Package ‘ABCoptim’

October 12, 2022

Type Package

Title Implementation of Artificial Bee Colony (ABC) Optimization

Version 0.15.0

Date 2017-11-05

Description An implementation of Karaboga (2005) Artificial Bee Colony Optimization algorithm <http://mf.erciyes.edu.tr/abc/pub/tr06_2005.pdf>. This (working) version is a Work-in-progress, which is why it has been implemented using pure R code. This was developed upon the basic version programmed in C and distributed at the algorithm's official website.

Classification/ACM G.1.6

Classification/JEL C61

Encoding UTF-8

URL http://github.com/gvegayon/ABCoptim, http://mf.erciyes.edu.tr/abc/

License MIT + file LICENSE

LazyLoad yes

LinkingTo Rcpp

Imports Rcpp, graphics, stats, utils

RoxygenNote 6.0.1

Suggests testthat, covr

NeedsCompilation yes

Author George Vega Yon [aut, cre],
Enyelbert Muñoz [ctb]

Maintainer George Vega Yon <g.vegayon@gmail.com>

Repository CRAN

Date/Publication 2017-11-06 08:55:46 UTC

R topics documented:

`ABCoptim-package` *An implementation of the Artificial Bee Colony (ABC) Algorithm*

Description

This is an implementation of Karaboga (2005) ABC optimization algorithm. It was developed upon the basic version programmed in `C` and distributed at the algorithm's official website (see the references).

Details

Please consider that this version is in alpha state of development, thus any evident (precision) error should be blaimed to the package author (not to the algorithm itself)

Please visit the project home for more information: https://github.com/gvegayon/ABCoptim.

Author(s)

George Vega Yon <g.vegayon@gmail.com> [aut],

Enyelbert Muñoz <enyeldoc2011@gmail.com> [cnt]

References

D. Karaboga, *An Idea based on Honey Bee Swarm for Numerical Optimization*, tech. report TR06,Erciyes University, Engineering Faculty, Computer Engineering Department, 2005 http://mf.erciyes.edu.tr/abc/pub/tr06_2005.pdf

Artificial Bee Colony (ABC) Algorithm (website) http://mf.erciyes.edu.tr/abc/index.htm

Basic version of the algorithm implemented in `C` (ABC's official website) http://mf.erciyes.edu.tr/abc/form.aspx

Examples

```
## Not run:
  demo(ABCoptim) # Some functions...

## End(Not run)
```

abc_optim *Artificial Bee Colony Optimization*

Description

Implements Karaboga (2005) Artificial Bee Colony (ABC) Optimization algorithm.

Usage

```
abc_optim(par, fn, ..., FoodNumber = 20, lb = rep(-Inf, length(par)),
  ub = rep(+Inf, length(par)), limit = 100, maxCycle = 1000,
  optiinteger = FALSE, criter = 50, parscale = rep(1, length(par)),
  fnscale = 1)

## S3 method for class 'abc_answer'
print(x, ...)

abc_cpp(par, fn, ..., FoodNumber = 20, lb = rep(-Inf, length(par)),
  ub = rep(+Inf, length(par)), limit = 100, maxCycle = 1000,
  criter = 50, parscale = rep(1, length(par)), fnscale = 1)

## S3 method for class 'abc_answer'
plot(x, y = NULL,
  main = "Trace of the Objective Function", xlab = "Number of iteration",
  ylab = "Value of the objective Function", type = "l", ...)
```

Arguments

`par`	Initial values for the parameters to be optimized over
`fn`	A function to be minimized, with first argument of the vector of parameters over which minimization is to take place. It should return a scalar result.
`...`	In the case of `abc_*`, further arguments to be passed to 'fn', otherwise, further arguments passed to the method.
`FoodNumber`	Number of food sources to exploit. Notice that the param `NP` has been deprecated.
`lb`	Lower bound of the parameters to be optimized.
`ub`	Upper bound of the parameters to be optimized.
`limit`	Limit of a food source.
`maxCycle`	Maximum number of iterations.
`optiinteger`	Whether to optimize binary parameters or not.
`criter`	Stop criteria (numer of unchanged results) until stopping
`parscale`	Numeric vector of length `length(par)`. Scale applied to the parameters (see `optim`).

fnscale	Numeric scalar. Scale applied function. If fnscale < 0, then the problem becomes a maximization problem (see optim).
x	An object of class abc_answer.
y	Ignored
main	Passed to plot.
xlab	Passed to plot.
ylab	Passed to plot.
type	Passed to plot.

Details

This implementation of the ABC algorithm was developed based on the basic version written in C and published at the algorithm's official website (see references).

abc_optim and abc_cpp are two different implementations of the algorithm, the former using pure R code, and the later using C++, via the **Rcpp** package. Besides of the output, another important difference between the two implementations is speed, with abc_cpp showing between 50% and 100% faster performance.

Upper and Lower bounds (ub, lb) equal to infinite will be replaced by either .Machine$double.xmax or -.Machine$double.xmax.

If D (the number of parameters to be optimzed) is greater than one, then lb and ub can be either scalars (assuming that all the parameters share the same boundaries) or vectors (the parameters have different boundaries each other).

The plot method shows the trace of the objective function as the algorithm unfolds. The line is merely the result of the objective function evaluated at each point (row) of the hist matrix return by abc_optim/abc_cpp.

For now, the function will return with error if ... was passed to abc_optim/abc_cpp, since those argumens are not stored with the result.

Value

An list of class abc_answer, holding the following elements:

Foods	Numeric matrix. Last position of the bees.
f	Numeric vector. Value of the function evaluated at each set of Foods.
fitness	Numeric vector. Fitness of each Foods.
trial	Integer vector. Number of trials at each Foods.
value	Numeric scalar. Value of the function evaluated at the optimum.
par	Numeric vector. Optimum found.
counts	Integer scalar. Number of cycles.
hist	Numeric matrix. Trace of the global optimums.

Author(s)

George Vega Yon <g.vegayon@gmail.com>

References

D. Karaboga, *An Idea based on Honey Bee Swarm for Numerical Optimization*, tech. report TR06,Erciyes University, Engineering Faculty, Computer Engineering Department, 2005 http://mf.erciyes.edu.tr/abc/pub/tr06_2005.pdf

Artificial Bee Colony (ABC) Algorithm (website) http://mf.erciyes.edu.tr/abc/index.htm

Basic version of the algorithm implemented in `C` (ABC's official website) http://mf.erciyes.edu.tr/abc/form.aspx

Examples

```
# EXAMPLE 1: The minimum is at (pi,pi) --------------------------------------

fun <- function(x) {
  -cos(x[1])*cos(x[2])*exp(-((x[1] - pi)^2 + (x[2] - pi)^2))
}

abc_optim(rep(0,2), fun, lb=-10, ub=10, criter=50)

# This should be equivalent
abc_cpp(rep(0,2), fun, lb=-10, ub=10, criter=50)

# We can also turn this into a maximization problem, and get the same
# results
fun <- function(x) {
  # We've removed the '-' from the equation
  cos(x[1])*cos(x[2])*exp(-((x[1] - pi)^2 + (x[2] - pi)^2))
}

abc_cpp(rep(0,2), fun, lb=-10, ub=10, criter=50, fnscale = -1)

# EXAMPLE 2: 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

ans <- abc_optim(50, fw, lb=-100, ub=100, criter=100)
ans[c("par", "counts", "value")]


# EXAMPLE 3: 5D sphere, global minimum at about (0,0,0,0,0) -----------------
fs <- function(x) sum(x^2)

ans <- abc_optim(rep(10,5), fs, lb=-100, ub=100, criter=200)
ans[c("par", "counts", "value")]


# EXAMPLE 4: An Ordinary Linear Regression ----------------------------------

set.seed(1231)
k <- 4
```

```
n <- 5e2

# Data generating process
w <- matrix(rnorm(k), ncol=1)     # This are the model parameters
X <- matrix(rnorm(k*n), ncol = k) # This are the controls
y <- X %*% w                      # This is the observed data

# Objective function
fun <- function(x) {
  sum((y - X%*%x)^2)
}

# Running the regression
ans <- abc_optim(rep(0,k), fun, lb = -10000, ub=10000)

# Here are the outcomes: Both columns should be the same
cbind(ans$par, w)
#             [,1]        [,2]
# [1,] -0.08051177 -0.08051177
# [2,]  0.69528553  0.69528553
# [3,] -1.75956316 -1.75956316
# [4,]  0.36156427  0.36156427


# This is just like OLS, with no constant
coef(lm(y~0+X))
#         X1          X2          X3          X4
#-0.08051177  0.69528553 -1.75956316  0.36156427
```

Index