Package: Jaya (via r-universe)

September 9, 2024

Type Package

Title Jaya, a Gradient-Free Optimization Algorithm

Version 0.1.9

Maintainer Neeraj Bokde <neerajdhanraj@gmail.com>

Description Maximization or Minimization of a fitness function using Jaya Algorithm (JA). A population based method which repeatedly modifies a population of individual solutions. Capable of solving both constrained and unconstrained optimization problems. It does not contain any hyperparameters. For further details: R.V. Rao (2016) <doi:10.5267/j.ijiec.2015.8.004> .

License GPL (>= 2)

Suggests knitr, rmarkdown, evaluate, testthat

Imports GA

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Mayur Shende [aut], Neeraj Bokde [aut, cre] (<https://orcid.org/0000-0002-3493-9302>)

Date/Publication 2019-11-12 11:20:02 UTC

Repository https://neerajdhanraj.r-universe.dev

RemoteUrl https://github.com/neerajdhanraj/jaya

RemoteRef HEAD

RemoteSha f15a0fe96ec9aec27d5f44f4ffcb5164ec52d93f

Contents

---

jaya *Jaya Algorithm, a gradient-free optimization algorithm. Maximization of a function using Jaya Algorithm (JA). A population based method which repeatedly modifies a population of individual solutions. Capable of solving both constrained and unconstrained optimization problems. Does not contain any hyperparameters.*

Description

Jaya Algorithm, a gradient-free optimization algorithm. Maximization of a function using Jaya Algorithm (JA). A population based method which repeatedly modifies a population of individual solutions. Capable of solving both constrained and unconstrained optimization problems. Does not contain any hyperparameters.

Usage

```
jaya(fun, lower, upper, popSize = 50, maxiter, n_var, seed = NULL,
  suggestions = data.frame(), opt = "minimize")
```

Arguments

fun	as a function to be optimized
lower	as a vector of lower bounds for the vaiables in the function
upper	as a vector of upper bounds for the vaiables in the function
popSize	as population size
maxiter	as number of iterations to run for finding optimum solution
n_var	as number of variables used in the function to optimize
seed	as an integer vector containing the random number generator state
suggestions	as a data frame of solutions string to be included in the initial population
opt	as a string either "maximize" or "minimize" the function

Examples

```
# Test Function to minimize
square <- function(x){return((x[1]^2)+(x[2]^2))}
jaya(fun = square, lower = c(-100,-100), upper = c(100,100), maxiter = 10, n_var = 2)
```

`plot.jaya` *#' Function to plot the 'best value' VS 'no. of iterations'*

Description

#' Function to plot the 'best value' VS 'no. of iterations'

Usage

```
## S3 method for class 'jaya'
plot(x, ...)
```

Arguments

`x`	as an output object from 'jaya' function
`...`	as Additional graphical parameters given to plot function

Value

Returns plot showing 'best value' VS 'no. of iterations'

`summary.jaya` *Function to summarize the Jaya function*

Description

Function to summarize the Jaya function

Usage

```
## S3 method for class 'jaya'
summary(object, ...)
```

Arguments

`object`	as an output object from 'jaya' function
`...`	Additional parameters given to the function

Value

returns the summary of output object from 'jaya' function

Examples

```
# Test Function to minimize
square <- function(x){return((x[1]^2)+(x[2]^2))}
a <- jaya(fun = square, lower = c(-100,-100), upper = c(100,100), maxiter = 10, n_var = 2)
summary(a)
```

Index