Package 'Jaya'

May 7, 2026

Title Gradient-Free Optimization Algorithm for Single and Multi-Objective Problems

Version 1.0.3

Description An implementation of the Jaya optimization algorithm for both single-objective and multi-objective problems. Jaya is a population-based, gradient-free optimization algorithm capable of solving constrained and unconstrained optimization problems without hyperparameters. This package includes features such as multi-objective Pareto optimization, adaptive population adjustment, and early stopping. For further details, see R.V. Rao (2016) <doi:10.5267/j.ijiec.2015.8.004>.

Depends R (>= 3.5.0)

Imports parallel

Suggests knitr, rmarkdown, evaluate, testthat

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/neerajdhanraj/Jaya

BugReports https://github.com/neerajdhanraj/Jaya/issues

NeedsCompilation no

RoxygenNote 7.3.2

VignetteBuilder knitr

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

Maintainer Neeraj Bokde <neerajdhanraj@gmail.com>

Repository CRAN

Date/Publication 2024-11-18 13:10:02 UTC

Contents

`jaya` *Jaya Algorithm for Single-Objective Optimization*

Description

Implements the Jaya optimization algorithm for single-objective optimization. The algorithm minimizes or maximizes the given objective function over specified bounds.

Usage

```
jaya(
  fun = NULL,
  lower,
  upper,
  popSize = 50,
  maxiter,
  n_var,
  seed = NULL,
  suggestions = data.frame(),
  opt = "minimize",
  objectives = NULL,
  constraints = list(),
  early_stopping = FALSE,
  tolerance = 1e-06,
  patience = 10,
  adaptive_pop = FALSE,
  min_popSize = 20,
  max_popSize = 100,
  parallel = FALSE,
  cores = NULL
)
```

Arguments

`fun`	Objective function to be minimized or maximized (single-objective).
`lower`	Vector of lower bounds for the decision variables.
`upper`	Vector of upper bounds for the decision variables.
`popSize`	Size of the population for the optimization process.
`maxiter`	Maximum number of iterations.
`n_var`	Number of decision variables.

seed	Optional random seed for reproducibility.
suggestions	Optional data frame of initial population suggestions.
opt	Specify whether to "minimize" or "maximize" the objective function.
objectives	(optional) A list of functions for multi-objective optimization.
constraints	(optional) A list of constraints as functions returning <= 0 for feasibility.
early_stopping	Logical. If TRUE, stops optimization early based on tolerance and patience.
tolerance	Numeric. Tolerance for early stopping.
patience	Integer. Number of iterations to wait for improvement before stopping early.
adaptive_pop	Logical. If TRUE, enables adaptive population size adjustment.
min_popSize	Integer. Minimum population size for adaptive adjustment.
max_popSize	Integer. Maximum population size for adaptive adjustment.
parallel	Logical. If TRUE, enables parallel computation for evaluating population.
cores	Integer. Number of cores to use for parallel computation. Defaults to all available cores minus one.

Value

A list containing the following: - 'Best': The best solution found (variable values and objective function value). - 'Iterations': Best objective function values at each iteration.

Examples

```
# Example: Single-objective optimization
sphere_function <- function(x) sum(x^2)
result <- jaya(
  fun = sphere_function,
  lower = rep(-5, 3),
  upper = rep(5, 3),
  popSize = 20,
  maxiter = 50,
  n_var = 3,
  opt = "minimize"
)
print(summary(result))
plot(result)
```

---

jaya_multi *Jaya Algorithm for Multi-Objective Optimization*

Description

Implements the Jaya optimization algorithm for multi-objective optimization. This algorithm supports non-dominated sorting and handles constraints and adaptive population sizes.

Usage

```
jaya_multi(
  objectives,
  lower,
  upper,
  popSize = 50,
  maxiter,
  n_var,
  seed = NULL,
  suggestions = data.frame(),
  constraints = list(),
  adaptive_pop = FALSE,
  min_popSize = 20,
  max_popSize = 100,
  early_stopping = FALSE,
  tolerance = 1e-06,
  patience = 10
)
```

Arguments

objectives	A list of objective functions to optimize.
lower	Numeric vector specifying the lower bounds for variables.
upper	Numeric vector specifying the upper bounds for variables.
popSize	Population size. Default is 50.
maxiter	Maximum number of iterations.
n_var	Number of variables.
seed	Random seed for reproducibility. Default is 'NULL'.
suggestions	Data frame of initial suggestions for starting population. Default is an empty data frame.
constraints	A list of constraint functions. Each constraint should return a non-positive value if satisfied.
adaptive_pop	Logical. Whether to adapt population size during optimization. Default is 'FALSE'.
min_popSize	Minimum population size if adaptive population is enabled. Default is 20.
max_popSize	Maximum population size if adaptive population is enabled. Default is 100.
early_stopping	Logical. Whether to stop early if no improvement is observed. Default is 'FALSE'.
tolerance	Numeric tolerance for early stopping. Default is 1e-6.
patience	Number of iterations to wait for improvement before stopping. Default is 10.

Value

A list containing: - 'Pareto_Front': A data frame of non-dominated solutions with decision variables and their corresponding objective values. - 'Solutions': The final population including decision variables and their objective values.

Examples

```
# Example: Multi-objective optimization
sphere_function_1 <- function(x) sum(x^2)
sphere_function_2 <- function(x) sum((x - 2)^2)
result <- jaya_multi(
  objectives = list(sphere_function_1, sphere_function_2),
  lower = rep(-5, 3),
  upper = rep(5, 3),
  popSize = 20,
  maxiter = 50,
  n_var = 3
)
print(summary(result))
```

`plot.jaya` *Plot Function for Jaya Algorithm Results*

Description

This function generates plots for single-objective optimization results from the Jaya algorithm. It visualizes the best objective function value against the number of iterations.

Usage

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

Arguments

`x`	An object of class `jaya` containing the optimization results from the `jaya` function.
`...`	Additional graphical parameters passed to the `plot` function.

Details

This function supports plotting results for single-objective optimization. It creates a plot of the best objective function value observed across iterations. Ensure that the input object is from the `jaya` function.

Examples

```
# Example: Single-objective optimization
sphere_function <- function(x) sum(x^2)

lower_bounds <- rep(-5, 3)
upper_bounds <- rep(5, 3)
pop_size <- 20
```

```
max_iterations <- 50
num_variables <- length(lower_bounds)

# Run optimization
single_result <- jaya(
  fun = sphere_function,
  lower = lower_bounds,
  upper = upper_bounds,
  popSize = pop_size,
  maxiter = max_iterations,
  n_var = num_variables,
  opt = "minimize"
)

# Plot the result
plot(single_result)
```

`plot_jaya_multi_pairwise` *Pairwise Plot Function for Multi-Objective Optimization Results*

Description

Generates pairwise 2D plots for all combinations of objectives in the Pareto front. This function visualizes trade-offs between different objectives.

Usage

```
plot_jaya_multi_pairwise(x, objectives = NULL, ...)
```

Arguments

`x`	An object of class `jaya_multi` containing the optimization results, including the Pareto front.
`objectives`	A vector of objective column names to include in the pairwise plots. If `NULL`, all objectives in the Pareto front are used.
`...`	Additional graphical parameters passed to the `plot` function.

Details

The function automatically detects objectives in the Pareto front if not specified. It creates pairwise plots for all possible combinations of objectives.

Examples

```
# Example usage of plot_jaya_multi_pairwise
# Define sample multi-objective optimization problem
objective1 <- function(x) sum(x^2)
objective2 <- function(x) sum(abs(x))
objective3 <- function(x) sum(x^3)
objective4 <- function(x) sum(x^4)

objectives <- list(objective1, objective2, objective3, objective4)
lower_bounds <- c(-5, -5, -5)
upper_bounds <- c(5, 5, 5)

# Run multi-objective optimization using jaya_multi
set.seed(42)
multi_result <- jaya_multi(
  objectives = objectives,
  lower = lower_bounds,
  upper = upper_bounds,
  popSize = 50,
  maxiter = 100,
  n_var = length(lower_bounds)
)

# Pairwise plot of objectives
plot_jaya_multi_pairwise(multi_result)
```

`summary.jaya` *Summary Method for Jaya Algorithm Optimization Results*

Description

Provides a summary of optimization results for both single-objective and multi-objective cases. Displays key parameters, limits, and results such as the best solution for single-objective optimization or the Pareto front for multi-objective optimization.

Usage

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

Arguments

`object`	An object of class `jaya` or `jaya_multi`, containing the results of the Jaya optimization.
`...`	Additional arguments (currently unused).

Details

- For single-objective optimization, the summary includes the best solution and the associated function value. - For multi-objective optimization, the summary displays the objectives, decision variable limits, and the first few entries of the Pareto front. - Automatically handles missing or incomplete attributes gracefully.

Examples

```
# Single-objective optimization example
sphere_function <- function(x) sum(x^2)
single_result <- jaya(
  fun = sphere_function,
  lower = c(-5, -5, -5),
  upper = c(5, 5, 5),
  popSize = 20,
  maxiter = 50,
  n_var = 3,
  opt = "minimize"
)
summary(single_result)

# Multi-objective optimization example
objective1 <- function(x) sum(x^2)
objective2 <- function(x) sum(abs(x))
multi_result <- jaya_multi(
  objectives = list(objective1, objective2),
  lower = c(-5, -5, -5),
  upper = c(5, 5, 5),
  popSize = 50,
  maxiter = 100,
  n_var = 3
)
summary(multi_result)
```

Index