Package 'Rcplex'

May 7, 2026

Version 0.3-8

Title R Interface to CPLEX

Description R interface to CPLEX solvers for linear, quadratic, and (linear and quadratic) mixed integer programs. Support for quadratically constrained programming is available. See the file ``INSTALL" for details on how to install the Rcplex package in Linux/Unix-like and Windows systems. Support for sparse matrices is provided by an S3-style class ``simple_triplet_matrix" from package slam and by objects from the Matrix package class hierarchy.

LazyLoad yes

Depends R (>= 2.6.0), slam

Imports methods

Enhances Matrix

SystemRequirements IBM ILOG CPLEX libraries and headers

License LGPL (>= 2.0)

URL https://R-Forge.R-project.org/projects/rcplex

NeedsCompilation yes

Author Hector Corrada Bravo [aut],
Kurt Hornik [ctb] (ORCID: <https://orcid.org/0000-0003-4198-9911>),
Stefan Theussl [aut, cre] (ORCID:
<https://orcid.org/0000-0002-6523-4620>)

Maintainer Stefan Theussl <Stefan.Theussl@R-project.org>

Repository CRAN

Date/Publication 2025-04-20 11:20:02 UTC

Contents

Rcplex *Solve optimization problem with CPLEX*

Description

Interface to CPLEX solvers for linear quadratic and (linear or quadratic) mixed-integer programs. The general statement of the problem is

$$\min \frac{1}{2} x'Qx + c'x$$

$$\text{s.t} Ax \leq b$$

$$lb \leq x \leq ub$$

If `Q==NULL` then the problem is linear, if any value of the `vtype` argument is "B" or "I" then the problem is a mixed-integer program. The `control` argument is used to set CPLEX's many parameters. See details. The `objsense` determines if the problem is a maximization or minimization problem. The `sense` argument is used to set the constraint directions.

Usage

```
Rcplex(cvec, Amat, bvec, Qmat = NULL,
       lb = 0, ub = Inf, control = list(),
       objsense = c("min", "max"), sense = "L", vtype = NULL, n = 1)
```

Arguments

`cvec`	The linear coefficient of the objective function
`Amat`	The constraint matrix (requires `ncol(Amat)==length(cvec)`)
`bvec`	The constraints right-hand side (requires `length(bvec)==nrow(Amat)`)
`Qmat`	The quadratic coefficient of the objective function. If `NULL` the problem is linear. If not `NULL`, it must be a symmetric positive semidefinite matrix of size `length(cvec)` by `length(cvec)`. Default `NULL`
`lb`	Lower bound on the problem variables. If `length(lb)==1` then `lb` is the lower bound of all variables. Otherwise, `length(lb)==length(cvec)`. Set `lb=-Inf` to have no lower bound. Default 0.
`ub`	Upper bound on the problem variables. See `lb` for further details. Default `Inf`.
`control`	A list of CPLEX parameters. See *Details*
`objsense`	Either `"max"` or `"min"`, determines the optimization direction. Default `"min"`
`sense`	The direction of the inequality in each constraint. If `length(sense)==1` then the same value is taken for each constraint. Can be one of `"L"` (less than or equal), `"G"` (reater than or equal) or `"E"` (equal). Requires `length(sense)==length(bvec)`. Default `"L"`.

`vtype` Determines the type of each problem variable. Can be one of `"C"` (continuous), `"I"` (integer) or `"B"` (binary). If `length(vtype)==1` the same value is taken for all variables. Otherwise, requires `length(vtype)==length(ctype)`. Default `"C"`.

`n` Determines the maximal number of solutions the solver should return in case of an MIP with more than one solution at optimum. If CPLEX should search for "all" solutions then `n` has to be set to `NA`. In CPLEX this is also called populating the solution pool. The parameters `solnpoolagap`, `solnpoolgap`, and `solnpoolintensity` influence the search for multiple solutions (see also the `control` argument below for details). Available from CPLEX 11.0 on. `Rcplex()` raises a warning if an older version of CPLEX is used and `n>1`. Default `1`.

Details

Matrices `A` and `C` may be sparse matrices from a class in the hierarchy defined by the **Matrix** package. In that case, the internal casting functions are used to create the proper data structures to pass to CPLEX, which is similar to the column-major storage mode defined by the `dgCMatrix-class` defined by the **Matrix** package.

We also provide a simple S3-style class for sparse matrices `simple_triplet_matrix`, as used in the **relations** package. Matrices `A` and `C` can be objects of this class. See the `examples` for example usage. `simple_triplet_matrix` objects MUST be in column-major order.

The `control` argument can be used to set CPLEX's many parameters, including the particular algorithm used for solving the given problem. See the *ILOG CPLEX Parameters* guide for further details. The following parameters are supported:

trace: Turn CPLEX output on (1) or off(0). Default 1.

maxcalls: Number of calls to the CPLEX optimizer before license is released. Set to 1 to get a new license on every call to Rcplex. Can be any positive number. Default 500.

method: Algorithm to use (Default 0):

- **0:** Automatic: CPLEX chooses algorithm automatically
- **1:** Primal Simplex
- **2:** Dual Simplex
- **3:** Network Simplex
- **4:** Barrier

preind: Turn presolver on (1) or off (0). Default 1.

aggind: Limit on the number of applications of the aggregator. Possible Values: -1 (automatic), 0 (do not use), any positive integer

itlim: Maximum number of simplex iterations. Can be any nonnegative number. Default 1e8.

epagap: Absolute MIP optimality gap tolerance. Can be any nonnegative number. Default 1e-6.

epgap: Relative MIP optimality gap tolerance. Can be any nonnegative number. Default 1e-4.

tilim: Time limit in seconds of call to optimizer. Can be any nonnegative number. Default 1e75.

disjcuts: Indicator for disjunctive cuts used in MIP solver. Must be in -1:3. Default 0 (automatic).

mipemphasis: Indicator for MIP solver emphasis. Must be in 0:4. Default 0 (balance optimality and feasibility)

cliques: Indicator for clique cuts in MIP solver. Must be in -1:2. Default 0 (automatic)

nodesel: Node selection strategy in MIP solver. Must be in 0:3. Default 1 (best-bound search).

probe: Probe level in MPI solver. Must be -1:3. Default 0 (automatic)

varsel: Variable selection strategy in MIP solver. Must be in -1:4. Default 0 (choose best method automatically).

flowcovers: Indicator for flowcover cuts in MIP solver. Must be in -1:2. Default 0 (automatic).

solnpoolagap: Sets an absolute tolerance on the objective value for the solutions in the solution pool. Can be any nonnegative real number. Ignored in versions < 11.0 of CPLEX. Default 0

solnpoolgap: Sets a relative tolerance on the objective value for the solutions in the solution pool. Can be any nonnegative real number. Ignored in versions < 11.0 of CPLEX. Default 0

solnpoolintensity: Controls the trade-off between the number of solutions generated for the solution pool and the amount of time and memory consumed. Must be in 0:4. Ignored in versions < 11.0 of CPLEX. Default 0 (automatic).

round: Flag indicating if integer solutions for MIPs should be rounded before returning. In some cases, CPLEX returns slightly infeasible integer solutions. Setting this option to 1 ensures that the returned solution is integral by rounding. Default 0 (no rounding).

Value

Returns a list with the following components, or, if `n > 1` a list of length equal to the number of optimal solutions containing the following components for each solution:

`xopt`	Values of problem variables at optimum.
`obj`	Value of objective function at optimum.
`status`	Solution status. See CPLEX documentation for meaning of status codes.
`extra`	List with extra information about solution with components
	slack: Values of slack variables for inequality constraints.
	nodecnt: (IF MIP PROBLEM) Number of nodes in the search tree evaluated
	lambda: (IF NOT MIP PROBLEM) Values of dual variables at optimum

Author(s)

Hector Corrada Bravo and Stefan Theussl

References

IBM ILOG CPLEX Optimization Studio documentation

See Also

`Rcplex.close`, `optim`

Examples

```
## A linear program (this is lpex1.c in the CPLEX examples)
cvec <- c(1,2,3)
Amat <- matrix(c(-1,1,1,-1,3,-1),byrow=TRUE,nc=3)
bvec <- c(20,-30)
ub <- c(40,Inf,Inf)

res <- Rcplex(cvec,Amat,bvec,ub=ub,objsense="max",sense=c('L','G'))
print(res)

## A linear program with random data
## use the barrier method
n = 20; m = 25
nnz <- trunc(.2 * m * n)

## entries in simple_triplet_matrix clas
##  *must* be in column major order
nnz <- sort(sample(m*n,nnz,replace=FALSE)-1)
Amat <- simple_triplet_matrix(
             i = (nnz %% m) + 1,
             j = trunc(nnz/m) + 1,
             v = rnorm(nnz),
             nrow=m,ncol=n)

x0 <- runif(n)
b <- as.matrix(Amat) %*% x0
cvec <- rnorm(n)

res <- Rcplex(cvec,Amat,b,sense='E',control=list(method=4))
print(res)

## A quadratic problem (this is qpex1.c in the CPLEX examples)
cvec <- c(1,2,3)
Qmat <- matrix(c(-33,6,0,
                 6,-22,11.5,
                 0,11.5,-11),
               byrow=TRUE,
               nc=3)
Amat <- matrix(c(-1,1,1,
                 1,-3,1),
               byrow=TRUE,nc=3)
bvec <- c(20,30)
ub <- c(40,Inf,Inf)

res <- Rcplex(cvec,Amat,bvec,Qmat,ub=ub,objsense="max")
print(res)

## A mixed integer linear program (mipex1.c in the CPLEX examples)
cvec <- c(1,2,3,1)
Amat <- matrix(c(-1,1,1,10,
                 1,-3,1,0,
                 0,1,0,-3.5),
```

```
               byrow=TRUE, nc=4)
bvec <- c(20,30,0)
lb <- c(0,0,0,2)
ub <- c(40,Inf,Inf,3)
vtype <- c(rep("C",3),"I")

res <- Rcplex(cvec,Amat,bvec,lb=lb,ub=ub,sense=c("L","L","E"),
              objsense="max",vtype=vtype)
print(res)

## A mixed integer quadratic program
cvec <- c(1,2,3,1)
Qmat <- matrix(c(-33,6,0,0,
                  6,-22,11.5,0,
                  0,11.5,-11,0,
                  0,0,0,0),
               byrow=TRUE, nc=4)
Amat <- matrix(c(-1,1,1,10,
                  1,-3,1,0,
                  0,1,0,-3.5),
               byrow=TRUE, nc=4)
bvec <- c(20,30,0)
ub <- c(40,Inf,Inf,3)
vtype <- c(rep("C",3),"I")

res <- Rcplex(cvec,Amat,bvec,Qmat=Qmat,ub=ub,sense=c("L","L","E"),
              objsense="max",vtype=vtype)
print(res)
Rcplex.close()
```

`Rcplex.close` *Release CPLEX license*

Description

This function releases the currently held CPLEX license.

Usage

```
Rcplex.close()
```

Author(s)

Hector Corrada Bravo

See Also

Rcplex

Rcplex_solve_QCP *Solve quadratically constrained optimization problem with CPLEX*

Description

Interface to CPLEX solvers for quadratically constrained linear, quadratic, and mixed-integer programs. The general statement of the problem is

$$\min \frac{1}{2}x'Qx + c'x$$

$$\text{s.t} Ax \leq b$$

$$\text{and} a_i'x + x'Q_ix \leq r_i for i = 1, \ldots, q$$

$$lb \leq x \leq ub$$

If `Q==NULL` then the problem is linear, if any value of the `vtype` argument is "B" or "I" then the problem is a mixed-integer program. The `control` argument is used to set CPLEX's many parameters. See details. The `objsense` determines if the problem is a maximization or minimization problem. The `sense` argument is used to set the constraint directions.

Usage

```
Rcplex_solve_QCP(cvec, Amat, bvec, Qmat = NULL, QC,
       lb = 0, ub = Inf, sense = "L", objsense = c("min", "max"), vtype
  = NULL, n = 1, control = list())
```

Arguments

`cvec`	The linear coefficient of the objective function
`Amat`	The constraint matrix (requires `ncol(Amat)==length(cvec)`)
`bvec`	The constraints right-hand side (requires `length(bvec)==nrow(Amat)`)
`Qmat`	The quadratic coefficient of the objective function. If `NULL` the problem is linear. If not `NULL`, it must be a symmetric positive semidefinite matrix of size `length(cvec)` by `length(cvec)`. Default `NULL`
`QC`	a list with three elements: `QC`, `dir`, and `b`. The element `QC` is a list with the quadratic part `Q`, a matrix, and the linear part of the constraint `L`, a numeric (currently nonzero values are not supported). `dir` has the same meaning as argument `sense` and `b` as `bvec`.
`lb`	Lower bound on the problem variables. If `length(lb)==1` then `lb` is the lower bound of all variables. Otherwise, `length(lb)==length(cvec)`. Set `lb=-Inf` to have no lower bound. Default 0.
`ub`	Upper bound on the problem variables. See `lb` for further details. Default `Inf`.
`control`	A list of CPLEX parameters. See *Details*
`objsense`	Either `"max"` or `"min"`, determines the optimization direction. Default `"min"`

sense	The direction of the inequality in each constraint. If length(sense)==1 then the same value is taken for each constraint. Can be one of "L" (less than or equal), "G" (reater than or equal) or "E" (equal). Requires length(sense)==length(bvec). Default "L".
vtype	Determines the type of each problem variable. Can be one of "C" (continuous), "I" (integer) or "B" (binary). If length(vtype)==1 the same value is taken for all variables. Otherwise, requires length(vtype)==length(ctype). Default "C".
n	Determines the maximal number of solutions the solver should return in case of an MIP with more than one solution at optimum. If CPLEX should search for "all" solutions then n has to be set to NA. In CPLEX this is also called populating the solution pool. The parameters solnpoolagap, solnpoolgap, and solnpoolintensity influence the search for multiple solutions (see also the control argument below for details). Available from CPLEX 11.0 on. Rcplex() raises a warning if an older version of CPLEX is used and n>1. Default 1.

Details

See function link[Rcplex]{Rcplex}() for more information about sparse matrix representation and control arguments.

Value

Returns a list with the following components, or, if n > 1 a list of length equal to the number of optimal solutions containing the following components for each solution:

xopt	Values of problem variables at optimum.
obj	Value of objective function at optimum.
status	Solution status. See CPLEX documentation for meaning of status codes.
extra	List with extra information about solution with components **slack:** Values of slack variables for inequality constraints. **nodecnt:** (IF MIP PROBLEM) Number of nodes in the search tree evaluated **lambda:** (IF NOT MIP PROBLEM) Values of dual variables at optimum

Author(s)

Hector Corrada Bravo and Stefan Theussl

References

IBM ILOG CPLEX Optimization Studio documentation

See Also

Rcplex.close, optim

Examples

```
## objective function
c <- c(1, 2, 3)
Q <- matrix(c(-33, 6, 0, 6, -22, 11.5, 0, 11.5, -11), nrow = 3)

## constraints

## linear part
A <- matrix(c(-1, 1, 1, -3, 1, 1), nrow = 2)
dir <- c("L", "L")
b <- c(20, 30)

## quadratic part
QC <- list(QC = list(Q = list(diag(1, nrow = 3)), L = NULL), dir = "L", b = 1)

## bounds
ub <- c(40, Inf, Inf)

## solve
res <- Rcplex_solve_QCP(c,A, b, Q, ub = ub, QC = QC, sense = dir, objsense = "max")
print(res)

## solve MIQCP
res <- Rcplex_solve_QCP(c, A, b, Q, ub = ub, QC = QC,
                        sense = dir, objsense = "max", vtype = c("C", "I", "C"))

## quadratic and linear part
QC <- list(QC = list(Q = list(diag(1, nrow = 3)), L = list(c(3,4,-3))), dir = "L", b = 1)

## solve
res <- Rcplex_solve_QCP(c,A, b, Q, ub = ub, QC = QC, sense = dir, objsense = "max")
print(res)


Rcplex.close()
```

Index