Package 'ExpDE'

May 7, 2026

Type Package

Title Modular Differential Evolution for Experimenting with Operators

Version 0.2.0

Depends R (>= 4.4.0)

Imports assertthat (>= 0.2.0)

Suggests smoof

Date 2025-09-20

URL https://github.com/fcampelo/ExpDE

Maintainer Felipe Campelo <f.campelo@bristol.ac.uk>

Description Modular implementation of the Differential Evolution algorithm for experimenting with different types of operators.

License GPL-2

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Felipe Campelo [aut, cre],
Moises Botelho [aut]

Repository CRAN

Date/Publication 2025-09-29 10:00:02 UTC

Contents

`check_stop_criteria` *Stop criteria for DE*

Description

Implements different stop criteria for the ExpDE framework

Usage

```
check_stop_criteria()
```

Value

logical flag indicating whether any stop condition has been reached.

Warning

This routine accesses the parent environment used in the main function `ExpDE()`, which means that changes made in the variables contained in `env` WILL change the original values. DO NOT change anything unless you're absolutely sure of what you're doing.

create_population *Create population*

Description

Creates a new population for the ExpDE framework

Usage

```
create_population(popsize, probpars)
```

Arguments

popsize	population size
probpars	list of named problem parameters (see ExpDE).

Value

A matrix containing the population for the ExpDE

evaluate_population *Evaluate DE population*

Description

Evaluates the DE population on a given objective function.

Usage

```
evaluate_population(probpars, Pop)
```

Arguments

probpars	problem parameters (see ExpDE for details).
Pop	population matrix (each row is a candidate solution, normalized to the [0, 1] interval,)

Value

numeric vector (with length nrow(Pop)) containing the function values of each point in the population.

ExpDE *Experimental Differential Evolution - ExpDE*

Description

Modular implementation of the Differential Evolution Algorithm for the experimental investigation of the effects of different operators on the performance of the algorithm.

Usage

```
ExpDE(
  popsize,
  mutpars = list(name = "mutation_rand", f = 0.2),
  recpars = list(name = "recombination_bin", cr = 0.8, nvecs = 1),
  selpars = list(name = "standard"),
  stopcrit,
  probpars,
  seed = NULL,
  showpars = list(show.iters = "none")
)
```

Arguments

popsize	population size
mutpars	list of named mutation parameters. See Mutation parameters for details.
recpars	list of named recombination parameters. See Recombination parameters for details.
selpars	list of named selection parameters. See Selection parameters for details.
stopcrit	list of named stop criteria parameters. See Stop criteria for details.
probpars	list of named problem parameters. See Problem Description for details.
seed	seed for the random number generator. See Random Seed for details.
showpars	parameters that regulate the echoing of progress indicators See Showpars for details.

Details

This routine is used to launch a differential evolution algorithm for the **minimization** of a given problem instance using different variants of the recombination, mutation and selection operators. The input parameters that describe those operators receive list objects describing the operator variants to be used in a given optimization procedure.

Value

A list object containing the final population (sorted by performance) , the performance vector, and some run statistics.

Mutation Parameters

`mutpars` is used to inform the routine the type of differential mutation to use, as well as any mutation-related parameter values. The current version accepts the following options:

- `mutation_best`
- `mutation_rand`
- `mutation_mean`
- `mutation_none`
- `mutation_wgi`

`mutpars` receives a list object with name field `mutpars$name` (containing the name of the function to be called, e.g., `name = "mutation_rand"`) as well as whatever parameters that function may require/accept (e.g., `mutpars$f = 0.7`, `mutpars$nvecs = 2`, etc.). See the specific documentation of each function for details.

Some examples are provided in the `Examples` section below.

Recombination parameters

As with the mutation parameters, `recpars` is used to define the desired recombination strategy. The current version accepts the following options:

- `recombination_arith`
- `recombination_bin`
- `recombination_blxAlphaBeta` (incl. special cases `blxAlpha` and `flat`)
- `recombination_eigen`
- `recombination_exp`
- `recombination_geo`
- `recombination_lbga`
- `recombination_linear`
- `recombination_mmax`
- `recombination_npoint`
- `recombination_none`
- `recombination_onepoint`
- `recombination_pbest`
- `recombination_sbx`
- `recombination_wright`

`recpars` receives a list object with name field `recpars$name` (containing the name of the function to be called, e.g., `name = "recombination_bin"`) as well as whatever parameters that function may require/accept (e.g., `recpars$cr = 0.8`, `recpars$minchange = TRUE`, etc.). See the specific documentation of each function for details.

Some examples are provided in the `Examples` section below.

Selection parameters

`selpars` follows the same idea as `mutpars` and `recpars`, and is used to define the selection operators. Currently, only the standard DE selection, `selection_standard`, is implemented.

Stop criteria

`stopcrit` is similar to `recpar` and the other list arguments, but with the difference that multiple stop criteria can be defined for the algorithm. The names of the stop criteria to be used are passed in the `stopcrit$names` field, which must contain a character vector. Other parameters to be used for stopping the algorithm (e.g., the maximum number of iterations `stopcrit$maxiter`) can also be included as `stopcrit` fields. Currently implemented criteria are:

- `"stop_maxiter"` (requires additional field `stopcrit$maxiter` = ? with the maximum number of iterations).
- `"stop_maxeval"` (requires additional field `stopcrit$maxevals` = ? with the maximum number of function calls).

See `check_stop_criteria` for details.

Problem description

The `probpars` parameter receives a list with all definitions related to the problem instance to be optimized. There are three required fields in this parameter:

- `probpars$name`, the name of the function that represents the problem to be solved.
- `probpars$xmin`, a vector containing the lower bounds of all optimization variables (i.e., a vector of length M, where M is the dimension of the problem).
- `probpars$xmax`, a vector containing the upper bounds of all optimization variables.

This list can also contain the following optional arguments

- `probpars$matrixEval`, indicates what kind of input is expected by the function provided in `probpars$name`. Valid entries are `"vector"`, `"colMatrix"` and `"rowMatrix"`. Defaults to `probpars$matrixEval = "rowMatrix"`

Important: the objective function routine must receive either a vector or a matrix of vectors to be evaluated in the form of an input parameter named either "x" or "X" or "Pop" (any one of the three is allowed).

Random Seed

The `seed` argument receives the desired seed for the PRNG. This value can be set for reproducibility purposes. The value of this parameter defaults to NULL, in which case the seed is arbitrarily set using `.Random.seed`.

Showpars

showpars is a list containing parameters that control the printed output of ExpDE. Parameter showpars can have the following fields:

- showpars$show.iters = c("dots", "numbers", "none"): type of output. Defaults to "numbers".
- showpars$showevery: positive integer that determines how frequently the routine echoes something to the terminal. Defaults to 1.

Author(s)

Felipe Campelo (<fcampelo@ufmg.br>) and Moises Botelho (<moisesufop@gmail.com>)

References

F. Campelo, M. Botelho, "Experimental Investigation of Recombination Operators for Differential Evolution", Genetic and Evolutionary Computation Conference, July 20-24, 2016, Denver/CO. DOI: 10.1145/2908812.2908852

Examples

```
# DE/rand/1/bin with population 40, F = 0.8 and CR = 0.5
popsize  <- 100
mutpars  <- list(name = "mutation_rand", f = 0.8)
recpars  <- list(name = "recombination_bin", cr = 0.5, minchange = TRUE)
selpars  <- list(name = "selection_standard")
stopcrit <- list(names = "stop_maxiter", maxiter = 100)
probpars <- list(name  = "sphere",
                 xmin = rep(-5.12,10), xmax = rep(5.12,10))
seed <- NULL
showpars <- list(show.iters = "numbers", showevery = 1)
ExpDE(popsize, mutpars, recpars, selpars, stopcrit, probpars, seed, showpars)


# DE/wgi/1/blxAlpha
recpars  <- list(name = "recombination_blxAlphaBeta", alpha = 0.1, beta = 0.1)
mutpars  <- list(name = "mutation_wgi", f = 0.8)
ExpDE(popsize, mutpars, recpars, selpars, stopcrit, probpars)

# DE/best/1/sbx
recpars  <- list(name = "recombination_sbx", eta = 10)
mutpars  <- list(name = "mutation_best", f = 0.6, nvecs = 1)
ExpDE(popsize, mutpars, recpars, selpars, stopcrit, probpars)

# DE/best/1/eigen/bin
recpars  <- list(name = "recombination_eigen",
                 othername = "recombination_bin",
                 cr = 0.5, minchange = TRUE)
showpars <- list(show.iters = "dots", showevery = 10)
stopcrit <- list(names = "stop_maxeval", maxevals = 10000)
ExpDE(popsize, mutpars, recpars, selpars, stopcrit, probpars, seed = 1234)
```

ExpDE2 *Experimental Differential Evolution - ExpDE*

Description

This function is an interface to call the main function ExpDE, usinga method specification generated by gen_methods and a problem specification generated by gen_problems. It is used to provide an easy data-generating process to study the experimental comparisons of algorithms. See the documentation of ExpDE for details of the underlying algorithm.

Usage

```
ExpDE2(method.name, problem.name, seed = NULL)
```

Arguments

method.name	A specific method key name, generated by gen_methods. This must be a character string in the format "DE.[integer].[single letter]".
problem.name	A specific problem key name, generated by gen_problems. This must be a character string in the format "Prob.[integer].[integer]".
seed	positive integer, seed for the random number generator.

Value

A single scalar value representing the final value returned by running the DE method specified by ‘method.name‘ on the problem specified by ‘problem.name‘.

Examples

```
library(ExpDE)
myprobs <- gen_problems(1234567, nprobs=10)
myalgos <- gen_methods(1234567)
onerun  <- ExpDE2(method.name  = "DE.1234567.A",
                  problem.name = "Prob.1234567.6")
```

gen_methods *Generate methods for testing*

Description

Generate a number of random DE configurations.

Usage

```
gen_methods(ID, echo = TRUE)
```

Arguments

ID	a unique positive integer (e.g., a student ID)
echo	Boolean, should the function print its output?

Value

List object with a number of configurations to use with ExpDE. Alternatively, the method names returned by this function can be passed directly to ExpDE2.

---

gen_problems *Generate test problems*

Description

Get a set of test problems for practicing the design and running of comparative experiments using ExpDE as a data-generating process.

Usage

```
gen_problems(ID, nprobs, echo = TRUE)
```

Arguments

ID	a unique positive integer (e.g., a student ID)
nprobs	number of problems to generate
echo	Boolean, should the function print its output?

Value

List object with definitions for nprobs test problems.

---

get_experimental_parameters_activity2 *Get experimental parameters*

Description

Get parameters for practicing the design and running of a factorial experiment using ExpDE as a data-generating process.

Usage

```
get_experimental_parameters_activity2(ID)
```

Arguments

ID	student ID

Value

List object with experimental parameters that need to be used in the design and execution of the experiment.

---

get_experimental_parameters_activity_versionB *Get experimental parameters*

Description

Get parameters for practicing the design and running of a factorial experiment using ExpDE as a data-generating process.

Usage

```
get_experimental_parameters_activity_versionB(ID)
```

Arguments

ID	student ID

Value

List object with experimental parameters that need to be used in the design and execution of the experiment.

---

get_exp_params_1 *Get experimental parameters*

Description

Get parameters for practicing the design and running of a comparative experiment using ExpDE as a data-generating process.

Usage

```
get_exp_params_1(ID, echo = TRUE)
```

Arguments

ID	a unique positive integer (e.g., a student ID)
echo	Boolean, should the function print its output?

Value

List object with experimental parameters that need to be used for the design of the experiment.

`mutation_best` */best mutation for DE*

Description

Implements the "/best/nvecs" mutation for the ExpDE framework

Usage

```
mutation_best(X, mutpars)
```

Arguments

`X`	population matrix
`mutpars`	mutation parameters (see `Mutation parameters` for details)

Value

Matrix `M` containing the mutated population

Mutation Parameters

The `mutpars` parameter contains all parameters required to define the mutation. `mutation_best()` understands the following fields in `mutpars`:

- `f` : scaling factor for difference vector(s).
  Accepts numeric vectors of size 1 or `nvecs`.
- `nvecs` : number of difference vectors to use.
  Accepts `1 <= nvecs <= (nrow(X)/2 - 2)`
  Defaults to 1.

Warning

This routine will search for the performance vector of population `X` (`J`) in the parent environment (using `parent.frame()`. This variable must be defined for `mutation_best()` to work.

References

K. Price, R.M. Storn, J.A. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization", Springer 2005

Author(s)

Felipe Campelo (<fcampelo@ufmg.br>)

`mutation_mean` */mean mutation for DE*

Description

Implements the "/mean/nvecs" mutation for the ExpDE framework

Usage

```
mutation_mean(X, mutpars)
```

Arguments

`X`	population matrix
`mutpars`	mutation parameters (see `Mutation parameters` for details)

Value

Matrix `M` containing the mutated population

Mutation Parameters

The `mutpars` parameter contains all parameters required to define the mutation. `mutation_mean()` understands the following fields in `mutpars`:

- `f` : scaling factor for difference vector(s).
  Accepts numeric vectors of size 1 or `nvecs`.
- `nvecs` : number of difference vectors to use.
  Accepts `1 <= nvecs <= (nrow(X)/2 - 2)`
  Defaults to 1.

References

K. Price, R.M. Storn, J.A. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization", Springer 2005

Author(s)

Felipe Campelo (<fcampelo@ufmg.br>)

mutation_none *NULL mutation for DE*

Description

Implements the "/none" mutation (i.e., no mutation performed) for the ExpDE framework

Usage

```
mutation_none(X, mutpars)
```

Arguments

X	population matrix
mutpars	mutation parameters (see Mutation parameters for details)

Value

@return The same matrix X used as an input.

Mutation Parameters

The mutpars parameter contains all parameters required to define the mutation. mutation_none() requires no fields in this parameter.

mutation_operators *Mutation operators available*

Description

List all available mutation operators in the ExpDE package

Usage

```
mutation_operators()
```

Value

Character vector with the names of all mutation operators

mutation_rand */rand mutation for DE*

Description

Implements the "/rand/nvecs" mutation for the ExpDE framework

Usage

```
mutation_rand(X, mutpars)
```

Arguments

X	population matrix
mutpars	mutation parameters (see Mutation parameters for details)

Value

Matrix M containing the mutated population

Mutation Parameters

The mutpars parameter contains all parameters required to define the mutation. mutation_rand() understands the following fields in mutpars:

- f : scaling factor for difference vector(s).
  Accepts numeric vectors of size 1 or nvecs.
- nvecs : number of difference vectors to use.
  Accepts 1 <= nvecs <= (nrow(X)/2 - 2)
  Defaults to 1.

References

K. Price, R.M. Storn, J.A. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization", Springer 2005

Author(s)

Felipe Campelo (<fcampelo@ufmg.br>)

mutation_wgi */wgi mutation for DE*

Description

Implements the "/wgi/nvecs" mutation (weighted global intermediate) for the ExpDE framework. This variant is based on a recombination strategy known as "weighted global intermediate recombination" (see the References section for details)

Usage

```
mutation_wgi(X, mutpars)
```

Arguments

X	population matrix
mutpars	mutation parameters (see Mutation parameters for details)

Value

Matrix M containing the mutated population

Mutation Parameters

The mutpars parameter contains all parameters required to define the mutation. mutation_wgi() understands the following fields in mutpars:

- f : scaling factor for difference vector(s).
  Accepts numeric vectors of size 1 or nvecs.
- nvecs : number of difference vectors to use.
  Accepts 1 <= nvecs <= (nrow(X)/2 - 2)
  Defaults to 1.

References

D. Arnold, "Weighted multirecombination evolution strategies". Theoretical Computer Science 361(1): 18-37, 2006.

T. Glasmachers, C. Igel, "Uncertainty handling in model selection for support vector machines". Proc. International Conference on Parallel Problem Solving from Nature (PPSN'08), 185-194, 2008.

Author(s)

Felipe Campelo (<fcampelo@ufmg.br>)

print_progress *Print progress of DE*

Description

Echoes the progress of DE to the terminal

Usage

```
print_progress()
```

Parameters

This routine accesses all variables defined in the calling environment using `parent.frame()`, so it does not require any explicit input parameters. However, the calling environment must contain:

- `showpars`: list containing parameters that control the printed output of `moead()`. Parameter `showpars` can have the following fields:
  - `$show.iters = c("dots", "numbers", "none")`: type of output. Defaults to `"numbers"`.
  - `$showevery`: positive integer that determines how frequently the routine echoes something to the terminal. Defaults to `1`.
- `iters()` : counter function that registers the iteration number

recombination_arith *Arithmetic recombination for DE*

Description

Implements the "/arith" (arithmetic) recombination for the ExpDE framework

Usage

```
recombination_arith(X, M, ...)
```

Arguments

`X`	population matrix (original)
`M`	population matrix (mutated)
`...`	optional parameters (unused)

Value

Matrix `U` containing the recombined population

References

F. Herrera, M. Lozano, A. M. Sanchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study", International Journal of Intelligent Systems 18(3) 309-338, 2003.

`recombination_bin` */bin recombination for DE*

Description

Implements the "/bin" (binomial) recombination for the ExpDE framework

Usage

```
recombination_bin(X, M, recpars)
```

Arguments

`X`	population matrix (original)
`M`	population matrix (mutated)
`recpars`	recombination parameters (see `Recombination parameters` for details)

Value

Matrix `U` containing the recombined population

Recombination Parameters

The `recpars` parameter contains all parameters required to define the recombination. `recombination_bin()` understands the following fields in `recpars`:

- `cr` : component-wise probability of using the value in `M`.
  Accepts numeric value `0 < cr <= 1`.
- `minchange` : logical flag to force each new candidate solution to inherit at least one component from its mutated 'parent'.
  Defaults to TRUE

References

K. Price, R.M. Storn, J.A. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization", Springer 2005

`recombination_blxAlphaBeta` *Blend Alpha Beta recombination for DE*

Description

Implements the "/blxAlphaBeta" (Blend Alpha Beta) recombination for the ExpDE framework

Usage

```
recombination_blxAlphaBeta(X, M, recpars)
```

Arguments

`X`	population matrix (original)
`M`	population matrix (mutated)
`recpars`	recombination parameters (see `Recombination parameters` for details)

Details

This routine also implements two special cases:

- BLX-alpha recombination (`blxAlpha`), by setting `recpars$alpha = recpars$beta`);
- Flat recombination (`flat`), by setting `recpars$alpha = recpars$beta = 0`)

Value

Matrix `U` containing the recombined population

Recombination Parameters

The `recpars` parameter contains all parameters required to define the recombination. `recombination_blxAlpha()` understands the following fields in `recpars`:

- `alpha` : extrapolation parameter for 'best' parent vector.
  Accepts real value `0 <= alpha <= 0.5`.
- `beta` : extrapolation parameter for 'worst' parent vector.
  Accepts real value `0 <= beta <= 0.5`.

@section Warning: This recombination operator evaluates the candidate solutions in `M`, which adds an extra `popsize` evaluations per iteration.

References

F. Herrera, M. Lozano, A. M. Sanchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study", International Journal of Intelligent Systems 18(3) 309-338, 2003.

recombination_eigen */eigen recombination for DE*

Description

Implements the "/eigen" (eigenvector-based) recombination for the ExpDE framework

Usage

```
recombination_eigen(X, M, recpars)
```

Arguments

X	population matrix (original)
M	population matrix (mutated)
recpars	recombination parameters (see Recombination parameters for details)

Value

Matrix U containing the recombined population

Recombination Parameters

The recpars parameter contains all parameters required to define the recombination. recombination_eigen() understands the following fields in recpars:

- othername: name of the recombination operator to be applied after the projection in the eigenvector basis
- ... : parameters required (or optional) to the operator defined by recpars$othername

References

Shu-Mei Guo e Chin-Chang Yang, "Enhancing differential evolution utilizing eigenvector-based crossover operator", IEEE Transactions on Evolutionary Computation 19(1):31-49, 2015.

recombination_exp *Exponential recombination for DE*

Description

Implements the "/exp" (exponential) recombination for the ExpDE framework

Usage

```
recombination_exp(X, M, recpars)
```

Arguments

X	population matrix (original)
M	population matrix (mutated)
recpars	recombination parameters (see Recombination parameters for details)

Value

Matrix U containing the recombined population

Recombination Parameters

The recpars parameter contains all parameters required to define the recombination. recombination_exp() understands the following fields in recpars:

- cr : component-wise probability of selection as a cut-point.
  Accepts numeric value 0 < cr <= 1.

References

K. Price, R.M. Storn, J.A. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization", Springer 2005

---

recombination_geo *Geometric recombination for DE*

Description

Implements the "/geo" (geometric) recombination for the ExpDE framework

Usage

```
recombination_geo(X, M, recpars)
```

Arguments

X	population matrix (original)
M	population matrix (mutated)
recpars	recombination parameters (see Recombination parameters for details)

Value

Matrix U containing the recombined population

Recombination Parameters

The recpars parameter contains all parameters required to define the recombination. recombination_geo() understands the following fields in recpars:

- alpha : exponent for geometrical recombination.
  Accepts numeric value 0 <= alpha <= 1 or NULL (in which case a random value is chosen for each recombination).

References

F. Herrera, M. Lozano, A. M. Sanchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study", International Journal of Intelligent Systems 18(3) 309-338, 2003.

---

recombination_lbga *Linear BGA recombination for DE*

Description

Implements the "/lbga" (Linear Breeder Genetic Algorithm) recombination for the ExpDE framework

Usage

```
recombination_lbga(X, M, ...)
```

Arguments

X	population matrix (original)
M	population matrix (mutated)
...	optional parameters (unused)

Value

Matrix U containing the recombined population

Warning

This recombination operator evaluates the candidate solutions in M, which adds an extra popsize evaluations per iteration.

References

F. Herrera, M. Lozano, A. M. Sanchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study", International Journal of Intelligent Systems 18(3) 309-338, 2003.
D. Schlierkamp-voosen , H. Muhlenbein, "Strategy Adaptation by Competing Subpopulations", Proc. Parallel Problem Solving from Nature (PPSN III), 199-208, 1994.

recombination_linear *Linear recombination for DE*

Description

Implements the "/linear" recombination for the ExpDE framework

Usage

```
recombination_linear(X, M, ...)
```

Arguments

X	population matrix (original)
M	population matrix (mutated)
...	optional parameters (unused)

Value

Matrix U containing the recombined population

Warning

This recombination operator evaluates 3*popsize candidate solutions per iteration of the algorithm. The value of the nfe counter and the vector of performance values G are updated in the calling environment.

References

F. Herrera, M. Lozano, A. M. Sanchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study", International Journal of Intelligent Systems 18(3) 309-338, 2003.
A.H. Wright, "Genetic Algorithms for Real Parameter Optimization", Proc. Foundations of Genetic Algorithms, 205-218, 1991.

recombination_mmax *Min Max Arithmetical recombination for DE*

Description

Implements the "/mmax" (min-max-arithmetical) recombination for the ExpDE framework

Usage

```
recombination_mmax(X, M, recpars = list(lambda = NULL))
```

Arguments

X	population matrix (original)
M	population matrix (mutated)
recpars	recombination parameters (see Recombination parameters for details)

Value

Matrix U containing the recombined population

Warning

This recombination operator evaluates 4*popsize candidate solutions per iteration of the algorithm. The value of the nfe counter and the vector of performance values G are updated in the calling environment.

Recombination Parameters

The recpars parameter contains all parameters required to define the recombination. recombination_pbest() understands the following fields in recpars:

- lambda : Recombination multiplier.
  Optional. Defaults to NULL Accepts numeric value 0 < lambda < 1 or NULL (in which case a random value is independently used for each variable of each recombination pair).

References

F. Herrera, M. Lozano, A. M. Sanchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study", International Journal of Intelligent Systems 18(3):309-338, 2003.
F Herrera, M. Lozano, J.L. Verdegay, "Tuning fuzzy logic controllers by genetic algorithms.", International Journal of Approximate Reasoning 12(3):299-315, 1995.

---

recombination_none *NULL recombination for DE*

Description

Implements the "/none" recombination (i.e., no recombination performed) for the ExpDE framework

Usage

```
recombination_none(X, M, ...)
```

Arguments

X	population matrix (original)
M	population matrix (mutated)
...	optional parameters (unused)

Value

The same matrix M used as an input.

---

recombination_npoint *n-point recombination for DE*

Description

Implements the "/npoint" (n-point) recombination for the ExpDE (as used in the Simple GA).

Usage

```
recombination_npoint(X, M, recpars = list(N = NULL))
```

Arguments

X	population matrix (original)
M	population matrix (mutated)
recpars	recombination parameters (see Recombination parameters for details)

Value

Matrix U containing the recombined population

Recombination Parameters

The recpars parameter contains all parameters required to define the recombination. recombination_npoint() understands the following fields in recpars:

- N : cut number points for crossover.
  Accepts integer value 0 <= N < n, where n is the dimension of the problem; Use N = 0 or N = NULL for randomly choosing a number of cut points.
  Defaults to NULL.

References

L.J. Eshelman, R.A. Caruana, J.D. Schaffer (1989), "Biases in the crossover landscape. In: Proceedings of the Third International Conference on Genetic Algorithms, pp. 10-19, San Francisco, CA, USA.

recombination_onepoint *One-point recombination for DE*

Description

Implements the one-point recombination (as used in the Simple GA).

Usage

```
recombination_onepoint(X, M, recpars = list(K = NULL))
```

Arguments

X	population matrix (original)
M	population matrix (mutated)
recpars	recombination parameters (see Recombination parameters for details)

Value

Matrix U containing the recombined population

Recombination Parameters

The recpars parameter contains all parameters required to define the recombination. recombination_onepoint() understands the following fields in recpars:

- K : cut point for crossover.
  Accepts integer value 0 <= K < n, where n is the dimension of the problem; Use K = 0 or K = NULL for randomly choosing a position for each pair of points.
  Defaults to NULL.

References

F. Herrera, M. Lozano, A. M. Sanchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study", International Journal of Intelligent Systems 18(3) 309-338, 2003.

recombination_operators *Recombination operators available*

Description

List all available recombination operators in the ExpDE package

Usage

```
recombination_operators()
```

Value

Character vector with the names of all recombination operator routines

recombination_pbest *p-Best recombination for DE*

Description

Implements the "/pbest" (p-Best) recombination for the ExpDE framework

Usage

```
recombination_pbest(X, M, recpars)
```

Arguments

`X`	population matrix (original)
`M`	population matrix (mutated)
`recpars`	recombination parameters (see `Recombination parameters` for details)

Value

Matrix `U` containing the recombined population

Recombination Parameters

The `recpars` parameter contains all parameters required to define the recombination. `recombination_pbest()` understands the following fields in `recpars`:

- `cr` : component-wise probability of using the value in `M`.
  Accepts numeric value `0 < cr <= 1`.

Warning

This routine will search for the iterations counter (`t`), the maximum number of iterations (`stopcrit$maxiter`), and the performance vector of population `X` (`J`) in the parent environment (using `parent.frame()`. These variables must be defined for `recombination_pbest()` to work.

References

S.M. Islam, S. Das, S. Ghosh, S. Roy, P.N. Suganthan, "An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization", IEEE. Trans. Systems, Man and Cybernetics - Part B 42(2), 482-500, 2012

`recombination_sbx` */sbx recombination for DE*

Description

Implements the "/sbx" (Simulated Binary) recombination for the ExpDE framework

Usage

```
recombination_sbx(X, M, recpars)
```

Arguments

`X`	population matrix (original)
`M`	population matrix (mutated)
`recpars`	recombination parameters (see `Recombination parameters` for details)

Value

Matrix `U` containing the recombined population

Recombination Parameters

The `recpars` parameter contains all parameters required to define the recombination. `recombination_sbx()` understands the following field in `recpars`:

- `eta` : spread factor.
  Accepts numeric value `eta > 0`.

References

K. Price, R.M. Storn, J.A. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization", Springer 2005
F. Herrera, M. Lozano, A. M. Sanchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study", International Journal of Intelligent Systems 18(3) 309-338, 2003.
K. Deb, R.B. Agrawal, "Simulated binary crossover for continuous search space", Complex Systems (9):115-148, 1995.

recombination_wright *Heuristic Wright recombination for DE*

Description

Implements the "/wright" (Heuristic Wright) recombination for the ExpDE framework.

Usage

```
recombination_wright(X, M, ...)
```

Arguments

X	population matrix (original)
M	population matrix (mutated)
...	optional parameters (unused)

Value

Matrix U containing the recombined population

Warning

This recombination operator evaluates the candidate solutions in M, which adds an extra popsize evaluations per iteration.

References

F. Herrera, M. Lozano, A. M. Sanchez, "A taxonomy for the crossover operator for real-coded genetic algorithms: an experimental study", International Journal of Intelligent Systems 18(3) 309-338, 2003.
A.H. Wright, "Genetic Algorithms for Real Parameter Optimization", Proc. Foundations of Genetic Algorithms, 205-218, 1991.

selection_standard *Standard selection for DE*

Description

Implements the standard selection (greedy) for the ExpDE framework

Usage

```
selection_standard(X, U, J, G)
```

Arguments

X	population matrix (original)
U	population matrix (recombined)
J	performance vector for population X
G	performance vector for population U

Value

list object containing the selected population (Xsel) and its corresponding performance values (Jsel).

Index