Package 'mco'

October 8, 2020

Version 1.15.6

Title Multiple Criteria Optimization Algorithms and Related Functions

Description A collection of function to solve multiple criteria optimization problems using genetic algorithms (NSGA-II). Also included is a collection of test functions.

Language en-US

Depends R (>= 3.0.0)

Suggests scatterplot3d, testthat

License GPL-2

URL https://github.com/olafmersmann/mco

Encoding UTF-8

LazyData yes

R topics documented:

`functions` *MCO test problems*

Description

Collection of functions implementing various MCO test problems.

Usage

```
belegundu(x)
belegundu.constr(x)
binh1(x)
binh2(x)
binh2.constr(x)
binh3(x)
deb3(x)
fonseca1(x)
fonseca2(x)
gianna(x)
hanne1(x)
hanne1.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

x	Input vector

Value

Function value.

Author(s)

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

Examples

```
## Not run:
nsga2(belegundu, 2, 2,
      constraints=belegundu.constr, cdim=2,
      lower.bounds=c(0, 0), upper.bounds=c(5, 3))

nsga2(binh1, 2, 2,
      lower.bounds=c(-5, -5), upper.bounds=c(10, 10))
nsga2(binh2, 2, 2,
      lower.bounds=c(0, 0), upper.bounds=c(5, 3),
      constraints=binh2.constr, cdim=2)
```

```
nsga2(binh3, 2, 3,
      lower.bounds=c(10e-6, 10e-6), upper.bounds=c(10e6, 10e6))

nsga2(deb3, 2, 2,
      lower.bounds=c(0, 0), upper.bounds=c(1, 1),
      generations=500)

nsga2(fonseca1, 2, 2,
      lower.bounds=c(-100, -100), upper.bounds=c(100, 100))

nsga2(fonseca2, 2, 2,
      lower.bounds=c(-4, -4), upper.bounds=c(4, 4))

nsga2(gianna, 1, 2,
      lower.bounds=5, upper.bounds=10)

nsga2(hanne1, 2, 2,
      lower.bounds=c(0, 0), upper.bounds=c(10, 10),
      constraints=hanne1.constr, cdim=1)

nsga2(hanne2, 2, 2,
      lower.bounds=c(0, 0), upper.bounds=c(10, 10),
      constraints=hanne2.constr, cdim=1)

nsga2(hanne3, 2, 2,
      lower.bounds=c(0, 0), upper.bounds=c(10, 10),
      constraints=hanne3.constr, cdim=1)

nsga2(hanne4, 2, 2,
      lower.bounds=c(0, 0), upper.bounds=c(10, 10),
      constraints=hanne4.constr, cdim=1)

nsga2(hanne5, 2, 2,
      lower.bounds=c(0, 0), upper.bounds=c(10, 10),
      constraints=hanne5.constr, cdim=1)

nsga2(jimenez, 2, 2,
      lower.bounds=c(0, 0), upper.bounds=c(100, 100),
      constraints=jimenez.constr, cdim=4)

nsga2(vnt, 2, 3,
      lower.bounds=rep(-3, 2), upper.bounds=rep(3, 2))

nsga2(zdt1, 30, 2,
      lower.bounds=rep(0, 30), upper.bounds=rep(1, 30))

nsga2(zdt2, 30, 2,
      lower.bounds=rep(0, 30), upper.bounds=rep(1, 30))

nsga2(zdt3, 30, 2,
      lower.bounds=rep(0, 30), upper.bounds=rep(1, 30))

## End(Not run)
```

`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://lopez-ibanez.eu/hypervolume. For a description of the algorithm see

Carlos M. Fonseca, Luis Paquete, and Manuel Lopez-Ibanez. *An improved dimension-sweep algorithm for the hypervolume indicator*. In IEEE Congress on Evolutionary Computation, pages 1157-1163, Vancouver, Canada, July 2006.

Author(s)

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

References

Carlos M. Fonseca, Luis Paquete, and Manuel Lopez-Ibanez. *An improved dimension-sweep algorithm for the hypervolume indicator*. In IEEE Congress on Evolutionary Computation, pages 1157-1163, Vancouver, Canada, July 2006.

Nicola Beume, Carlos M. Fonseca, Manuel Lopez-Ibanez, Luis Paquete, and J. Vahrenhold. *On the complexity of computing the hypervolume indicator*. IEEE Transactions on Evolutionary Computation, 13(5):1075-1082, 2009.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., and Grunert da Fonseca, V (2003): *Performance Assessment of Multiobjective Optimizers: An Analysis and Review*. IEEE Transactions on Evolutionary Computation, 7(2), 117-132.

Examples

```
## Estimate true front:
## Not run:
tf <- nsga2(fonseca2, 2, 2,
            lower.bounds=c(-4, -4), upper.bounds=c(4, 4),
            popsize=1000, generations=100)
res <- nsga2(fonseca2, 2, 2,
             lower.bounds=c(-4, -4), upper.bounds=c(4, 4),
             popsize=16, generations=c(2, 4, 6, 8, 10, 20, 50))
n <- length(res)
sapply(1:n, function(i) dominatedHypervolume(res[[i]], c(1, 1)))
sapply(1:n, function(i) generationalDistance(res[[i]], tf))
sapply(1:n, function(i) generalizedSpread(res[[i]], tf))
sapply(1:n, function(i) epsilonIndicator(res[[i]], tf))

## 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

$$n_eval = popsize * (generations + 1)$$

when `generations` is a single number and

$$n_eval = popsize * (max(generations) + 1)$$

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 replace with a very large number (currently +/-4.49423283715579e+307).

Usage

```
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'
`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

Deb, K., Pratap, A., and Agarwal, S.. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6 (8)** (2002), 182-197.

See Also

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

Examples

```
## Binh 1 problem:
binh1 <- function(x) {
  y <- numeric(2)
  y[1] <- crossprod(x, x)
  y[2] <- crossprod(x - 5, x - 5)
  return (y)
}
r1 <- nsga2(binh1, 2, 2,
           generations=150, popsize=100,
           cprob=0.7, cdist=20,
           mprob=0.2, mdist=20,
           lower.bounds=rep(-5, 2),
           upper.bounds=rep(10, 2))
plot(r1)
```

```
## VNT problem:
vnt <- function(x) {
  y <- numeric(3)
  xn <- crossprod(x, x)
  y[1] <- xn/2 + sin(xn);
  y[2] <- (crossprod(c(3, -2), x) + 4)^2/8 + (crossprod(c(1, -1), x) + 1)^2/27 + 15
  y[3] <- 1/(xn + 1) - 1.1*exp(-xn)
  return (y)
}

r2 <- nsga2(vnt, 2, 3,
           generations=150, popsize=100,
           lower.bounds=rep(-3, 2),
           upper.bounds=rep(3, 2))
plot(r2)

## Example using constraints:
## minimize     f(x) = (x[1]^2, x[2]^2)
## subject to   g(x) = (sum(x) - 5) >= 0
f <- function(x) { x^2 }
g <- function(x) { sum(x) - 5 }
res <- nsga2(f, 2, 2, generations=500,
             lower.bounds=c(0, 0), upper.bounds=c(10, 10),
             constraints=g, cdim=1)
opar <-par(mfrow=c(1,2))
plot(res, xlab="y1", ylab="y2", main="Objective space")
plot(res$par, xlab="x1", ylab="x2", main="Parameter space")
par(opar)
```

`paretoFront` *Pareto Front and pareto set getters*

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