Package 'vntrs'

May 7, 2026

Title Variable Neighborhood Trust Region Search

Version 0.2.1

Description Implements the variable neighborhood trust region search (VNTRS) algorithm for nonlinear global optimization, following Bierlaire et al. (2009) ``A Heuristic for Nonlinear Global Optimization'' <doi:10.1287/ijoc.1090.0343>. The method combines neighborhood exploration with a trust-region framework to search the solution space efficiently. It can terminate a local search early when the iterates converge toward a previously visited local optimum or when further improvement within the current region is unlikely. The algorithm can also be used to identify multiple local optima.

URL https://loelschlaeger.de/vntrs/

BugReports https://github.com/loelschlaeger/vntrs/issues

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Imports checkmate, oeli (>= 0.7.5), Rcpp

LinkingTo Rcpp, RcppArmadillo

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Lennart Oelschläger [aut, cre] (ORCID: <https://orcid.org/0000-0001-5421-9313>)

Maintainer Lennart Oelschläger <oelschlaeger.lennart@gmail.com>

Repository CRAN

Date/Publication 2026-05-07 17:50:34 UTC

Contents

vntrs *Variable neighborhood trust region search*

Description

Run the variable neighborhood trust region search algorithm.

Usage

```
vntrs(
  f,
  npar,
  minimize = TRUE,
  init_runs = 5L,
  init_min = -1,
  init_max = 1,
  init_iterlim = 20L,
  neighborhoods = 5L,
  neighbors = 5L,
  beta = 0.05,
  iterlim = 100L,
  tolerance = 1e-06,
  inferior_tolerance = 1e-06,
  time_limit = NULL,
  cores = 1L,
  lower = NULL,
  upper = NULL,
  collect_all = FALSE,
  quiet = TRUE
)
```

Arguments

f	[function] A function that accepts a numeric parameter vector and returns either • a numeric objective value, or • a list with value and optional gradient and hessian components. Missing derivatives are approximated by finite differences.
npar	[integer(1)] The number of parameters of f.
minimize	[logical(1)] If TRUE, minimize f; otherwise, maximize it.
init_runs	[integer(1)] Number of random starting points for the initialization stage.

`init_min`, `init_max`
: [numeric(1)]
Lower and upper bounds for the uniform initialization range.

`init_iterlim`
: [integer(1)]
Maximum trust-region iterations during initialization.

`neighborhoods`
: [integer(1)]
Number of neighborhood expansions to try.

`neighbors`
: [integer(1)]
Number of trial points sampled in each neighborhood.

`beta`
: [numeric(1)]
Non-negative scaling factor for neighborhood expansion.

`iterlim`
: [integer(1)]
Maximum trust-region iterations during the main search.

`tolerance`
: [numeric(1)]
Minimum Euclidean distance for two optima to be treated as distinct.

`inferior_tolerance`
: [numeric(1)]
Maximum objective-value gap from the best known solution before a local optimum is discarded early.

`time_limit`
: [numeric(1) | NULL]
Optional time limit in seconds. If reached, the search stops early with a warning.

`cores`
: [integer(1)]
Number of CPU cores used for parallel evaluation.

`lower`, `upper`
: [numeric(npar) | NULL]
Optional lower and upper parameter bounds. Use `NULL` for unbounded dimensions.

`collect_all`
: [logical(1)]
If `TRUE`, keep all converged local optima and disable early stopping for optima that are inferior to the best known solution.

`quiet`
: [logical(1)]
If `TRUE`, suppress progress messages.

Value

A `data.frame` summarizing the identified optima or `NULL` if none could be determined.

References

Bierlaire et al. (2009) "A Heuristic for Nonlinear Global Optimization" doi:10.1287/ijoc.1090.0343.

Examples

```
rosenbrock <- function(x) 100 * (x[2] - x[1]^2)^2 + (1 - x[1])^2
vntrs(f = rosenbrock, npar = 2)
```

Index