Package 'ROI.plugin.lpsolve'

January 20, 2025

Version 1.0-2

Title 'lp_solve' Plugin for the 'R' Optimization Infrastructure

Description Enhances the 'R' Optimization Infrastructure ('ROI') package
with the 'lp_solve' solver.

Imports stats, methods, utils, ROI (>= 0.3-0), lpSolveAPI (>=
5.5.2.0-1)

Suggests slam

License GPL-3

URL https://roigrp.gitlab.io,

https://gitlab.com/roigrp/solver/ROI.plugin.lpsolve

NeedsCompilation no

Author Florian Schwendinger [aut, cre]

Maintainer Florian Schwendinger <FlorianSchwendinger@gmx.at>

Repository CRAN

Date/Publication 2023-07-07 11:50:05 UTC

Contents

1

---

control   *Control Variables*

Description

The control variables are all optional, but can in some cases be used to improve the solving time and the solutions.

Arguments

basis	an optional named list used to set the initial basis of the LP, e.g., R> control <- list(basis=list(basis = c(1, 2, 3), + nonbasic = TRUE, default = TRUE)). The elements of basis are passed to the **lpSolveAPI** function set.basis as arguments.

- basis a numeric vector giving the indices of the basis.
- nonbasic an optional logical, if TRUE the nonbasic variables included in the basis as well.
- default an optional logical, if TRUE the default basis is used and the arguments basis and nonbasic are ignored.

branch.mode	an optional list used to set the branch and bound mode, e.g., R> control <- list(branch.mode=list(columns=c(1, 2, 3), + modes=c("ceiling", "auto", "floor")) The elements of branch.mode are passed to **lpSolveAPI** function set.branch.mode as arguments.

- columns a vector of integer giving the column indices for which the mode is set.
- modes a character vector giving the modes of the columns specified in columns.

branch.weights	an optional numeric vector giving the weights for the decision variables. The length of the branch.weights must be equal to length of the objective function.
verbose	a character string (for more information see lp.control).
anti.degen	a character vector (for more information see lp.control).
basis.crash	a character string (for more information see lp.control).
bb.depthlimit	a integer giving the maximum branch-and-bound depth (for more information see lp.control).
bb.floorfirst	a character string (for more information see lp.control).
bb.rule	a character vector giving the branch-and-bound rule (for more information see lp.control).
break.at.first	a logical controlling whether the branch-and-bound algorithm should be stopped at the first solution or the branch-and-bound algorithm continuous until an optimal solution is found (for more information see lp.control).
break.at.value	a numeric, if given the branch-and-bound algorithm stops when the objective function becomes smaller than the specified value. (for more information see lp.control).

Example-1 3

epslevel	a character string giving the type of thresholds (for more information see `lp.control`).
epsb	a numeric giving the tolerance for the right-hand-side (for more information see `lp.control`).
epsd	a numeric (for more information see `lp.control`).
epsel	a numeric (for more information see `lp.control`).
epsint	a numeric (for more information see `lp.control`).
epsperturb	a numeric (for more information see `lp.control`).
epspivot	a numeric (for more information see `lp.control`).
improve	a character vector (for more information see `lp.control`).
infinite	a numeric (for more information see `lp.control`).
maxpivot	a integer (for more information see `lp.control`).
mip.gap	a numeric vector (for more information see `lp.control`).
negrange	a numeric (for more information see `lp.control`).
obj.in.bas	a logical (for more information see `lp.control`).
pivoting	a character vector (for more information see `lp.control`).
presolve	a character vector (for more information see `lp.control`).
scalelimit	a numeric (for more information see `lp.control`).
scaling	a character vector (for more information see `lp.control`).
simplextype	a character vector which an take one of the following values c("primal"), c("dual"), c("primal", "dual") or c("dual", "primal") (for more information see `lp.control`).
timeout	a integer giving the number of seconds till a timeout occurs (for more information see `lp.control`).

Example-1 *Linear Problem 1*

Description

$$maximize\ \ 2x_1 + 4x_2 + 3x_3$$

$$subject\ to:$$

$$3x_1 + 4x_2 + 2x_3 \leq 60$$

$$2x_1 + x_2 + 2x_3 \leq 40$$

$$x_1 + 3x_2 + 2x_3 \leq 80$$

$$x_1, x_2, x_3 \geq 0$$

Examples

```
library(ROI)
mat <- matrix(c(3, 4, 2,
                2, 1, 2,
                1, 3, 2), nrow=3, byrow=TRUE)
x <- OP(objective = c(2, 4, 3),
        constraints = L_constraint(L = mat,
                                   dir = c("<=", "<=", "<="),
                                   rhs = c(60, 40, 80)),
        maximum = TRUE)


opt <- ROI_solve(x, solver = "lpsolve")
opt
## Optimal solution found.
## The objective value is: 7.666667e+01
solution(opt)
## [1]  0.000000  6.666667 16.666667
```

---

read.lp   *Read Optimization Problem*

Description

Read a optimization problem from a file.

Usage

```
read.lp(file, type=c("lp", "mps", "freemps"))
```

Arguments

file	a character giving the name of the file the optimization problem is read from.
type	a character giving the name of the file format the optimization problem is stored in.

Details

The optimization problems can be read from the three file formats "lp", "mps" and "freemps". Where it seems important to note that the "lp" format refers to lpsolves native file format (https://lpsolve.sourceforge.net/5.5/lp-format.htm) and not to the CPLEX LP format.

Examples

```
## Not run:
op <- read.lp("optimization_problem.lp")
sol <- ROI_solve(op)
solution(sol)

## End(Not run)
```

---

write.lp	*Write Optimization Problem*

Description

Write a optimization problem to a file.

Usage

```
write.lp(x, file, type=c("lp", "mps", "freemps"))
```

Arguments

x	an object of type OP.
file	a character giving the name of the file the optimization problem is written to.
type	a character giving the name of the file format used to store the optimization problem.

Details

The optimization problems can be written to the three file formats "lp", "mps" and "freemps". Where it seems important to note that the "lp" format refers to lpsolves native file format (https://lpsolve.sourceforge.net/5.5/lp-format.htm) and not to the CPLEX LP format.

Examples

```
## Not run:
mat <- matrix(c(3, 4, 2,
                2, 1, 2,
                1, 3, 2), nrow=3, byrow=TRUE)
x <- OP(objective = c(2, 4, 3),
        constraints = L_constraint(L = mat,
                                   dir = c("<=", "<=", "<="),
                                   rhs = c(60, 40, 80)),
        bounds = V_bound(ui = seq_len(3), ub = c(1000, Inf, 1000), nobj = 3),
        maximum = TRUE)
write.lp(x, "optimization_problem.lp")

## End(Not run)
```

Index