

Package ‘scip’

May 9, 2026

Title Interface to the SCIP Optimization Suite

Version 1.10.0-3

Description

Provides an R interface to SCIP (Solving Constraint Integer Programs), a framework for mixed-integer programming (MIP), mixed-integer nonlinear programming (MINLP), and constraint integer programming (2025, <[doi:10.48550/arXiv.2511.18580](https://doi.org/10.48550/arXiv.2511.18580)>). Supports linear, quadratic, SOS, indicator, and knapsack constraints with continuous, binary, and integer variables. Includes a one-shot solver interface and a model-building API for incremental problem construction.

License Apache License (>= 2)

URL <https://bnaras.github.io/scip/>, <https://scipopt.org/>

BugReports <https://github.com/bnaras/scip/issues>

Depends R (>= 4.0)

Imports methods, Matrix

Suggests tinytest, slam, knitr, rmarkdown

VignetteBuilder knitr

SystemRequirements CMake (>= 3.11), GNU make, C++17

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation yes

Author Balasubramanian Narasimhan [aut, cre] (ORCID:
<<https://orcid.org/0000-0001-5852-7639>>),
SCIP Optimization Suite Authors [cph] (SCIP, SoPlex, PaPILO libraries)

Maintainer Balasubramanian Narasimhan <naras@stanford.edu>

Repository CRAN

Date/Publication 2026-04-10 08:20:02 UTC

Contents

print.scip_control	2
scip_add_indicator_cons	3
scip_add_linear_cons	3
scip_add_quadratic_cons	4
scip_add_sos1_cons	5
scip_add_sos2_cons	5
scip_add_var	6
scip_add_vars	6
scip_control	7
scip_get_info	10
scip_get_nsols	10
scip_get_objval	11
scip_get_sol	11
scip_get_solution	12
scip_get_status	12
scip_model	13
scip_model_free	13
scip_optimize	14
scip_set_objective_sense	14
scip_set_param	15
scip_solve	15

Index	17
--------------	-----------

print.scip_control	<i>Print method for scip_control objects</i>
--------------------	--

Description

Print method for scip_control objects

Usage

```
## S3 method for class 'scip_control'
print(x, ...)
```

Arguments

x	A scip_control object.
...	Ignored.

Value

Invisible x.

 scip_add_indicator_cons

Add an indicator constraint to a SCIP model

Description

If $\text{binvar} = 1$ then $\text{sum}(\text{coefs} * x[\text{vars}]) \leq \text{rhs}$.

Usage

```
scip_add_indicator_cons(model, binvar, vars, coefs, rhs, name = NULL)
```

Arguments

model	A SCIP model.
binvar	Integer; 1-based index of the binary indicator variable.
vars	Integer vector; 1-based variable indices.
coefs	Numeric vector; coefficients.
rhs	Numeric; right-hand side.
name	Character; constraint name.

Value

Integer; 1-based constraint index.

 scip_add_linear_cons *Add a linear constraint to a SCIP model*

Description

Adds $\text{lhs} \leq \text{sum}(\text{coefs} * x[\text{vars}]) \leq \text{rhs}$.

Usage

```
scip_add_linear_cons(model, vars, coefs, lhs = -Inf, rhs = Inf, name = NULL)
```

Arguments

model	A SCIP model.
vars	Integer vector; 1-based variable indices.
coefs	Numeric vector; coefficients (same length as vars).
lhs	Numeric; left-hand side. Default $-\text{Inf}$.
rhs	Numeric; right-hand side. Default Inf .
name	Character; constraint name. Default auto-generated.

Value

Integer; 1-based constraint index.

scip_add_quadratic_cons

Add a quadratic constraint to a SCIP model

Description

Adds $lhs \leq linexpr + quadexpr \leq rhs$ where $quadexpr = \sum(quadcoefs[k] * x[quadvars1[k]] * x[quadvars2[k]])$.

Usage

```
scip_add_quadratic_cons(
    model,
    linvars = integer(0),
    lincoefs = double(0),
    quadvars1 = integer(0),
    quadvars2 = integer(0),
    quadcoefs = double(0),
    lhs = -Inf,
    rhs = Inf,
    name = NULL
)
```

Arguments

model	A SCIP model.
linvars	Integer vector; 1-based variable indices for linear part.
lincoefs	Numeric vector; linear coefficients.
quadvars1, quadvars2	Integer vectors; 1-based variable indices for quadratic terms.
quadcoefs	Numeric vector; quadratic coefficients.
lhs, rhs	Numeric; constraint bounds.
name	Character; constraint name.

Value

Integer; 1-based constraint index.

scip_add_sos1_cons *Add a SOS1 constraint to a SCIP model*

Description

At most one variable in the set can be nonzero.

Usage

```
scip_add_sos1_cons(model, vars, weights = NULL, name = NULL)
```

Arguments

model	A SCIP model.
vars	Integer vector; 1-based variable indices.
weights	Numeric vector; weights determining branching order.
name	Character; constraint name.

Value

Integer; 1-based constraint index.

scip_add_sos2_cons *Add a SOS2 constraint to a SCIP model*

Description

At most two adjacent variables in the set can be nonzero.

Usage

```
scip_add_sos2_cons(model, vars, weights = NULL, name = NULL)
```

Arguments

model	A SCIP model.
vars	Integer vector; 1-based variable indices.
weights	Numeric vector; weights determining adjacency order.
name	Character; constraint name.

Value

Integer; 1-based constraint index.

scip_add_var	<i>Add a variable to a SCIP model</i>
--------------	---------------------------------------

Description

Add a variable to a SCIP model

Usage

```
scip_add_var(model, obj, lb = 0, ub = Inf, vtype = "C", name = NULL)
```

Arguments

model	A SCIP model (external pointer from scip_model).
obj	Numeric; objective coefficient.
lb	Numeric; lower bound. Default 0.
ub	Numeric; upper bound. Default Inf.
vtype	Character; variable type: "C" (continuous), "B" (binary), or "I" (integer). Default "C".
name	Character; variable name. Default auto-generated.

Value

Integer; 1-based variable index.

scip_add_vars	<i>Add multiple variables to a SCIP model</i>
---------------	---

Description

Add multiple variables to a SCIP model

Usage

```
scip_add_vars(model, obj, lb = 0, ub = Inf, vtype = "C", names = NULL)
```

Arguments

model	A SCIP model.
obj	Numeric vector; objective coefficients.
lb	Numeric; lower bounds (scalar or vector). Default 0.
ub	Numeric; upper bounds (scalar or vector). Default Inf.
vtype	Character; variable types (scalar or vector). Default "C".
names	Character vector; variable names. Default auto-generated.

Value

Integer; 1-based index of first variable added.

scip_control	<i>SCIP solver control parameters</i>
--------------	---------------------------------------

Description

Create a list of control parameters for the SCIP solver. Parameters are organized into logical groups: output, limits, tolerances, presolving, LP, branching, and heuristics. Any SCIP parameter can also be set directly using its native path via

Usage

```
scip_control(
  verbose = TRUE,
  verbosity_level = 3L,
  display_freq = 100L,
  time_limit = Inf,
  node_limit = -1L,
  stall_node_limit = -1L,
  sol_limit = -1L,
  best_sol_limit = -1L,
  mem_limit = Inf,
  restart_limit = -1L,
  gap_limit = 0,
  abs_gap_limit = 0,
  feastol = 1e-06,
  dualfeastol = 1e-07,
  epsilon = 1e-09,
  presolving = TRUE,
  presolve_rounds = -1L,
  lp_threads = 1L,
  lp_iteration_limit = -1L,
  lp_scaling = TRUE,
  branching_score = "p",
  heuristics_emphasis = "default",
  threads = 1L,
  ...
)
```

Arguments

verbose	Logical; print solver output. Default TRUE.
verbosity_level	Integer 0–5; verbosity detail. Default 3.

display_freq	Integer; node display frequency. Default 100.
time_limit	Numeric; time limit in seconds. Default Inf.
node_limit	Integer; max nodes. Default -1L.
stall_node_limit	Integer; stall nodes. Default -1L.
sol_limit	Integer; solution limit. Default -1L.
best_sol_limit	Integer; improving solution limit. Default -1L.
mem_limit	Numeric; memory limit in MB. Default Inf.
restart_limit	Integer; restart limit. Default -1L.
gap_limit	Numeric; relative MIP gap. Default 0.
abs_gap_limit	Numeric; absolute gap. Default 0.
feastol	Numeric; feasibility tolerance. Default 1e-6.
dualfeastol	Numeric; dual feasibility tolerance. Default 1e-7.
epsilon	Numeric; zero tolerance. Default 1e-9.
presolving	Logical; enable presolving. Default TRUE.
presolve_rounds	Integer; presolve rounds. Default -1L.
lp_threads	Integer; LP solver threads. Default 1L.
lp_iteration_limit	Integer; LP iteration limit. Default -1L.
lp_scaling	Logical; LP scaling. Default TRUE.
branching_score	Character; score function. Default "p".
heuristics_emphasis	Character; heuristic emphasis. Default "default".
threads	Integer; parallel solving threads. Default 1L. See Parallel section for caveats.
...	Additional SCIP parameters as name-value pairs, using SCIP's native hierarchical parameter paths (e.g., "lp/fastmip" = 1, "conflict/enable" = FALSE). See the SCIP parameter documentation for the full list.

Value

A named list of class "scip_control" with components:

verbose Logical; verbosity flag.

scip_params Named list; all parameters as SCIP native paths.

Output

verbose Logical; print solver output. Default TRUE.

verbosity_level Integer 0-5; verbosity detail (0 = none, 3 = normal, 5 = full). Default 3. Ignored if verbose = FALSE.

display_freq Integer; display a status line every this many nodes (-1 = never). Default 100.

Termination Limits

time_limit Numeric; maximum solving time in seconds. Default Inf (no limit).
 node_limit Integer; maximum number of branch-and-bound nodes. Default -1L (no limit).
 stall_node_limit Integer; nodes without improvement before stopping. Default -1L (no limit).
 sol_limit Integer; stop after finding this many feasible solutions. Default -1L (no limit).
 best_sol_limit Integer; stop after this many improving solutions. Default -1L (no limit).
 mem_limit Numeric; memory limit in MB. Default Inf (no limit).
 restart_limit Integer; maximum restarts. Default -1L (no limit).

Tolerances

gap_limit Numeric; relative MIP gap tolerance; solver stops when the gap falls below this value.
 Default 0 (prove optimality).
 abs_gap_limit Numeric; absolute gap between primal and dual bound. Default 0.
 feastol Numeric; feasibility tolerance for LP constraints. Default 1e-6.
 dualfeastol Numeric; dual feasibility tolerance. Default 1e-7.
 epsilon Numeric; absolute values below this are treated as zero. Default 1e-9.

Presolving

presolving Logical; enable presolving. Default TRUE.
 presolve_rounds Integer; maximum presolving rounds (-1 = unlimited). Default -1L.

LP

lp_threads Integer; number of threads for LP solver. Default 1L.
 lp_iteration_limit Integer; LP iteration limit per solve (-1 = no limit). Default -1L.
 lp_scaling Logical; enable LP scaling. Default TRUE.

Branching

branching_score Character; branching score function: "s" (sum), "p" (product), "q" (quotient).
 Default "p".

Heuristics

heuristics_emphasis Character; heuristic emphasis setting: "default", "aggressive", "fast",
 or "off". Default "default".

Parallel

threads Integer; number of threads for concurrent solving. Default 1L. Note: concurrent solving may require a SCIP build compiled with parallel support (e.g., PARASCIP=true); not all installations provide this.

See Also

[scip_solve](#), [scip_set_param](#)

Examples

```
## Quick solve with 60-second time limit
ctrl <- scip_control(time_limit = 60)

## Quiet solve with 1% gap tolerance
ctrl <- scip_control(verbose = FALSE, gap_limit = 0.01)

## Aggressive heuristics, no presolving
ctrl <- scip_control(heuristics_emphasis = "aggressive", presolving = FALSE)

## Pass a native SCIP parameter directly
ctrl <- scip_control("conflict/enable" = FALSE, "separating/maxrounds" = 5L)
```

scip_get_info	<i>Get solver information</i>
---------------	-------------------------------

Description

Get solver information

Usage

```
scip_get_info(model)
```

Arguments

model A SCIP model (after [scip_optimize](#)).

Value

A list with solve_time, nodes, iterations, gap, sol_count.

scip_get_nsols	<i>Get number of solutions found</i>
----------------	--------------------------------------

Description

Get number of solutions found

Usage

```
scip_get_nsols(model)
```

Arguments

model A SCIP model (after `scip_optimize`).

Value

Integer.

scip_get_objval *Get objective value of best solution*

Description

Get objective value of best solution

Usage

```
scip_get_objval(model)
```

Arguments

model A SCIP model (after `scip_optimize`).

Value

Numeric; objective value, or NA if no solution.

scip_get_sol *Get the k-th solution from the solution pool*

Description

Get the k-th solution from the solution pool

Usage

```
scip_get_sol(model, k)
```

Arguments

model A SCIP model.

k Integer; 1-based solution index (1 = best).

Value

A list with `objval` and `x`.

scip_get_solution *Get the best solution*

Description

Get the best solution

Usage

```
scip_get_solution(model)
```

Arguments

model A SCIP model (after `scip_optimize`).

Value

A list with objval and x.

scip_get_status *Get solver status*

Description

Get solver status

Usage

```
scip_get_status(model)
```

Arguments

model A SCIP model (after `scip_optimize`).

Value

Character; status string (e.g., "optimal", "infeasible").

scip_model	<i>Create a SCIP model</i>
------------	----------------------------

Description

Creates a new SCIP optimization model for incremental problem construction.

Usage

```
scip_model(name = "scip_model")
```

Arguments

name Character; problem name. Default "scip_model".

Value

An external pointer representing the SCIP model.

scip_model_free	<i>Free a SCIP model</i>
-----------------	--------------------------

Description

Explicitly frees the SCIP model and all associated memory. The model is also freed automatically when garbage collected.

Usage

```
scip_model_free(model)
```

Arguments

model A SCIP model.

Value

Invisible NULL.

scip_optimize	<i>Solve a SCIP model</i>
---------------	---------------------------

Description

Solve a SCIP model

Usage

```
scip_optimize(model)
```

Arguments

model	A SCIP model.
-------	---------------

Value

Invisible NULL. Use [scip_get_status](#) and [scip_get_solution](#) to retrieve results.

scip_set_objective_sense	<i>Set objective sense</i>
--------------------------	----------------------------

Description

Set objective sense

Usage

```
scip_set_objective_sense(model, sense = "minimize")
```

Arguments

model	A SCIP model.
sense	Character; "minimize" (default) or "maximize".

Value

Invisible NULL.

scip_set_param	<i>Set a SCIP parameter</i>
----------------	-----------------------------

Description

Set a SCIP parameter

Usage

```
scip_set_param(model, name, value)
```

Arguments

model	A SCIP model.
name	Character; SCIP parameter name (e.g., "limits/time").
value	The parameter value (type is auto-detected by SCIP).

Value

Invisible NULL.

scip_solve	<i>Solve a linear or mixed-integer program using SCIP</i>
------------	---

Description

One-shot interface to the SCIP solver. Formulates and solves:

$$\min_x \text{obj}'x$$

subject to constraint rows defined by A, b, sense, with variable types vtype and bounds lb, ub.

Usage

```
scip_solve(obj, A, b, sense, vtype = "C", lb = 0, ub = Inf, control = list())
```

Arguments

obj	Numeric vector of length n; objective coefficients.
A	Constraint matrix (m x n). Can be a dense matrix, dgMatrix, or simple_triplet_matrix.
b	Numeric vector of length m; constraint right-hand side.
sense	Character vector of length m; constraint sense. Each element must be "<=", ">=", or "==".

vtype	Character; variable types. Either a single value applied to all variables, or a vector of length n. Values: "C" (continuous), "B" (binary), "I" (integer). Default "C".
lb	Numeric; lower bounds for variables. Single value or vector of length n. Default \emptyset .
ub	Numeric; upper bounds for variables. Single value or vector of length n. Default Inf.
control	A list of solver parameters, typically from scip_control .

Value

A named list with components:

status Character; solver status (e.g., "optimal", "infeasible", "unbounded").

objval Numeric; optimal objective value (or NA if no solution).

x Numeric vector; primal solution (or NULL if no solution).

sol_count Integer; number of solutions found.

gap Numeric; relative optimality gap.

info List with additional solver information (solve_time, iterations, nodes).

Index

print.scip_control, 2

scip_add_indicator_cons, 3

scip_add_linear_cons, 3

scip_add_quadratic_cons, 4

scip_add_sos1_cons, 5

scip_add_sos2_cons, 5

scip_add_var, 6

scip_add_vars, 6

scip_control, 7, 16

scip_get_info, 10

scip_get_nsols, 10

scip_get_objval, 11

scip_get_sol, 11

scip_get_solution, 12, 14

scip_get_status, 12, 14

scip_model, 6, 13

scip_model_free, 13

scip_optimize, 10–12, 14

scip_set_objective_sense, 14

scip_set_param, 10, 15

scip_solve, 10, 15