Package ‘mco’

November 29, 2014

Version 1.0-15.1

Title Multiple Criteria Optimization Algorithms and Related Functions

Description Functions for multiple criteria optimization using genetic algorithms and related test problems

Depends R (>= 3.0.0)

Suggests scatterplot3d, testthat

License GPL-2

URL http://git.p-value.net/p/mco.git

LazyData yes

Author Olaf Mersmann [aut, cre],
Heike Trautmann [ctb],
Detlef Steuer [ctb],
Bernd Bischl [ctb],
Kalyanmoy Deb [cph]

Maintainer Olaf Mersmann <olafm@p-value.net>

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-09-30 08:58:58

R topics documented:

functions ....................................................... 2
generationalDistance ..................................... 4
normalizeFront .............................................. 5
nsga2 ......................................................... 6
paretoFront .................................................. 8

Index 10
functions

MCO test problems

Description
Description

Collection of functions implementing various MCO test problems.

Usage
Usage

belegundu(x)
belegundu.constr(x)
binh1(x)
binh2(x)
binh2.constr(x)
binh3(x)
deb3(x)
fonseca1(x)
fonseca2(x)
gianna(x)
hannel(x)
hannel.constr(x)
hanne2(x)
hanne2.constr(x)
hanne3(x)
hanne3.constr(x)
hanne4(x)
hanne4.constr(x)
hanne5(x)
hanne5.constr(x)
jimenez(x)
jimenez.constr(x)
vnt(x)
zdt1(x)
zdt2(x)
zdt3(x)

Arguments
Arguments

x Input vector

Value
Value

Function value.

Author(s)
Author(s)

Heike Trautmann <trautmann@statistik.tu-dortmund.de>, Detlef Steuer <steuer@hsu-hamburg.de>
and Olaf Mersmann <olafm@statistik.tu-dortmund.de>
## Examples

```r
## Not run:
ns2a(belegundu, 2, 2,
  constraints=belegundu.constr, cdim=2,
  lower.bounds=c(0, 0), upper.bounds=c(5, 3))

ns2a(binh1, 2, 2,
  lower.bounds=c(-5, -5), upper.bounds=c(10, 10))
ns2a(binh2, 2, 2,
  lower.bounds=c(0, 0), upper.bounds=c(5, 3),
  constraints=binh2.constr, cdim=2)

ns2a(binh3, 2, 3,
  lower.bounds=c(10e-6, 10e-6), upper.bounds=c(10e6, 10e6))

ns2a(deb3, 2, 2,
  lower.bounds=c(0, 0), upper.bounds=c(1, 1),
  generations=500)

ns2a(fonseca1, 2, 2,
  lower.bounds=c(-100, -100), upper.bounds=c(100, 100))

ns2a(fonseca2, 2, 2,
  lower.bounds=c(-4, -4), upper.bounds=c(4, 4))

ns2a(gianna, 1, 2,
  lower.bounds=5, upper.bounds=10)

ns2a(hanne1, 2, 2,
  lower.bounds=c(0, 0), upper.bounds=c(10, 10),
  constraints=hanne1.constr, cdim=1)

ns2a(hanne2, 2, 2,
  lower.bounds=c(0, 0), upper.bounds=c(10, 10),
  constraints=hanne2.constr, cdim=1)

ns2a(hanne3, 2, 2,
  lower.bounds=c(0, 0), upper.bounds=c(10, 10),
  constraints=hanne3.constr, cdim=1)

ns2a(hanne4, 2, 2,
  lower.bounds=c(0, 0), upper.bounds=c(10, 10),
  constraints=hanne4.constr, cdim=1)

ns2a(hanne5, 2, 2,
  lower.bounds=c(0, 0), upper.bounds=c(10, 10),
  constraints=hanne5.constr, cdim=1)

ns2a(jimenez, 2, 2,
  lower.bounds=c(0, 0), upper.bounds=c(100, 100),
  constraints=jimenez.constr, cdim=4)

ns2a(vnt, 2, 3,
  lower.bounds=rep(-3, 2), upper.bounds=rep(3, 2))

ns2a(zdt1, 30, 2,
  lower.bounds=rep(-3, 30), upper.bounds=rep(3, 30))
```

```
generationalDistance

generationalDistance = Quality measures for MCO solutions

Description

Functions to evaluate the quality of the estimated pareto front.

Usage

generationalDistance(x, o)
generalizedSpread(x, o)
epsilonIndicator(x, o)
dominatedHypervolume(x, ref)

Arguments

x Estimated pareto front or an object which has a paretoFront method
o True pareto front or an object which has a paretoFront method
ref Reference point (may be omitted).

Details

Instead of the pareto front, one can also pass an object for which a paretoFront method exists to both methods.

For dominatedHypervolume, if no reference point is given, the maximum in each dimension is used as the reference point.

Value

The respective quality measure.

Note

This code uses version 1.3 of the hypervolume code available from http://iridia.ulb.ac.be/~manuel/hypervolume. For a description of the algorithm see

normalizeFront

**Author(s)**

Heike Trautmann <trautmann@statistik.uni-dortmund.de>, Detlef Steuer <steuer@hsu-hamburg.de> and Olaf Mersmann <olafm@statistik.uni-dortmund.de>

**References**


**Examples**

```r
## estimate true front:
## not run:

## end(Not run)
```

---

**normalizeFront**

Normalize a pareto front

**Description**

Rescales a pareto front to be in the unit hypercube

**Usage**

`normalizeFront(front, minval, maxval)`

**Arguments**

- `front`: Matrix containing the pareto front
- `minval`: Vector containing the minimum value of each objective. May be omitted.
- `maxval`: Vector containing the maximum value of each objective. May be omitted.
**Value**

Matrix containing the rescaled pareto front.

**Author(s)**

Heike Trautmann <trautmann@statistik.uni-dortmund.de>, Detlef Steuer <steuer@hsu-hamburg.de> and Olaf Mersmann <olafm@statistik.uni-dortmund.de>

---

**nsga2**

**NSGA II MOEA**

**Description**

The NSGA-II algorithm minimizes a multidimensional function to approximate its Pareto front and Pareto set. It does this by successive sampling of the search space, each such sample is called a population. The number of samples taken is governed by the generations parameter, the size of the sample by the popsize parameter. Each population is obtained by creating so called offspring search points from the best individuals in the previous population. The best individuals are calculated by non-dominated sorting breaking ties using the crowding distance. The total number of function evaluations used is

\[ n_{val} = \text{popsize} \times (\text{generations} + 1) \]

when generations is a single number and

\[ n_{val} = \text{popsize} \times (\max(\text{generations}) + 1) \]

when generations is a vector of numbers. Note the additional generation of evaluations in the above equation. These stem from the initial population which must be evaluated before the algorithm can start evolving new individuals.

While the algorithm supports unbounded minimization, it will throw a warning and best results are obtained when a sensible upper and lower bound are given. No attempt is made to find such a sensible region of interest, instead if any element of the upper or lower bound is infinite, it is replaced with a very large number (currently +/-4.49423283715579e+307).

**Usage**

```r
nsga2(fn, idim, odim, ..., 
  constraints = NULL, cdim = 0, 
  lower.bounds = rep(-Inf, idim), upper.bounds = rep(Inf, idim), 
  popsize = 100, generations = 100, 
  cprob = 0.7, cdist = 5, 
  mprob = 0.2, mdist = 10, 
  vectorized=FALSE)
```

**Arguments**

- **fn** Function to be minimized
- **idim** Input dimension
- **odim** Output dimension
... Arguments passed through to 'fn'

c constraints Constraint function
cdim constraint dimension
lower.bounds Lower bound of parameters
upper.bounds Upper bound of parameters
popsize Size of population
generations Number of generations to breed. If a vector, then the result will contain the population at each given generation.
cprob Crossover probability
cdist Crossover distribution index
mprob Mutation probability
mdist Mutation distribution index
vectorized If TRUE, the objective and constraint functions must be vectorized, i.e. accept a matrix instead of a vector and return a matrix instead of a vector. The matrix is structured such that one individual parameter combination is contained in each row (the matrix has shape popsize * idim) and each objective is stored in a row of the returned matrix (the returned matrix must have shape odim * popsize).

A vectorized of a function fn should behave like apply(x, 1, f) for a population stored in the matrix x.

Value

If generation is an integer, a list describing the final population with components par, value and pareto.optimal. If generations is a vector, a list is returned. The i-th element of the list contains the population after generations[i] generations, this is not necessarily the set of new individuals that were evaluated in this generation. Some of the new individuals might have been eliminated in the selection phase.

Author(s)

Heike Trautmann <trautmann@statistik.uni-dortmund.de>, Detlef Steuer <steuer@hsu-hamburg.de> and Olaf Mersmann <olafm@statistik.uni-dortmund.de>

References


See Also

zdt1 for more examples and a list of multiobjective test functions.

Examples

```r
## Binh 1 problem:
binh1 <- function(x) {
  y <- numeric(2)
  y[1] <- crossprod(x, x)
  y[2] <- crossprod(x - 5, x - 5)
  return (y)
}
```
paretoFront

Description

Extract the pareto front or pareto set from an mco result object.
Filter an mco result and extract the pareto-optimal solutions.

Usage

paretoFront(x, ...)
paretoSet(x, ...)
paretoFilter(x, ...)

Arguments

x matrix or mco result object
... Ignored
**Value**

A matrix containing the pareto front or pareto set.

`paretoFilter` returns those values in `x` which are not dominated by any other solution.

**Author(s)**

Heike Trautmann <trautmann@statistik.uni-dortmund.de>, Detlef Steuer <steuer@hsu-hamburg.de> and Olaf Mersmann <olafm@statistik.uni-dortmund.de>
Index

*Topic optimize*

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>belegundu</td>
<td>2</td>
</tr>
<tr>
<td>binh1</td>
<td>2</td>
</tr>
<tr>
<td>binh2</td>
<td>2</td>
</tr>
<tr>
<td>binh3</td>
<td>2</td>
</tr>
<tr>
<td>deb3</td>
<td>2</td>
</tr>
</tbody>
</table>

**dominatedHypervolume**

<table>
<thead>
<tr>
<th>Measure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(generationalDistance)</td>
<td>4</td>
</tr>
</tbody>
</table>

**epsilonIndicator**

<table>
<thead>
<tr>
<th>Measure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>(generationalDistance)</td>
<td>4</td>
</tr>
</tbody>
</table>

**fonseca1**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>fonseca2</td>
<td>2</td>
</tr>
</tbody>
</table>

**functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>generalizedSpread</td>
<td>4</td>
</tr>
<tr>
<td>(generationalDistance)</td>
<td>4</td>
</tr>
<tr>
<td>gianna</td>
<td>2</td>
</tr>
</tbody>
</table>

**hanne1**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>hanne2</td>
<td>2</td>
</tr>
<tr>
<td>hanne3</td>
<td>2</td>
</tr>
<tr>
<td>hanne4</td>
<td>2</td>
</tr>
<tr>
<td>hanne5</td>
<td>2</td>
</tr>
</tbody>
</table>

**hanneT**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>hanneU</td>
<td>2</td>
</tr>
</tbody>
</table>

**jimenez**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>normalizeFront</td>
<td>5</td>
</tr>
<tr>
<td>nsga2</td>
<td>6</td>
</tr>
</tbody>
</table>

**paretoFilter**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>paretoFront</td>
<td>8</td>
</tr>
</tbody>
</table>

**paretoSet**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>paretoFront</td>
<td>8</td>
</tr>
</tbody>
</table>

**vnt**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>vnt</td>
<td>2</td>
</tr>
</tbody>
</table>

**zdt1**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>zdt2</td>
<td>2</td>
</tr>
<tr>
<td>zdt3</td>
<td>2</td>
</tr>
</tbody>
</table>