Package 'mcGlobaloptim'

February 20, 2015

Version 0.1

Date 2013-10-11

Title Global optimization using Monte Carlo and Quasi Monte Carlo simulation

Author Thierry Moudiki

Maintainer Thierry Moudiki <thierry.moudiki@gmail.com>

Depends R (>= 2.12.1), utils

Description The package performs global optimization combining Monte Carlo and Quasi Monte Carlo simulation with a local search. \{}n The local searches can be easily speeded-up by using a network of local workstations.

License GPL (>= 2)

Imports randtoolbox, snow

URL http://www.r-project.org

NeedsCompilation no

Repository CRAN

Date/Publication 2013-10-12 00:42:10

R topics documented:

mcGlobaloptim-package *Global optimization using Monte Carlo and Quasi Monte Carlo simultation*

Description

Global optimization combining Monte Carlo and Quasi Monte Carlo simulation with a local search. The local searches can be easily speeded up by using a network of local workstations.

Details

Package:	mcGlobaloptim
Type:	Package
Version:	0.5
Date:	2013-10-10
License:	GPL-2 | GPL-3

Author(s)

Thierry Moudiki Maintainer: <thierry.moudiki@gmail.com>

References

C. Dutang, P. Savicky (2013). randtoolbox: Generating and Testing Random Numbers. R package version 1.14.
M. J. A. Eugster, J. Knaus, C. Porzelius, M. Schmidberger, E. Vicedo (2011). Hands-on tutorial for parallel computing with R. Computational Statistics. Springer.
M. Gilli, E. Schumann (2010). A Note on 'Good Starting Values' in Numerical Optimisation. Available at SSRN.
F. Glover, G. Kochenberger (2003). Handbook of Metaheuristics. Kluwer Academic Publishers.
H. Niederreiter (1992). Random Number Generation and Quasi-Monte Carlo Methods. Society for Industrial and Applied Mathematics.
M. Schmidberger, M. Morgan, D. Eddelbuettel, H. Yu, L. Tierney, U. Mansmann(2009). State of the art in parallel computing with R. Journal of Statistical Software.
L. Tierney, A. J. Rossini, Na Li and H. Sevcikova (2013). snow: Simple Network of Workstations. R package version 0.3-12.
A. Zhigljavsky, A. Zilinkas (2008). Stochastic Global Optimization. Springer Science+Business Media.

multiStartoptim *Multistart global optimization using Monte Carlo and Quasi Monte Carlo simulation.*

Description

multiStartoptim generates pseudo-random and quasi-random numbers within the search domain specified by its bounds. Local searches are then performed by a user-selected method, either on the whole set of numbers generated as starting points, or only on the points lying under the objective function's median. When (and only when) a high number of local searches are to be performed, parallel computation can be used to speed-up the search.

Usage

```
multiStartoptim(start0 = NULL, objectivefn, gradient = NULL, ...,
 hessian = NULL, lower = -Inf, upper = Inf, control = list(),
method = c("L-BFGS-B", "Nelder-Mead", "nlminb"), nbtrials = NULL,
 typerunif = c("runifbase", "runifantithetics", "sobol", "torus",
 "niederreiterlodisp"), localsearch = c("exhaustive", "median"),
 verb = FALSE, nbclusters = NULL)
```

Arguments

start0	Starting value (a numeric value or a vector). If start0 is provided, a simple call to optim or nlminb is performed. No numbers' simulation, no parallel computation.
objectivefn	The function to be minimized (the objective function).
gradient	The gradient of the objective function.
...	Additional parameters to be provided to objective function (see in examples)
hessian	The hessian parameter (or function), to be use like in optim (or nlminb).
lower	lower bound(s) for the search domain. Should be provided and finite.
upper	upper bound(s) for the search domain. Should be provided and finite.
control	Optimization control parameters defined exactly as in optim or nlminb.
method	The method used for local search. Can be either "L-BFGS-B", "Nelder-Mead" from optim, or a PORT routine implemented by nlminb.
nbtrials	Number of pseudo-random and quasi-random numbers generated within the search domain specified by the lower and upper bounds
typerunif	The type of pseudo-random and quasi-random numbers simulation chosen. The default method "runifbase" is based on uniform pseudo-random numbers generated from SF-Mersenne Twister algorithm, implemented in package randtoolbox. Please refer to randtoolbox package's vignette for more in-depth information on the simulation methods used.

localsearch	The local searches are performed by a user-selected method, either on the whole set of numbers generated as starting points, or only on the points lying under the objective function's median.
verb	If TRUE, the user can have details on the computation, typically the iterations and the objective function values obtained when the algorithm finds a better solution.
nbclusters	The number of clusters on a local host to be used for parallel computation.

Details

For details on the methods used for local searches, refer to optim or nlminb. For details on the random or quasi-random simulation, refer to package randtoolbox, especially its vignette. For details on parallel computation on a simple network of workstations, refer to package snow.

Value

If verb != TRUE, a list with the values obtained from optim or nlminb. Or else, the list of values from optim or nlminb, plus a list of additional information on the iterations and the objective function values obtained whenever the algorithm finds a better solution.

Note

Minimization is performed by default. To maximize, try scaling the objective with -1.

Author(s)

Thierry Moudiki

References

C. Dutang, P. Savicky (2013). randtoolbox: Generating and Testing Random Numbers. R package version 1.14.
M. J. A. Eugster, J. Knaus, C. Porzelius, M. Schmidberger, E. Vicedo (2011). Hands-on tutorial for parallel computing with R. Computational Statistics. Springer.
M. Gilli, E. Schumann (2010). A Note on 'Good Starting Values' in Numerical Optimisation. Available at SSRN.
F. Glover, G. Kochenberger (2003). Handbook of Metaheuristics. Kluwer Academic Publishers.
H. Niederreiter (1992). Random Number Generation and Quasi-Monte Carlo Methods. Society for Industrial and Applied Mathematics.
M. Schmidberger, M. Morgan, D. Eddelbuettel, H. Yu, L. Tierney, U. Mansmann(2009). State of the art in parallel computing with R. Journal of Statistical Software.
L. Tierney, A. J. Rossini, Na Li and H. Sevcikova (2013). snow: Simple Network of Workstations. R package version 0.3-12.
A. Zhigljavsky, A. Zilinkas (2008). Stochastic Global Optimization. Springer Science+Business Media.

Examples

```
### Example from optim :
# "wild" function, global minimum at about -15.81515
```

```
fw <- function (x)
{
  10*sin(0.3*x)*sin(1.3*x^2) + 0.00001*x^4 + 0.2*x+80
}
plot(fw, -50, 50, n = 1000, main = "optim() minimising 'wild function'")
(minfw <- multiStartoptim(objectivefn = fw, lower = -40,
                          upper = 40, method = "nlminb", nbtrials = 500,
                          typerunif = "sobol", verb = TRUE))
points(minfw$res$par, minfw$res$objective, pch = 8, lwd = 2, col = "red", cex = 2)

### Calibrating the Nelson-Siegel-Svensson model (from Gilli, Schumann (2010)) :
# Nelson - Siegel - Svensson model
NSS2 <- function(betaV, mats)
{
  gam1 <- mats / betaV [5]
  gam2 <- mats / betaV [6]
  aux1 <- 1 - exp (- gam1)
  aux2 <- 1 - exp (- gam2)
  betaV[1] + betaV[2] * (aux1 / gam1) +
    betaV[3] * (aux1 / gam1 + aux1 - 1) +
    betaV[4] * (aux2 / gam2 + aux2 - 1)
}

betaTRUE <- c(5, -2 ,5, -5 ,1 ,3)
mats <- c(1 ,3 ,6 ,9 ,12 ,15 ,18 ,21 ,24 ,30 ,36 ,48 ,60 ,72 ,84 ,
          96,108 ,120)/ 12
yM <- NSS2 (betaTRUE, mats)
dataList <- list ( yM = yM, mats = mats, model = NSS2)
plot (mats, yM, xlab = " maturities in years ", ylab =" yields in pct. ")

# define objective function
OF <- function (betaV, dataList) {
  mats <- dataList$mats
  yM <- dataList$yM
  model <- dataList$model
  y <- model(betaV, mats)
  aux <- y - yM
  crossprod(aux)
}

settings <- list (min = c( 0, -15, -30, -30 ,0 ,3),
                  max = c (15, 30, 30, 30 ,3 ,6), d = 6)
NSStest <- multiStartoptim(objectivefn = OF, data = dataList,
                           lower = settings$min,
                           upper = settings$max,
                           method = "nlminb",
                           nbtrials = 50, typerunif = "torus")
lines(mats, NSS2(NSStest$par, mats), col = 'blue')

# (Only) when nbtrials is high, parallelization makes the computation faster.
# Try a lower number of trials and compare the expressions with a timer.
nbtrials <- 500
#t1 <- multiStartoptim(objectivefn = OF, data = dataList,
```

```
#                      lower = settings$min,
#                      upper = settings$max,
#                      nbtrials = nbtrials,
#                      typerunif = "sobol",
#                      method="nlminb",
#                      nbclusters=2)
t0 <- multiStartoptim(objectivefn = OF, data = dataList,
                      lower = settings$min,
                      upper = settings$max,
                      nbtrials=nbtrials,
                      typerunif="sobol",
                      method="nlminb")

#all.equal(t0, t1)
```

Index