Package 'lpsymphony'

November 12, 2026

Title Symphony integer linear programming solver in R

Version 1.40.0

Description

This package was derived from Rsymphony_0.1-17 from CRAN. These packages provide an R interface to SYMPHONY, an open-source linear programming solver written in C++. The main difference between this package and Rsymphony is that it includes the solver source code (SYMPHONY version 5.6), while Rsymphony expects to find header and library files on the users' system. Thus the intention of lpsymphony is to provide an easy to install interface to SYMPHONY. For Windows, precompiled DLLs are included in this package.

Maintainer Vladislav Kim <Vladislav.Kim@embl.de>

License EPL

Depends R (>= 3.0.0)

Enhances slam

Suggests BiocStyle, knitr, testthat

URL http://R-Forge.R-project.org/projects/rsymphony,
https://projects.coin-or.org/SYMPHONY,
http://www.coin-or.org/download/source/SYMPHONY/

BugReports https://github.com/Huber-group-EMBL/lpsymphony/issues

biocViews Infrastructure, ThirdPartyClient

VignetteBuilder knitr

NeedsCompilation yes

SystemRequirements GNU make

git_url https://git.bioconductor.org/packages/lpsymphony

git_branch RELEASE_3_23

git_last_commit 94a28c3

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Vladislav Kim [aut, cre],
Ted Ralphs [ctb],
Menal Guzelsoy [ctb],
Ashutosh Mahajan [ctb],
Reinhard Harter [ctb],
Kurt Hornik [ctb],
Cyrille Szymanski [ctb],
Stefan Theussl [ctb],
Mike Smith [ctb] (ORCID: <https://orcid.org/0000-0002-7800-3848>)

Contents

lpsymphony_solve_LP 2

Index **5**

`lpsymphony_solve_LP` *COIN-OR SYMPHONY Linear and Mixed Integer Programming Solver*

Description

High level R interface to the COIN-OR SYMPHONY solver for linear as well as mixed integer linear programming problems (MILPs).

Usage

```
lpsymphony_solve_LP(obj, mat, dir, rhs, bounds = NULL, types = NULL,
                    max = FALSE, verbosity = -2, time_limit = -1,
                    node_limit = -1, gap_limit = -1, first_feasible = FALSE,
                    write_lp = FALSE, write_mps = FALSE)
```

Arguments

`obj`	a vector with the objective coefficients
`mat`	a vector or a matrix of the constraint coefficients
`dir`	a character vector with the directions of the constraints. Each element must be one of `"<"`, `"<="`, `">"`, `">="`, `"=="` or `"!="`.
`rhs`	the right hand side of the constraints
`bounds`	`NULL` (default) or a list with elements `upper` and `lower` containing the indices and corresponding bounds of the objective variables. The default for each variable is a bound between 0 and `Inf`.
`types`	a character vector giving the types of the objective variables, with `"C"`, `"I"`, and `"B"` corresponding to continuous, integer, and binary, respectively, or `NULL` (default), taken as all-continuous. Recycled as needed.

max	a logical giving the direction of the optimization. TRUE means that the objective is to maximize the objective function, FALSE (default) means to minimize it.
verbosity	an integer defining the level of verbosity, -2 (default) means no output.
time_limit	an integer defining the time limit in seconds, -1 (default) means no time limit.
node_limit	an integer defining the limit in number of iterations, -1 (default) means no node limit.
gap_limit	when the gap between the lower and the upper bound reaches this point, the solution process will stop and the best solution found to that point will be returned, -1 (default) means no gap limit.
first_feasible	a logical defining if the solution process should stop after the first feasible solution has been found, FALSE (default) means that the solution process does not stop after the first feasible solution has been found.
write_lp	a logical value indicating if an LP representation of the problem should be written for debugging purposes, FALSE (default) means no LP file is written.
write_mps	a logical value indicating if an MPS representation of the problem should be written for debugging purposes, FALSE (default) means no MPS file is written.

Details

SYMPHONY is an open source solver for solving mixed integer linear programs (MILPs). The current version can be found at https://projects.coin-or.org/SYMPHONY. Package **lpsymphony** uses the C interface of the callable library provided by SYMPHONY, and supplies a high level solver function in R using the low level C interface.

Value

A list containing the optimal solution, with the following components.

solution	the vector of optimal coefficients
objval	the value of the objective function at the optimum
status	an integer with status information about the solution returned: 0 if the optimal solution was found, a non-zero value otherwise.

Author(s)

Reinhard Harter, Kurt Hornik and Stefan Theussl

References

SYMPHONY development home page (https://projects.coin-or.org/SYMPHONY/wiki).

See Also

lp in package **lpSolve**; Rglpk_solve_LP in package **Rglpk**.

Examples

```
## Simple linear program.
## maximize:   2 x_1 + 4 x_2 + 3 x_3
## subject to: 3 x_1 + 4 x_2 + 2 x_3 <= 60
##             2 x_1 +   x_2 +   x_3 <= 40
##               x_1 + 3 x_2 + 2 x_3 <= 80
##               x_1, x_2, x_3 are non-negative real numbers

obj <- c(2, 4, 3)
mat <- matrix(c(3, 2, 1, 4, 1, 3, 2, 2, 2), nrow = 3)
dir <- c("<=", "<=", "<=")
rhs <- c(60, 40, 80)
max <- TRUE

lpsymphony_solve_LP(obj, mat, dir, rhs, max = max)

## Simple mixed integer linear program.
## maximize:    3 x_1 + 1 x_2 + 3 x_3
## subject to: -1 x_1 + 2 x_2 +   x_3 <= 4
##                      4 x_2 - 3 x_3 <= 2
##                x_1 - 3 x_2 + 2 x_3 <= 3
##                x_1, x_3 are non-negative integers
##                x_2 is a non-negative real number

obj <- c(3, 1, 3)
mat <- matrix(c(-1, 0, 1, 2, 4, -3, 1, -3, 2), nrow = 3)
dir <- c("<=", "<=", "<=")
rhs <- c(4, 2, 3)
max <- TRUE
types <- c("I", "C", "I")

lpsymphony_solve_LP(obj, mat, dir, rhs, types = types, max = max)

## Same as before but with bounds replaced by
## -Inf <  x_1 <= 4
##    0 <= x_2 <= 100
##    2 <= x_3 <  Inf

bounds <- list(lower = list(ind = c(1L, 3L), val = c(-Inf, 2)),
               upper = list(ind = c(1L, 2L), val = c(4, 100)))

lpsymphony_solve_LP(obj, mat, dir, rhs, types = types, max = max,
                    bounds = bounds)
```

Index

∗ **optimize**
lpsymphony_solve_LP, 2

lp, *3*
lpsymphony_solve_LP, 2

Rglpk_solve_LP, *3*