean)	FALSE
should propagator be delayed, if other propagators found reductions?	
<code>propagating/vbounds/presoldelay</code> (boolean)	TRUE
should presolver be delayed, if other propagators found reductions?	
<code>propagating/vbounds/presolpriority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	0
presolving priority of propagator <vbounds>	
<code>propagating/vbounds/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	2000000
priority of propagator <vbounds>	

Separation

<code>separating/cgmip/addviolationcons</code> (boolean)	TRUE
add constraint to subscip that only allows violated cuts?	
<code>separating/cgmip/addviolconshdlr</code> (boolean)	FALSE
add constraint handler to filter out violated cuts?	
<code>separating/cgmip/allowlocal</code> (boolean)	FALSE
allow to generate local cuts?	
<code>separating/cgmip/cmirownbounds</code> (boolean)	FALSE
tell CMIR-generator which bounds to used in rounding?	
<code>separating/cgmip/conshdlrusenorm</code> (boolean)	TRUE
should the violation constraint handler use the norm of a cut to check for feasibility?	
<code>separating/cgmip/contconvert</code> (boolean)	TRUE
convert some integral variables to be continuous to reduce the size of the sub-MIP?	
<code>separating/cgmip/contconvfrac</code> ($0 \leq \text{real} \leq 1$)	0.5
fraction of integral variables converted to be continuous (if contconvert)	
<code>separating/cgmip/contconvmin</code> ($-1 \leq \text{integer}$)	100
minimum number of integral variables before some are converted to be continuous	
<code>separating/cgmip/dynamiccuts</code> (boolean)	TRUE
should generated cuts be removed from the LP if they are no longer tight?	
<code>separating/cgmip/earlyterm</code> (boolean)	TRUE
terminate separation if a violated (but possibly sub-optimal) cut has been found?	
<code>separating/cgmip/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling separator <cgmip> (-1: never, 0: only in root node)	
<code>separating/cgmip/maxdepth</code> ($-1 \leq \text{integer}$)	-1
maximal depth at which the separator is applied (-1: unlimited)	
<code>separating/cgmip/maxrounds</code> ($-1 \leq \text{integer}$)	5
maximal number of cgmip separation rounds per node (-1: unlimited)	
<code>separating/cgmip/maxroundsroot</code> ($-1 \leq \text{integer}$)	50
maximal number of cgmip separation rounds in the root node (-1: unlimited)	
<code>separating/cgmip/maxrowage</code> ($-1 \leq \text{integer}$)	-1
maximal age of rows to consider if onlyactiverows is false	
<code>separating/cgmip/nodelimit</code> ($-1 \leq \text{integer}$)	10000
node limit for sub-MIP (-1: unlimited)	
<code>separating/cgmip/objlone</code> (boolean)	FALSE
should the objective of the sub-MIP minimize the l1-norm of the multipliers?	
<code>separating/cgmip/onlyactiverows</code> (boolean)	FALSE
use only active rows to generate cuts?	
<code>separating/cgmip/onlyintvars</code> (boolean)	FALSE

generate cuts for problems with only integer variables?	
<code>separating/cgmip/onlyrankone</code> (boolean) whether only rank 1 inequalities should be separated	FALSE
<code>separating/cgmip/primalseparation</code> (boolean) only separate cuts that are tight for the best feasible solution?	TRUE
<code>separating/cgmip/usecmir</code> (boolean) use CMIR-generator (otherwise add cut directly)?	TRUE
<code>separating/cgmip/usecutpool</code> (boolean) use cutpool to store CG-cuts even if the are not efficient?	TRUE
<code>separating/cliq/freq</code> ($-1 \leq$ integer) frequency for calling separator <cliq> (-1: never, 0: only in root node)	0
<code>separating/cliq/maxsepacuts</code> ($-1 \leq$ integer) maximal number of cliq cuts separated per separation round (-1: no limit)	10
<code>separating/closecuts/freq</code> ($-1 \leq$ integer) frequency for calling separator <closecuts> (-1: never, 0: only in root node)	-1
<code>separating/cmir/dynamiccuts</code> (boolean) should generated cuts be removed from the LP if they are no longer tight?	TRUE
<code>separating/cmir/freq</code> ($-1 \leq$ integer) frequency for calling separator <cmir> (-1: never, 0: only in root node)	0
<code>separating/cmir/maxrounds</code> ($-1 \leq$ integer) maximal number of cmir separation rounds per node (-1: unlimited)	3
<code>separating/cmir/maxroundsroot</code> ($-1 \leq$ integer) maximal number of cmir separation rounds in the root node (-1: unlimited)	10
<code>separating/cmir/maxsepacuts</code> ($0 \leq$ integer) maximal number of cmir cuts separated per separation round	100
<code>separating/cmir/maxsepacutsroot</code> ($0 \leq$ integer) maximal number of cmir cuts separated per separation round in the root node	500
<code>separating/flowcover/dynamiccuts</code> (boolean) should generated cuts be removed from the LP if they are no longer tight?	TRUE
<code>separating/flowcover/freq</code> ($-1 \leq$ integer) frequency for calling separator <flowcover> (-1: never, 0: only in root node)	0
<code>separating/flowcover/maxrounds</code> ($-1 \leq$ integer) maximal number of separation rounds per node (-1: unlimited)	5
<code>separating/flowcover/maxroundsroot</code> ($-1 \leq$ integer) maximal number of separation rounds in the root node (-1: unlimited)	10
<code>separating/flowcover/maxsepacuts</code> ($0 \leq$ integer) maximal number of flow cover cuts separated per separation round	100
<code>separating/flowcover/maxsepacutsroot</code> ($0 \leq$ integer) maximal number of flow cover cuts separated per separation round in the root	200
<code>separating/gomory/dynamiccuts</code> (boolean) should generated cuts be removed from the LP if they are no longer tight?	TRUE
<code>separating/gomory/freq</code> ($-1 \leq$ integer) frequency for calling separator <gomory> (-1: never, 0: only in root node)	0
<code>separating/gomory/maxrounds</code> ($-1 \leq$ integer) maximal number of gomory separation rounds per node (-1: unlimited)	5
<code>separating/gomory/maxroundsroot</code> ($-1 \leq$ integer)	10

maximal number of gomory separation rounds in the root node (-1: unlimited)	
<code>separating/gomory/maxsepacuts</code> ($0 \leq \text{integer}$)	50
maximal number of gomory cuts separated per separation round	
<code>separating/gomory/maxsepacutsroot</code> ($0 \leq \text{integer}$)	50
maximal number of gomory cuts separated per separation round in the root node	
<code>separating/impliedbounds/freq</code> ($-1 \leq \text{integer}$)	0
frequency for calling separator <code><impliedbounds></code> (-1: never, 0: only in root node)	
<code>separating/intobj/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling separator <code><intobj></code> (-1: never, 0: only in root node)	
<code>separating/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	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)	
<code>separating/maxcuts</code> ($0 \leq \text{integer}$)	100
maximal number of cuts separated per separation round (0: disable local separation)	
<code>separating/maxcutsroot</code> ($0 \leq \text{integer}$)	2000
maximal number of separated cuts at the root node (0: disable root node separation)	
<code>separating/maxrounds</code> ($-1 \leq \text{integer}$)	5
maximal number of separation rounds per node (-1: unlimited)	
<code>separating/maxroundsroot</code> ($-1 \leq \text{integer}$)	-1
maximal number of separation rounds in the root node (-1: unlimited)	
<code>separating/maxstallrounds</code> ($-1 \leq \text{integer}$)	5
maximal number of consecutive separation rounds without objective or integrality improvement (-1: no additional restriction)	
<code>separating/mcf/dynamiccuts</code> (boolean)	TRUE
should generated cuts be removed from the LP if they are no longer tight?	
<code>separating/mcf/freq</code> ($-1 \leq \text{integer}$)	0
frequency for calling separator <code><mcf></code> (-1: never, 0: only in root node)	
<code>separating/mcf/maxsepacuts</code> ($-1 \leq \text{integer}$)	100
maximal number of mcf cuts separated per separation round	
<code>separating/mcf/maxsepacutsroot</code> ($-1 \leq \text{integer}$)	200
maximal number of mcf cuts separated per separation round in the root node – default separation	
<code>separating/minefficacy</code> ($0 \leq \text{real}$)	0.05
minimal efficacy for a cut to enter the LP	
<code>separating/minefficacyroot</code> ($0 \leq \text{real}$)	0.01
minimal efficacy for a cut to enter the LP in the root node	
<code>separating/minortho</code> ($0 \leq \text{real} \leq 1$)	0.5
minimal orthogonality for a cut to enter the LP	
<code>separating/minorthoroot</code> ($0 \leq \text{real} \leq 1$)	0.5
minimal orthogonality for a cut to enter the LP in the root node	
<code>separating/oddcycle/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling separator <code><oddcycle></code> (-1: never, 0: only in root node)	
<code>separating/oddcycle/liftoddcycles</code> (boolean)	FALSE
should odd cycle cuts be lifted?	
<code>separating/oddcycle/maxrounds</code> ($-1 \leq \text{integer}$)	10
maximal number of oddcycle separation rounds per node (-1: unlimited)	
<code>separating/oddcycle/maxroundsroot</code> ($-1 \leq \text{integer}$)	10
maximal number of oddcycle separation rounds in the root node (-1: unlimited)	

<code>separating/oddcycle/maxsepacuts</code> ($0 \leq \text{integer}$)	5000
maximal number of oddcycle cuts separated per separation round	
<code>separating/oddcycle/maxsepacutsroot</code> ($0 \leq \text{integer}$)	5000
maximal number of oddcycle cuts separated per separation round in the root node	
<code>separating/oddcycle/useclassical</code> (boolean)	TRUE
should classical search method by Groetschel, Lovasz, Schrijver be used? Otherwise use levelgraph method by Hoffman, Padberg.	
<code>separating/poolfreq</code> ($-1 \leq \text{integer}$)	0
separation frequency for the global cut pool (-1: disable global cut pool, 0: only separate pool at the root)	
<code>separating/rapidlearning/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling separator <rapidlearning> (-1: never, 0: only in root node)	
<code>separating/strongcg/dynamiccuts</code> (boolean)	TRUE
should generated cuts be removed from the LP if they are no longer tight?	
<code>separating/strongcg/freq</code> ($-1 \leq \text{integer}$)	0
frequency for calling separator <strongcg> (-1: never, 0: only in root node)	
<code>separating/strongcg/maxrounds</code> ($-1 \leq \text{integer}$)	5
maximal number of strong CG separation rounds per node (-1: unlimited)	
<code>separating/strongcg/maxroundsroot</code> ($-1 \leq \text{integer}$)	20
maximal number of strong CG separation rounds in the root node (-1: unlimited)	
<code>separating/strongcg/maxsepacuts</code> ($0 \leq \text{integer}$)	50
maximal number of strong CG cuts separated per separation round	
<code>separating/strongcg/maxsepacutsroot</code> ($0 \leq \text{integer}$)	500
maximal number of strong CG cuts separated per separation round in the root node	
<code>separating/zerohalf/dynamiccuts</code> (boolean)	TRUE
should generated cuts be removed from the LP if they are no longer tight?	
<code>separating/zerohalf/freq</code> ($-1 \leq \text{integer}$)	-1
frequency for calling separator <zerohalf> (-1: never, 0: only in root node)	
<code>separating/zerohalf/maxrounds</code> ($-1 \leq \text{integer}$)	5
maximal number of zerohalf separation rounds per node (-1: unlimited)	
<code>separating/zerohalf/maxroundsroot</code> ($-1 \leq \text{integer}$)	10
maximal number of zerohalf separation rounds in the root node (-1: unlimited)	
<code>separating/zerohalf/maxsepacuts</code> ($0 \leq \text{integer}$)	50
maximal number of 0,1/2-cuts separated per separation round	
<code>separating/zerohalf/maxsepacutsroot</code> ($0 \leq \text{integer}$)	500
maximal number of 0,1/2-cuts separated per separation round in the root node	
<code>separating/zerohalf/preprocessing/decomposeproblem</code> (boolean)	FALSE
should problem be decomposed into subproblems (if possible) before applying preprocessing?	
<code>separating/zerohalf/preprocessing/delta</code> ($0 \leq \text{real} \leq 1$)	0.5
value of delta parameter used in preprocessing method 'd'	
<code>separating/zerohalf/preprocessing/ppmethods</code> (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/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
separating/zerohalf/separating/auxip/penaltyfactor (0 ≤ real ≤ 1) 0.001
penalty factor used with objective function 'p' of auxiliary IP
separating/zerohalf/separating/auxip/settingsfile (string) -
optional settings file of the auxiliary IP (-: none)
separating/zerohalf/separating/auxip/sollimit (-1 ≤ integer) -1
limits/solutions setting of the 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/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 ≤ 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

Separation (advanced options)
separating/cgmip/delay (boolean) FALSE
should separator be delayed, if other separators found cuts?
separating/cgmip/maxbounddist (0 ≤ real ≤ 1) 0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound
for applying separator <cgmip> (0.0: only on current best node, 1.0: on all nodes)
separating/cgmip/memorylimit (0 ≤ real) ∞
memory limit for sub-MIP
separating/cgmip/objweight (0 ≤ real) 0.001

```

weight used for the row combination coefficient in the sub-MIP objective	
<code>separating/cgmip/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$) priority of separator <cgmip>	-1000
<code>separating/cgmip/timelimit</code> ($0 \leq \text{real}$) time limit for sub-MIP	∞
<code>separating/cliq/trackbackfreq</code> ($0 \leq \text{integer}$) frequency for premature backtracking up to tree level 1 (0: no backtracking)	1000
<code>separating/cliq/cliqdensity</code> ($0 \leq \text{real} \leq 1$) minimal density of cliques to use a dense clique table	0.05
<code>separating/cliq/cliqtablemem</code> ($0 \leq \text{real} \leq 2.09715 \cdot 10^6$) maximal memory size of dense clique table (in kb)	20000
<code>separating/cliq/delay</code> (boolean) should separator be delayed, if other separators found cuts?	FALSE
<code>separating/cliq/maxbounddist</code> ($0 \leq \text{real} \leq 1$) maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <cliq> (0.0: only on current best node, 1.0: on all nodes)	0
<code>separating/cliq/maxtreenodes</code> ($-1 \leq \text{integer}$) maximal number of nodes in branch and bound tree (-1: no limit)	10000
<code>separating/cliq/maxzeroextensions</code> ($-1 \leq \text{integer}$) maximal number of zero-valued variables extending the clique (-1: no limit)	1000
<code>separating/cliq/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$) priority of separator <cliq>	-5000
<code>separating/cliq/scaleval</code> ($1 \leq \text{real}$) factor for scaling weights	1000
<code>separating/closecuts/closethres</code> ($-1 \leq \text{integer}$) threshold on number of generated cuts below which the ordinary separation is started	50
<code>separating/closecuts/delay</code> (boolean) should separator be delayed, if other separators found cuts?	FALSE
<code>separating/closecuts/inclobjcutoff</code> (boolean) include an objective cutoff when computing the relative interior?	FALSE
<code>separating/closecuts/maxbounddist</code> ($0 \leq \text{real} \leq 1$) maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <closecuts> (0.0: only on current best node, 1.0: on all nodes)	1
<code>separating/closecuts/maxunsuccessful</code> ($-1 \leq \text{integer}$) turn off separation in current node after unsuccessful calls (-1 never turn off)	0
<code>separating/closecuts/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$) priority of separator <closecuts>	1000000
<code>separating/closecuts/recomputerelint</code> (boolean) recompute relative interior point in each separation call?	FALSE
<code>separating/closecuts/relintnormtype</code> (character) type of norm to use when computing relative interior: 'o'ne norm, 's'upremum norm	o
<code>separating/closecuts/sepacombvalue</code> ($0 \leq \text{real} \leq 1$) convex combination value for close cuts	0.3
<code>separating/closecuts/separelint</code> (boolean) generate close cuts w.r.t. relative interior point (best solution otherwise)?	TRUE
<code>separating/closecuts/separootonly</code> (boolean) generate close cuts in the root only?	TRUE

<code>separating/cmir/aggrtol</code> ($0 \leq \text{real}$)	0.1
tolerance for bound distances used to select continuous variable in current aggregated constraint to be eliminated	
<code>separating/cmir/delay</code> (boolean)	FALSE
should separator be delayed, if other separators found cuts?	
<code>separating/cmir/densityoffset</code> ($0 \leq \text{integer}$)	100
additional number of variables allowed in row on top of density	
<code>separating/cmir/densityscore</code> ($0 \leq \text{real}$)	0.0001
weight of row density in the aggregation scoring of the rows	
<code>separating/cmir/fixintegralrhs</code> (boolean)	TRUE
should an additional variable be complemented if $f_0 = 0$?	
<code>separating/cmir/maxaggdensity</code> ($0 \leq \text{real} \leq 1$)	0.2
maximal density of aggregated row	
<code>separating/cmir/maxaggrs</code> ($0 \leq \text{integer}$)	3
maximal number of aggregations for each row per separation round	
<code>separating/cmir/maxaggrsroot</code> ($0 \leq \text{integer}$)	6
maximal number of aggregations for each row per separation round in the root node	
<code>separating/cmir/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <cmir> (0.0: only on current best node, 1.0: on all nodes)	
<code>separating/cmir/maxconts</code> ($0 \leq \text{integer}$)	10
maximal number of active continuous variables in aggregated row	
<code>separating/cmir/maxcontsroot</code> ($0 \leq \text{integer}$)	10
maximal number of active continuous variables in aggregated row in the root node	
<code>separating/cmir/maxfails</code> ($-1 \leq \text{integer}$)	20
maximal number of consecutive unsuccessful aggregation tries (-1: unlimited)	
<code>separating/cmir/maxfailsroot</code> ($-1 \leq \text{integer}$)	100
maximal number of consecutive unsuccessful aggregation tries in the root node (-1: unlimited)	
<code>separating/cmir/maxrowdensity</code> ($0 \leq \text{real} \leq 1$)	0.05
maximal density of row to be used in aggregation	
<code>separating/cmir/maxrowfac</code> ($0 \leq \text{real}$)	10000
maximal row aggregation factor	
<code>separating/cmir/maxslack</code> ($0 \leq \text{real}$)	0
maximal slack of rows to be used in aggregation	
<code>separating/cmir/maxslackroot</code> ($0 \leq \text{real}$)	0.1
maximal slack of rows to be used in aggregation in the root node	
<code>separating/cmir/maxtestdelta</code> ($-1 \leq \text{integer}$)	-1
maximal number of different deltas to try (-1: unlimited)	
<code>separating/cmir/maxtries</code> ($-1 \leq \text{integer}$)	100
maximal number of rows to start aggregation with per separation round (-1: unlimited)	
<code>separating/cmir/maxtriesroot</code> ($-1 \leq \text{integer}$)	-1
maximal number of rows to start aggregation with per separation round in the root node (-1: unlimited)	
<code>separating/cmir/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-3000
priority of separator <cmir>	
<code>separating/cmir/slackscore</code> ($0 \leq \text{real}$)	0.001
weight of slack in the aggregation scoring of the rows	
<code>separating/cmir/trynegscaling</code> (boolean)	TRUE
should negative values also be tested in scaling?	

<code>separating/cutagelimit</code> ($-1 \leq \text{integer}$)	100
maximum age a cut can reach before it is deleted from the global cut pool, or -1 to keep all cuts	
<code>separating/efficacynorm</code> (character)	e
row norm to use for efficacy calculation ('e'ucclidean, 'm'aximum, 's'um, 'd'iscrete)	
<code>separating/flowcover/delay</code> (boolean)	FALSE
should separator be delayed, if other separators found cuts?	
<code>separating/flowcover/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <flowcover> (0.0: only on current best node, 1.0: on all nodes)	
<code>separating/flowcover/maxfails</code> ($-1 \leq \text{integer}$)	50
maximal number of consecutive fails to generate a cut per separation round (-1: unlimited)	
<code>separating/flowcover/maxfailsroot</code> ($-1 \leq \text{integer}$)	100
maximal number of consecutive fails to generate a cut per separation round in the root (-1: unlimited)	
<code>separating/flowcover/maxrowdensity</code> ($0 \leq \text{real} \leq 1$)	1
maximal density of row to separate flow cover cuts for	
<code>separating/flowcover/maxslack</code> ($0 \leq \text{real}$)	∞
maximal slack of rows to separate flow cover cuts for	
<code>separating/flowcover/maxslackroot</code> ($0 \leq \text{real}$)	∞
maximal slack of rows to separate flow cover cuts for in the root	
<code>separating/flowcover/maxtestdelta</code> ($0 \leq \text{integer}$)	10
cut generation heuristic: maximal number of different deltas to try	
<code>separating/flowcover/maxtries</code> ($-1 \leq \text{integer}$)	100
maximal number of rows to separate flow cover cuts for per separation round (-1: unlimited)	
<code>separating/flowcover/maxtriesroot</code> ($-1 \leq \text{integer}$)	-1
maximal number of rows to separate flow cover cuts for per separation round in the root (-1: unlimited)	
<code>separating/flowcover/multbyminusone</code> (boolean)	TRUE
should flow cover cuts be separated for 0-1 single node flow set with reversed arcs in addition?	
<code>separating/flowcover/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-4000
priority of separator <flowcover>	
<code>separating/flowcover/slackscore</code> ($0 \leq \text{real}$)	0.001
weight of slack in the scoring of the rows	
<code>separating/gomory/delay</code> (boolean)	FALSE
should separator be delayed, if other separators found cuts?	
<code>separating/gomory/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <gomory> (0.0: only on current best node, 1.0: on all nodes)	
<code>separating/gomory/maxweightrange</code> ($1 \leq \text{real}$)	10000
maximal valid range $\max(\text{weights})/\min(\text{weights})$ of row weights	
<code>separating/gomory/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1000
priority of separator <gomory>	
<code>separating/impliedbounds/delay</code> (boolean)	FALSE
should separator be delayed, if other separators found cuts?	
<code>separating/impliedbounds/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <impliedbounds> (0.0: only on current best node, 1.0: on all nodes)	
<code>separating/impliedbounds/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-50
priority of separator <impliedbounds>	

<code>separating/intobj/delay</code> (boolean)	FALSE
should separator be delayed, if other separators found cuts?	
<code>separating/intobj/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <code><intobj></code> (0.0: only on current best node, 1.0: on all nodes)	
<code>separating/intobj/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-100
priority of separator <code><intobj></code>	
<code>separating/maxaddrounds</code> ($-1 \leq \text{integer}$)	1
maximal additional number of separation rounds in subsequent price-and-cut loops (-1: no additional restriction)	
<code>separating/maxroundsrootsubrun</code> ($-1 \leq \text{integer}$)	1
maximal number of separation rounds in the root node of a subsequent run (-1: unlimited)	
<code>separating/maxruns</code> ($-1 \leq \text{integer}$)	-1
maximal number of runs for which separation is enabled (-1: unlimited)	
<code>separating/mcf/checkcutshoreconnectivity</code> (boolean)	TRUE
should we separate only if the cuts shores are connected?	
<code>separating/mcf/delay</code> (boolean)	FALSE
should separator be delayed, if other separators found cuts?	
<code>separating/mcf/fixintegralrhs</code> (boolean)	TRUE
should an additional variable be complemented if $f_0 = 0$?	
<code>separating/mcf/maxarcinconsistencyratio</code> ($0 \leq \text{real}$)	0.5
maximum inconsistency ratio of arcs not to be deleted	
<code>separating/mcf/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <code><mcf></code> (0.0: only on current best node, 1.0: on all nodes)	
<code>separating/mcf/maxinconsistencyratio</code> ($0 \leq \text{real}$)	0.02
maximum inconsistency ratio for separation at all	
<code>separating/mcf/maxtestdelta</code> ($-1 \leq \text{integer}$)	20
maximal number of different deltas to try (-1: unlimited) – default separation	
<code>separating/mcf/maxweightrange</code> ($1 \leq \text{real}$)	10^6
maximal valid range $\max(\text{weights})/\min(\text{weights})$ of row weights	
<code>separating/mcf/modeltype</code> ($0 \leq \text{integer} \leq 2$)	0
model type of network (0: auto, 1:directed, 2:undirected)	
<code>separating/mcf/nclusters</code> ($2 \leq \text{integer} \leq 32$)	5
number of clusters to generate in the shrunken network – default separation	
<code>separating/mcf/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-10000
priority of separator <code><mcf></code>	
<code>separating/mcf/separateflowcutset</code> (boolean)	TRUE
should we separate flowcutset inequalities on the network cuts?	
<code>separating/mcf/separateknapsack</code> (boolean)	TRUE
should we separate knapsack cover inequalities on the network cuts?	
<code>separating/mcf/separatesinglenodecuts</code> (boolean)	TRUE
should we separate inequalities based on single-node cuts?	
<code>separating/mcf/trynegscaling</code> (boolean)	FALSE
should negative values also be tested in scaling?	
<code>separating/objparalfac</code> ($0 \leq \text{real}$)	0.0001
factor to scale objective parallelism of cut in separation score calculation	
<code>separating/oddcycle/addselfarcs</code> (boolean)	TRUE

add links between a variable and its negated

<code>separating/oddcycle/allowmultiplecuts</code> (boolean)	TRUE
even if a variable is already covered by a cut, still allow another cut to cover it too	
<code>separating/oddcycle/delay</code> (boolean)	FALSE
should separator be delayed, if other separators found cuts?	
<code>separating/oddcycle/includetriangles</code> (boolean)	TRUE
separate triangles found as 3-cycles or repaired larger cycles	
<code>separating/oddcycle/lpliftcoef</code> (boolean)	FALSE
choose lifting candidate by $\text{coef} * \text{lpvalue}$ or only by coef	
<code>separating/oddcycle/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <code><oddcycle></code> (0.0: only on current best node, 1.0: on all nodes)	
<code>separating/oddcycle/maxcutslevel</code> ($0 \leq \text{integer}$)	50
maximal number of <code>oddcycle</code> cuts generated in every level of the level graph	
<code>separating/oddcycle/maxcutsroot</code> ($0 \leq \text{integer}$)	1
maximal number of <code>oddcycle</code> cuts generated per chosen variable as root of the level graph	
<code>separating/oddcycle/maxnlevels</code> ($0 \leq \text{integer}$)	20
maximal number of levels in level graph	
<code>separating/oddcycle/maxpernodeslevel</code> ($0 \leq \text{integer} \leq 100$)	100
percentage of nodes allowed in the same level of the level graph [0-100]	
<code>separating/oddcycle/maxreference</code> ($0 \leq \text{integer}$)	0
minimal weight on an edge (in level graph or bipartite graph)	
<code>separating/oddcycle/multiplecuts</code> (boolean)	FALSE
even if a variable is already covered by a cut, still try it as start node for a cycle search	
<code>separating/oddcycle/offsetnodeslevel</code> ($0 \leq \text{integer}$)	10
offset of nodes allowed in the same level of the level graph (additional to the percentage of <code>levelnodes</code>)	
<code>separating/oddcycle/offsettestvars</code> ($0 \leq \text{integer}$)	100
offset of variables to try the chosen method on (additional to the percentage of <code>testvars</code>)	
<code>separating/oddcycle/percenttestvars</code> ($0 \leq \text{integer} \leq 100$)	0
percentage of variables to try the chosen method on [0-100]	
<code>separating/oddcycle/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-15000
priority of separator <code><oddcycle></code>	
<code>separating/oddcycle/recalcliftcoef</code> (boolean)	TRUE
calculate lifting coefficient of every candidate in every step (or only if its chosen)	
<code>separating/oddcycle/repaircycles</code> (boolean)	TRUE
try to repair violated cycles with double appearance of a variable	
<code>separating/oddcycle/scalingfactor</code> ($1 \leq \text{integer}$)	1000
factor for scaling of the arc-weights	
<code>separating/oddcycle/sortrootneighbors</code> (boolean)	TRUE
sort level of the root neighbors by fractionality (<code>maxfrac</code>)	
<code>separating/oddcycle/sortswitch</code> ($0 \leq \text{integer} \leq 4$)	3
use sorted variable array (<code>unsorted(0),maxlp(1),minlp(2),maxfrac(3),minfrac(4)</code>)	
<code>separating/orthofac</code> ($0 \leq \text{real}$)	1
factor to scale orthogonality of cut in separation score calculation (0.0 to disable orthogonality calculation)	
<code>separating/orthofunc</code> (character)	e
function used for calc. scalar prod. in orthogonality test ('e'ucclidean, 'd'iscrete)	

<code>separating/rapidlearning/applybdchgs</code> (boolean)	TRUE
should the found global bound deductions be applied in the original SCIP?	
<code>separating/rapidlearning/applyconflicts</code> (boolean)	TRUE
should the found conflicts be applied in the original SCIP?	
<code>separating/rapidlearning/applyinffervals</code> (boolean)	TRUE
should the inference values be used as initialization in the original SCIP?	
<code>separating/rapidlearning/applyprimalsol</code> (boolean)	TRUE
should the incumbent solution be copied to the original SCIP?	
<code>separating/rapidlearning/applysolved</code> (boolean)	TRUE
should a solved status be copied to the original SCIP?	
<code>separating/rapidlearning/contvars</code> (boolean)	FALSE
should rapid learning be applied when there are continuous variables?	
<code>separating/rapidlearning/contvarsquot</code> ($0 \leq \text{real} \leq 1$)	0.3
maximal portion of continuous variables to apply rapid learning	
<code>separating/rapidlearning/copycuts</code> (boolean)	TRUE
should all active cuts from cutpool be copied to constraints in subproblem?	
<code>separating/rapidlearning/delay</code> (boolean)	FALSE
should separator be delayed, if other separators found cuts?	
<code>separating/rapidlearning/lpiterquot</code> ($0 \leq \text{real}$)	0.2
maximal fraction of LP iterations compared to node LP iterations	
<code>separating/rapidlearning/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	1
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <rapidlearning> (0.0: only on current best node, 1.0: on all nodes)	
<code>separating/rapidlearning/maxnconss</code> ($0 \leq \text{integer}$)	10000
maximum problem size (constraints) for which rapid learning will be called	
<code>separating/rapidlearning/maxnodes</code> ($0 \leq \text{integer}$)	5000
maximum number of nodes considered in rapid learning run	
<code>separating/rapidlearning/maxnvars</code> ($0 \leq \text{integer}$)	10000
maximum problem size (variables) for which rapid learning will be called	
<code>separating/rapidlearning/minnodes</code> ($0 \leq \text{integer}$)	500
minimum number of nodes considered in rapid learning run	
<code>separating/rapidlearning/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-1200000
priority of separator <rapidlearning>	
<code>separating/rapidlearning/reducedinfer</code> (boolean)	FALSE
should the inference values only be used when rapidlearning found other reductions?	
<code>separating/strongcg/delay</code> (boolean)	FALSE
should separator be delayed, if other separators found cuts?	
<code>separating/strongcg/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <strongcg> (0.0: only on current best node, 1.0: on all nodes)	
<code>separating/strongcg/maxweightrange</code> ($1 \leq \text{real}$)	10000
maximal valid range $\max(\text{weights})/\min(\text{weights})$ of row weights	
<code>separating/strongcg/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-2000
priority of separator <strongcg>	
<code>separating/zerohalf/delay</code> (boolean)	FALSE
should separator be delayed, if other separators found cuts?	
<code>separating/zerohalf/ignoreprevzhcuts</code> (boolean)	FALSE

should zerohalf cuts found in previous callbacks ignored?	
<code>separating/zerohalf/maxbounddist</code> ($0 \leq \text{real} \leq 1$)	0
maximal relative distance from current node's dual bound to primal bound compared to best node's dual bound for applying separator <code><zerohalf></code> (0.0: only on current best node, 1.0: on all nodes)	
<code>separating/zerohalf/maxcutsfound</code> ($0 \leq \text{integer}$)	100
maximal number of 0,1/2-cuts determined per separation round # (this includes separated but inefficacious cuts)	
<code>separating/zerohalf/maxcutsfoundroot</code> ($0 \leq \text{integer}$)	1000
maximal number of 0,1/2-cuts determined per separation round in the root node # (this includes separated but inefficacious cuts)	
<code>separating/zerohalf/maxdepth</code> ($-1 \leq \text{integer}$)	-1
separating cuts only if depth \leq maxdepth (-1: unlimited)	
<code>separating/zerohalf/maxncalls</code> ($-1 \leq \text{integer}$)	-1
maximal number of calls (-1: unlimited)	
<code>separating/zerohalf/maxtestdelta</code> ($-1 \leq \text{integer}$)	10
maximal number of different deltas to try for cmir (-1: unlimited, 0: delta=1)	
<code>separating/zerohalf/onlyorigrows</code> (boolean)	FALSE
should only original LP rows be considered (i.e. ignore previously added LP rows)?	
<code>separating/zerohalf/priority</code> ($-536870912 \leq \text{integer} \leq 536870911$)	-6000
priority of separator <code><zerohalf></code>	
<code>separating/zerohalf/relaxcontvars</code> (boolean)	FALSE
should continuous variables be relaxed by adding variable bounds?	
<code>separating/zerohalf/scalefraccoeffs</code> (boolean)	TRUE
should rows be scaled to make fractional coefficients integer?	
<code>separating/zerohalf/trynegscaling</code> (boolean)	TRUE
should negative values also be tested in scaling for cmir?	
<code>separating/zerohalf/usezhcutpool</code> (boolean)	TRUE
should zerohalf cuts be filtered using a cutpool?	
Timing	
<code>timing/clocktype</code> ($1 \leq \text{integer} \leq 2$)	1
default clock type (1: CPU user seconds, 2: wall clock time)	
<code>timing/enabled</code> (boolean)	TRUE
is timing enabled?	
<code>timing/reading</code> (boolean)	FALSE
belongs reading time to solving time?	

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 Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 427–444. Springer, 2012.
- [6] Stefan Vigerske. *Decomposition of Multistage Stochastic Programs and a Constraint Integer Programming Approach to Mixed-Integer Nonlinear Programming*. PhD thesis, Humboldt Universität zu Berlin, 2012. Submitted.
- [7] 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>.
- [8] Kati Wolter. Implementation of cutting plane separators for mixed integer programs. Diploma thesis, Technische Universität Berlin, 2006.
- [9] 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>.