Package ‘ROI.plugin.alabama’

August 29, 2020

Version 1.0-0

Title 'alabama' Plug-in for the 'R' Optimization Infrastructure

Author Florian Schwendinger [aut, cre]

Maintainer Florian Schwendinger <`FlorianSchwendinger@gmx.at`>

Description Enhances the R Optimization Infrastructure ('ROI') package
with the 'alabama' solver for solving nonlinear optimization problems.

Imports methods, stats, utils, ROI (>= 0.3-0), alabama (>= 1.0.1)

License GPL-3

URL `http://roi.r-forge.r-project.org/`,
`https://r-forge.r-project.org/projects/roi/`

NeedsCompilation no

Repository CRAN

Date/Publication 2020-08-29 15:50:02 UTC

R topics documented:

`control` *alabama*

Description

This package provides a **ROI** plugin to the **alabama** package. The following description of the control parameters is mostly copied from the **alabama** manual.

- [`start`:] The initial values for the parameter vector.
- [`method`:] Unconstrained optimization algorithm for inner loop optimization. Allowed values are `"Nelder-Mead"`, `"BFGS"`, `"CG"`, `"L-BFGS-B"`, `"SANN"`, `"Brent"` and `"nlminb"`.
- [`lam0`:] Initial value for the Lagrangian parameter.
- [`sig0`:] A scaling parameter for penalty term that is augmented to the Lagrangian.
- [`tol`:] Tolerance for convergence of outer iterations of the barrier and/or augmented lagrangian algorithm
- [`max_iter`:] Maximum number of outer iterations.
- [`ilack.max`:] Maximum number of outer iterations where no change in parameters is tolerated.
- [`verbose`:] A logical variable indicating whether information on outer iterations should be printed out. If `TRUE`, at each outer iteration information is displayed on: (i) how well the inequality and equalities are satisfied, (ii) current parameter values, and (iii) current objective function value.
- [`NMinit`:] A logical variable indicating whether `"Nelder-Mead"` algorithm should be used for the first outer iteration.
- [`i.scale`:] A vector of length equal to number of inequalities that may be used to scale the inequalities or it can be a scalar in which case all the inequalities are scaled by the same value.
- [`e.scale`:] A vector of length equal to number of equalities that may be used to scale the equalities or it can be a scalar in which case all the equalities are scaled by the same value.
- [`kkt2.check`:] A logical variable (`TRUE`/`FALSE`) indicating whether the second-order KKT condition should be checked. Deafult is TRUE. It may be set to FALSE in problems where the Hessian computation can b etime consuming.
- [`control.optim`:] A list of control parameters to be used by the unconstrained optimization algorithm in the inner loop. Identical to that used in **optim** or in **nlminb**.

References

Ravi Varadhan (2015). alabama: Constrained Nonlinear Optimization. R package version 2015.3-1. https://CRAN.R-project.org/package=alabama

Examples

```
library(ROI)

n <- 2L
x <- OP(F_objective(sum, n = n),
        bounds = V_bound(nobj = 2, ld = -1, ud = 1))

control_optim <- list(trace = 0, fnscale = 1, parscale = rep.int(1, n),
                      ndeps = rep.int(0.001, n), maxit = 100L, abstol = -Inf,
                      reltol = sqrt(.Machine$double.eps), alpha = 1,
                          beta = 0.5, gamma = 2, REPORT = 10, type = 1, lmm = 5,
                          factr = 1e+07, pgtol = 0, tmax = 10, temp = 10)

control <- list(start = c(0, 0), method = "BFGS", lam0 = 10, sig0 = 100,
                tol = 1e-07, max_iter = 50, verbose = FALSE, NMinit = FALSE,
                ilack.max = 6, i.scale = 1, e.scale = 1, kkt2.check = TRUE,
                control.optim = control_optim)

s <- ROI_solve(x, solver = "alabama", control)
```

Example-1 *Banana*

Description

The following example is also known as Rosenbrock's banana function (https://en.wikipedia.org/wiki/Rosenbrock_function).

$$minimize\ f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Solution: `c(1,1)`

Examples

```
library(ROI)

f <- function(x) {
    return( 100 * (x[2] - x[1]^2)^2 + (1 - x[1])^2 )
}

f.gradient <- function(x) {
    return( c( -400 * x[1] * (x[2] - x[1] * x[1]) - 2 * (1 - x[1]),
               200 * (x[2] - x[1] * x[1])) )
}

x <- OP(objective = F_objective(f, n = 2L, G = f.gradient),
        bounds = V_bound(li = 1:2, ui = 1:2, lb = c(-3, -3), ub = c(3,  3)))

nlp <- ROI_solve(x, solver = "alabama", start = c(-2, 2.4), method = "BFGS")
nlp
## Optimal solution found.
## The objective value is: 3.049556e-23
solution(nlp)
## [1] 1 1
```

Example-2 *Hock-Schittkowski-Collection Problem 16*

Description

The following example solves problem 16 from the `Hock-Schittkowski-Collection`.

$$minimize\ f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$subject\ to: \quad x_1 + x_2^2 \geq 0 \quad x_1^2 + x_2 \geq 0$$

$$-2 \geq x_1 \geq 0.5 \quad x_2 \geq 1$$

Solution: `c(0.5,0.25)`

Examples

```
library(ROI)

f <- function(x) {
    return( 100 * (x[2] - x[1]^2)^2 + (1 - x[1])^2 )
}

f.gradient <- function(x) {
    return( c( -400 * x[1] * (x[2] - x[1] * x[1]) - 2 * (1 - x[1]),
               200 * (x[2] - x[1] * x[1])) )
}

x <- OP( objective = F_objective(f, n=2L, G=f.gradient),
         constraints = c(F_constraint(F=function(x) x[1] + x[2]^2, ">=", 0,
                                      J=function(x) c(1, 2*x[2])),
                         F_constraint(F=function(x) x[1]^2 + x[2], ">=", 0,
                                      J=function(x) c(2*x[1], x[2]))),
         bounds = V_bound(li=1:2, ui=1:2, lb=c(-2, -Inf), ub=c(0.5,  1)) )

nlp <- ROI_solve(x, solver="alabama", start=c(-2, 1))
nlp
## Optimal solution found.
## The objective value is: 2.499999e-01
solution(nlp)
## [1] 0.5000001 0.2499994
```

Example-3 *Hock-Schittkowski-Collection Problem 36*

Description

The following example solves exmaple 36 from the Hock-Schittkowski-Collection.

$$minimize\ \ -x_1 x_2 x_3$$

$$subject\ to:\ \ x_1 + 2x_2 + x_3 \leq 72$$

$$0 \leq x_1 \leq 20,\ 0 \leq x_2 \leq 11,\ 0 \leq x_3 \leq 42$$

Examples

```
library(ROI)

hs036_obj <- function(x) {
    -x[1] * x[2] * x[3]
}

hs036_con <- function(x) {
    x[1] + 2 * x[2] + 2 * x[3]
}


x <- OP( objective = F_objective(hs036_obj, n = 3L),
         constraints = F_constraint(hs036_con, "<=", 72),
         bounds = V_bound(ub = c(20, 11, 42)) )

nlp <- ROI_solve(x, solver = "alabama", start = c(10, 10, 10))
nlp
## Optimal solution found.
## The objective value is: -3.300000e+03
solution(nlp, "objval")
## [1] -3300
solution(nlp)
## [1] 20 11 15
```

Index