Package 'vntrs'

October 12, 2022

Title Variable Neighborhood Trust Region Search

Version 0.1.0

Date 2021-10-09

Description An implementation of the variable neighborhood trust region algorithm Bierlaire et al. (2009) ``A Heuristic for Nonlinear Global Optimization'' <doi:10.1287/ijoc.1090.0343>.

Imports trust

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

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

Maintainer Lennart Oelschläger <lennart.oelschlaeger@uni-bielefeld.de>

Repository CRAN

Date/Publication 2021-10-18 14:30:02 UTC

R topics documented:

check_controls *Check* controls.

Description

This function checks the input controls for the vntrs package.

Usage

```
check_controls(controls)
```

Arguments

controls	Either NULL or a named list with the following elements. Missing elements are set to the default values in parentheses.

- init_runs (5): The number of initial searches.
- init_min (-1): The minimum argument value for the random initialization.
- init_max (1): The maximum argument value for the random initialization.
- init_iterlim (20): The number of iterations for the initial searches.
- neighborhoods (5): The number of nested neighborhoods.
- neighbors (5): The number of neighbors in each neighborhood.
- beta (0.05): A non-negative weight factor to account for the function's curvature in the selection of the neighbors. If beta = 0, the curvature is ignored. The higher the value, the higher the probability of selecting a neighbor in the direction of the highest function curvature.
- iterlim (1000): The maximum number of iterations to be performed before the local search is terminated.
- tolerance (1e-6): A positive scalar giving the tolerance for comparing different optimal arguments for equality.
- time_limit (NULL): The time limit in seconds for the algorithm.

Value

The checked and filled list controls.

check_f *Check* f.

Description

This function checks the input f for the vntrs package.

Usage

```
check_f(f, npar, controls)
```

Arguments

f	A function that computes value, gradient, and Hessian of the function to be optimized and returns them as a named list with elements value, gradient, and hessian.
npar	The number of parameters of f.
controls	Either NULL or a named list with the following elements. Missing elements are set to the default values in parentheses.

- init_runs (5): The number of initial searches.
- init_min (-1): The minimum argument value for the random initialization.
- init_max (1): The maximum argument value for the random initialization.
- init_iterlim (20): The number of iterations for the initial searches.
- neighborhoods (5): The number of nested neighborhoods.
- neighbors (5): The number of neighbors in each neighborhood.
- beta (0.05): A non-negative weight factor to account for the function's curvature in the selection of the neighbors. If beta = 0, the curvature is ignored. The higher the value, the higher the probability of selecting a neighbor in the direction of the highest function curvature.
- iterlim (1000): The maximum number of iterations to be performed before the local search is terminated.
- tolerance (1e-6): A positive scalar giving the tolerance for comparing different optimal arguments for equality.
- time_limit (NULL): The time limit in seconds for the algorithm.

Value

No return value, called for side-effects.

---

initialize *Initialize VNTRS.*

Description

Function that initializes the variable neighborhood trust region search.

Usage

```
initialize(f, npar, minimize, controls)
```

Arguments

f	A function that computes value, gradient, and Hessian of the function to be optimized and returns them as a named list with elements value, gradient, and hessian.
npar	The number of parameters of f.

`minimize` If `TRUE`, `f` gets minimized. If `FALSE`, maximized.

`controls` Either `NULL` or a named list with the following elements. Missing elements are set to the default values in parentheses.

- `init_runs` (`5`): The number of initial searches.
- `init_min` (`-1`): The minimum argument value for the random initialization.
- `init_max` (`1`): The maximum argument value for the random initialization.
- `init_iterlim` (`20`): The number of iterations for the initial searches.
- `neighborhoods` (`5`): The number of nested neighborhoods.
- `neighbors` (`5`): The number of neighbors in each neighborhood.
- `beta` (`0.05`): A non-negative weight factor to account for the function's curvature in the selection of the neighbors. If `beta = 0`, the curvature is ignored. The higher the value, the higher the probability of selecting a neighbor in the direction of the highest function curvature.
- `iterlim` (`1000`): The maximum number of iterations to be performed before the local search is terminated.
- `tolerance` (`1e-6`): A positive scalar giving the tolerance for comparing different optimal arguments for equality.
- `time_limit` (`NULL`): The time limit in seconds for the algorithm.

Value

A list of

- the list `L` of identified optima which contains lists with
  - `value` and
  - `argument`

  of each identified optimum.
- best initial point `x_best`.

`interruption` *Interrupt local search.*

Description

This function checks if the local search can be interrupted prematurely.

Usage

```
interruption(f, point, L, minimize)
```

Arguments

f	A function that computes value, gradient, and Hessian of the function to be optimized and returns them as a named list with elements value, gradient, and hessian.
point	The current location of the local search.
L	A list of identified optima which contains lists with • value and • argument of each identified optimum.
minimize	If TRUE, f gets minimized. If FALSE, maximized.

Value

TRUE for premature interruption, FALSE if not.

---

local *Perform trust region local search.*

Description

Function that links to trust.

Usage

```
local(f, parinit, minimize, controls, L)
```

Arguments

f	A function that computes value, gradient, and Hessian of the function to be optimized and returns them as a named list with elements value, gradient, and hessian.
parinit	Passed on to trust.
minimize	If TRUE, f gets minimized. If FALSE, maximized.
controls	Either NULL or a named list with the following elements. Missing elements are set to the default values in parentheses. • init_runs (5): The number of initial searches. • init_min (-1): The minimum argument value for the random initialization. • init_max (1): The maximum argument value for the random initialization. • init_iterlim (20): The number of iterations for the initial searches. • neighborhoods (5): The number of nested neighborhoods. • neighbors (5): The number of neighbors in each neighborhood.

- `beta` (`0.05`): A non-negative weight factor to account for the function's curvature in the selection of the neighbors. If `beta = 0`, the curvature is ignored. The higher the value, the higher the probability of selecting a neighbor in the direction of the highest function curvature.
- `iterlim` (`1000`): The maximum number of iterations to be performed before the local search is terminated.
- `tolerance` (`1e-6`): A positive scalar giving the tolerance for comparing different optimal arguments for equality.
- `time_limit` (`NULL`): The time limit in seconds for the algorithm.

`L` A list of identified optima which contains lists with

- `value` and
- `argument`

of each identified optimum.

Value

A list of

- `success`: A boolean, determining wether the local search successfully converged.
- `value`: The value at the point where the local search terminated.
- `argument`: The point where the local search terminated.

`select_neighbors` *Select neighbors.*

Description

Function that selects neighbors around a given point `x`.

Usage

```
select_neighbors(f, x, neighborhood_expansion, controls)
```

Arguments

`f` A function that computes value, gradient, and Hessian of the function to be optimized and returns them as a named list with elements `value`, `gradient`, and `hessian`.

`x` A point in the domain of `f`.

`neighborhood_expansion`
A scaling factor, specifying the expansion of the neighborhood.

`controls` Either `NULL` or a named list with the following elements. Missing elements are set to the default values in parentheses.

- `init_runs` (`5`): The number of initial searches.

- `init_min` (`-1`): The minimum argument value for the random initialization.
- `init_max` (`1`): The maximum argument value for the random initialization.
- `init_iterlim` (`20`): The number of iterations for the initial searches.
- `neighborhoods` (`5`): The number of nested neighborhoods.
- `neighbors` (`5`): The number of neighbors in each neighborhood.
- `beta` (`0.05`): A non-negative weight factor to account for the function's curvature in the selection of the neighbors. If `beta = 0`, the curvature is ignored. The higher the value, the higher the probability of selecting a neighbor in the direction of the highest function curvature.
- `iterlim` (`1000`): The maximum number of iterations to be performed before the local search is terminated.
- `tolerance` (`1e-6`): A positive scalar giving the tolerance for comparing different optimal arguments for equality.
- `time_limit` (`NULL`): The time limit in seconds for the algorithm.

Value

A list points in the domain of `f` which neighbors of `x`.

`unique_optimum` *Check new optimum for uniqueness.*

Description

This function checks if a new optimum `argument` is not yet contained in `L`.

Usage

```
unique_optimum(L, argument, tolerance)
```

Arguments

`L`	A list of identified optima which contains lists with • `value` and • `argument` of each identified optimum.
`argument`	The argument of a candidate optimum.
`tolerance`	A non-negative numeric value. For an identified optimum and a candidate optimum, if all coordinate differences are smaller than `tolerance`, they are considered as equal.

Value

A boolean. If `TRUE`, `argument` is not contained in `L`. If `FALSE`, `argument` is already contained in `L`.

vntrs *Variable neighborhood trust region search.*

Description

This function performs variable neighborhood trust region search.

Usage

```
vntrs(f, npar, minimize = TRUE, controls = NULL, quiet = TRUE, seed = NULL)
```

Arguments

f	A function that computes value, gradient, and Hessian of the function to be optimized and returns them as a named list with elements `value`, `gradient`, and `hessian`.
npar	The number of parameters of `f`.
minimize	If `TRUE`, `f` gets minimized. If `FALSE`, maximized.
controls	Either `NULL` or a named list with the following elements. Missing elements are set to the default values in parentheses. • `init_runs` (`5`): The number of initial searches. • `init_min` (`-1`): The minimum argument value for the random initialization. • `init_max` (`1`): The maximum argument value for the random initialization. • `init_iterlim` (`20`): The number of iterations for the initial searches. • `neighborhoods` (`5`): The number of nested neighborhoods. • `neighbors` (`5`): The number of neighbors in each neighborhood. • `beta` (`0.05`): A non-negative weight factor to account for the function's curvature in the selection of the neighbors. If `beta = 0`, the curvature is ignored. The higher the value, the higher the probability of selecting a neighbor in the direction of the highest function curvature. • `iterlim` (`1000`): The maximum number of iterations to be performed before the local search is terminated. • `tolerance` (`1e-6`): A positive scalar giving the tolerance for comparing different optimal arguments for equality. • `time_limit` (`NULL`): The time limit in seconds for the algorithm.
quiet	If `TRUE`, progress messages are suppressed.
seed	Set a seed for the sampling of the random starting points.

Value

A data frame. Each row contains information of an identified optimum. The first `npar` columns `"p1"`,...,`"p<npar>"` store the argument values, the next column `"value"` has the optimal function values and the last column `"global"` contains `TRUE` for global optima and `FALSE` for local optima.

References

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

Examples

```
rosenbrock = function(x) {
  stopifnot(is.numeric(x))
  stopifnot(length(x) == 2)
  f = expression(100 * (x2 - x1^2)^2 + (1 - x1)^2)
  g1 = D(f, "x1")
  g2 = D(f, "x2")
  h11 = D(g1, "x1")
  h12 = D(g1, "x2")
  h22 = D(g2, "x2")
  x1 = x[1]
  x2 = x[2]
  f = eval(f)
  g = c(eval(g1), eval(g2))
  h = rbind(c(eval(h11), eval(h12)), c(eval(h12), eval(h22)))
  list(value = f, gradient = g, hessian = h)
}
vntrs(f = rosenbrock, npar = 2, seed = 1, controls = list(neighborhoods = 1))
```

Index