Package: smoof (via r-universe)

September 11, 2024

Type Package

Title Single and Multi-Objective Optimization Test Functions

Description Provides generators for a high number of both single- and multi- objective test functions which are frequently used for the benchmarking of (numerical) optimization algorithms. Moreover, it offers a set of convenient functions to generate, plot and work with objective functions.

Version 1.7.0

Date 2023-10-26

Maintainer Jakob Bossek <j.bossek@gmail.com>

URL https://jakobbossek.github.io/smoof/, https://github.com/jakobbossek/smoof

BugReports https://github.com/jakobbossek/smoof/issues

License BSD_2_clause + file LICENSE

Depends ParamHelpers (>= 1.8), checkmate (>= 1.1)

Imports BBmisc (>= 1.6), ggplot2 (>= 2.2.1), Rcpp (>= 0.11.0),

Suggests testthat, MASS, plot3D, plotly, mco, RColorBrewer, reticulate, covr

ByteCompile yes

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.3

Encoding UTF-8

Repository https://jakobbossek.r-universe.dev

RemoteUrl https://github.com/jakobbossek/smoof

RemoteRef HEAD

RemoteSha 4a10acb2052690cb76df042690699505dbd5fad0

Contents

`smoof-package` *smoof: Single and Multi-Objective Optimization test functions.*

Description

The **smoof** R package provides generators for huge set of single- and multi-objective test functions, which are frequently used in the literature to benchmark optimization algorithms. Moreover the package provides methods to create arbitrary objective functions in an object-orientated manner, extract their parameters sets and visualize them graphically.

Some more details

Given a set of criteria $\mathcal{F} = \{f_1, \ldots, f_m\}$ with each $f_i : S \subseteq \mathbf{R}^d \to \mathbf{R}, i = 1, \ldots, m$ being an objective-function, the goal in *Global Optimization (GO)* is to find the best solution $\mathbf{x}^* \in S$. The set S is termed the *set of feasible solutions*. In the case of only a single objective function f, - which we want to restrict ourselves in this brief description - the goal is to minimize the objective, i. e.,

$$\min_{\mathbf{x}} f(\mathbf{x}).$$

Sometimes we may be interested in maximizing the objective function value, but since $min(f(\mathbf{x})) = -\min(-f(\mathbf{x}))$, we do not have to tackle this separately. To compare the robustness of optimization algorithms and to investigate their behaviour in different contexts, a common approach in the literature is to use *artificial benchmarking functions*, which are mostly deterministic, easy to evaluate and given by a closed mathematical formula. A recent survey by Jamil and Yang lists 175 single-objective benchmarking functions in total for global optimization [1]. The **smoof** package offers implementations of a subset of these functions beside some other functions as well as generators for large benchmarking sets like the noiseless BBOB2009 function set [2] or functions based on the multiple peaks model 2 [3].

Sanity checks before evaluation

Almost all continuous smoof function check the input before evaluating the function by default. It checks whether the numeric vector has the expected length, there are no missing values and all values are finite. Not that currently box constraints are not checked though. Evaluating a function millions of times can be slowed down by these checks significantly. Set the option "smoof.check_input_before_evaluation" to `FALSE` to deactivate those checks.

References

[1] Momin Jamil and Xin-She Yang, A literature survey of benchmark functions for global optimization problems, Int. Journal of Mathematical Modelling and Numerical Optimisation, Vol. 4, No. 2, pp. 150-194 (2013). [2] Hansen, N., Finck, S., Ros, R. and Auger, A. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Technical report RR-6829. INRIA, 2009. [3] Simon Wessing, The Multiple Peaks Model 2, Algorithm Engineering Report TR15-2-001, TU Dortmund University, 2015.

`addCountingWrapper` *Return a function which counts its function evaluations.*

Description

This is a counting wrapper for a `smoof_function`, i.e., the returned function first checks whether the given argument is a vector or matrix, saves the number of function evaluations of the wrapped function to compute the function values and finally passes down the argument to the wrapped `smoof_function`.

Usage

```
addCountingWrapper(fn)
```

Arguments

`fn`	[`smoof_function`] Smoof function which should be wrapped.

Value

[`smoof_counting_function`] The function that includes the counting feature.

See Also

`getNumberOfEvaluations`, `resetEvaluationCounter`

Examples

```
fn = makeBBOBFunction(dimensions = 2L, fid = 1L, iid = 1L)
fn = addCountingWrapper(fn)

# We get a value of 0 since the function has not been called yet
print(getNumberOfEvaluations(fn))

# Now call the function 10 times consecutively
for (i in seq(10L)) {
  fn(runif(2))
}
print(getNumberOfEvaluations(fn))
```

```
# Here we pass a (2x5) matrix to the function with each column representing
# one input vector
x = matrix(runif(10), ncol = 5L)
fn(x)
print(getNumberOfEvaluations(fn))
```

---

addLoggingWrapper *Return a function which internally stores x or y values.*

Description

Often it is desired and useful to store the optimization path, i.e., the evaluated function values and/or the parameters. Not all optimization algorithms offer such a trace. This wrapper makes a smoof function handle x/y-values itself.

Usage

```
addLoggingWrapper(fn, logg.x = FALSE, logg.y = TRUE, size = 100L)
```

Arguments

fn	[smoof_function] Smoof function.
logg.x	[logical(1)] Should x-values be logged? Default is FALSE.
logg.y	[logical(1)] Should objective values be logged? Default is TRUE.
size	[integer(1)] Initial size of the internal data structures used for logging. Default is 100. I.e., there is space reserved for 100 function evaluations. In case of an overflow (i.e., more function evaluations than space reserved) the data structures are re-initialized by adding space for another size evaluations. This comes handy if you know the number of function evaluations (or at least an upper bound thereof) a-priori and may serve to reduce the time complexity of logging values.

Value

[smoof_logging_function] The function with an added logging capability.

Note

Logging values, in particular logging x-values, will substantially slow down the evaluation of the function.

Examples

```
# We first build the smoof function and apply the logging wrapper to it
fn = makeSphereFunction(dimensions = 2L)
fn = addLoggingWrapper(fn, logg.x = TRUE)

# We now apply an optimization algorithm to it and the logging wrapper keeps
# track of the evaluated points.
res = optim(fn, par = c(1, 1), method = "Nelder-Mead")

# Extract the logged values
log.res = getLoggedValues(fn)
print(log.res$pars)
print(log.res$obj.vals)
log.res = getLoggedValues(fn, compact = TRUE)
print(log.res)
```

---

autoplot.smoof_function

Generate ggplot2 object.

Description

This function expects a smoof function and returns a ggplot object depicting the function landscape. The output depends highly on the decision space of the smoof function or more technically on the ParamSet of the function. The following distinctions regarding the parameter types are made. In case of a single numeric parameter a simple line plot is drawn. For two numeric parameters or a single numeric vector parameter of length 2 either a contour plot or a heatmap (or a combination of both depending on the choice of additional parameters) is depicted. If there are both up to two numeric and at least one discrete vector parameter, ggplot faceting is used to generate subplots of the above-mentioned types for all combinations of discrete parameters.

Usage

```
## S3 method for class 'smoof_function'
autoplot(
  object,
  ...,
  show.optimum = FALSE,
  main = getName(x),
  render.levels = FALSE,
  render.contours = TRUE,
  log.scale = FALSE,
  length.out = 50L
)
```

Arguments

`object`	[`smoof_function`] Objective function.
`...`	[any] Not used.
`show.optimum`	[`logical(1)`] If the function has a known global optimum, should its location be plotted by a point or multiple points in case of multiple global optima? Default is `FALSE`.
`main`	[`character(1L)`] Plot title. Default is the name of the smoof function.
`render.levels`	[`logical(1)`] For 2D numeric functions only: Should an image map be plotted? Default is `FALSE`.
`render.contours`	[`logical(1)`] For 2D numeric functions only: Should contour lines be plotted? Default is `TRUE`.
`log.scale`	[`logical(1)`] Should the z-axis be plotted on log-scale? Default is `FALSE`.
`length.out`	[`integer(1)`] Desired length of the sequence of equidistant values generated for numeric parameters. Higher values lead to more smooth resolution in particular if `render.levels` is `TRUE`. Avoid using a very high value here especially if the function at hand has many parameters. Default is 50.

Value

[ggplot]

Note

Keep in mind, that the plots for mixed parameter spaces may be very large and computationally expensive if the number of possible discrete parameter values is large. I.e., if we have d discrete parameter with each n_1, n_2, ..., n_d possible values we end up with n_1 x n_2 x ... x n_d subplots.

Examples

```
library(ggplot2)

# Simple 2D contour plot with activated heatmap for the Himmelblau function
fn = makeHimmelblauFunction()
print(autoplot(fn))
print(autoplot(fn, render.levels = TRUE, render.contours = FALSE))
print(autoplot(fn, show.optimum = TRUE))

# Now we create 4D function with a mixed decision space (two numeric, one discrete,
# and one logical parameter)
fn.mixed = makeSingleObjectiveFunction(
```

```
  name = "4d SOO function",
  fn = function(x) {
    if (x$disc1 == "a") {
      (x$x1^2 + x$x2^2) + 10 * as.numeric(x$logic)
    } else {
      x$x1 + x$x2 - 10 * as.numeric(x$logic)
    }
  },
  has.simple.signature = FALSE,
  par.set = makeParamSet(
    makeNumericParam("x1", lower = -5, upper = 5),
    makeNumericParam("x2", lower = -3, upper = 3),
    makeDiscreteParam("disc1", values = c("a", "b")),
    makeLogicalParam("logic")
  )
)
pl = autoplot(fn.mixed)
print(pl)

# Since autoplot returns a ggplot object we can modify it, e.g., add a title
# or hide the legend
pl + ggtitle("My fancy function") + theme(legend.position = "none")
```

`computeExpectedRunningTime` *Compute the Expected Running Time (ERT) performance measure.*

Description

The functions can be called in two different ways

- 1. Pass a vector of function evaluations and a logical vector which indicates which runs were successful (see details).
- 2. Pass a vector of function evaluation, a vector of reached target values and a single target value. In this case the logical vector of option 1. is computed internally.

Usage

```
computeExpectedRunningTime(
  fun.evals,
  fun.success.runs = NULL,
  fun.reached.target.values = NULL,
  fun.target.value = NULL,
  penalty.value = Inf
)
```

Arguments

`fun.evals` [numeric]
Vector containing the number of function evaluations.

`fun.success.runs`
[logical]
Boolean vector indicating which algorithm runs were successful, i.e., which runs reached the desired target value. Default is NULL.

`fun.reached.target.values`
[numeric | NULL]
Numeric vector with the objective values reached in the runs. Default is NULL.

`fun.target.value`
[numeric(1) | NULL]
Target value which shall be reached. Default is NULL.

`penalty.value` [numeric(1)]
Penalty value which should be returned if none of the algorithm runs was successful. Default is Inf.

Details

The Expected Running Time (ERT) is one of the most popular performance measures in optimization. It is defined as the expected number of function evaluations needed to reach a given precision level, i.e., to reach a certain objective value for the first time.

Value

[numeric(1)] Estimated Expected Running Time.

References

A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005), pages 1777-1784, 2005.

---

conversion *Conversion between minimization and maximization problems.*

Description

We can minimize f by maximizing -f. The majority of predefined objective functions in **smoof** should be minimized by default. However, there are a handful of functions, e.g., Keane or Alpine02, which shall be maximized by default. For benchmarking studies it might be beneficial to inverse the direction. The functions `convertToMaximization` and `convertToMinimization` do exactly that, keeping the attributes.

Usage

```
convertToMaximization(fn)

convertToMinimization(fn)
```

Arguments

fn	[smoof_function] Smoof function.

Value

[smoof_function] Converted smoof function

Note

Both functions will quit with an error if multi-objective functions are passed.

Examples

```
# create a function which should be minimized by default
fn = makeSphereFunction(1L)
print(shouldBeMinimized(fn))
# Now invert the objective direction ...
fn2 = convertToMaximization(fn)
# and invert it again
fn3 = convertToMinimization(fn2)
# Now to convince ourselves we render some plots
opar = par(mfrow = c(1, 3))
plot(fn)
plot(fn2)
plot(fn3)
par(opar)
```

---

doesCountEvaluations *Check whether the function is counting its function evaluations.*

Description

In this case, the function is of type smoof_counting_function or it is further wrapped by another wrapper. This function then checks recursively, if there is a counting wrapper.

Usage

```
doesCountEvaluations(object)
```

Arguments

`object`	[any] Arbitrary R object.

Value

`logical(1)` `TRUE` if the function counts its evaluations, `FALSE` otherwise.

See Also

`addCountingWrapper`

`exportNKFunction` *Export/import (rM)NK-landscapes*

Description

NK-landscapes and rMNK-landscapes are randomly generated combinatorial structures. In contrast to continuous benchmark function it thus makes perfect sense to store the landscape definitions in a text-based format.

Usage

```
exportNKFunction(x, path)

importNKFunction(path)
```

Arguments

`x`	[`smoof_function`] NK-landscape of rMNK-landscape.
`path`	[`character(1)`] Path to file.

Details

The format uses two comment lines with basic information like the package version, the date of storage etc. The third line contains ρ, M and N separated by a single-whitespace. Following that follow epistatic links from which the number of epistatic links can be attracted. There are $M * N$ lines for a MNK-landscape with M objectives and input dimension `N`. The first N lines contain the links for the first objective an so on. Following that the tabular values follow in the same manner. For every position $i = 1, \ldots, N$ there is a line with 2^{K_i+1} values.

Note: `exportNKFunction` overwrites existing files without asking.

Value

Silently returns `TRUE` on success.

See Also

importNKFunction

Other nk_landscapes: makeMNKFunction(), makeNKFunction()

---

filterFunctionsByTags *Get a list of implemented test functions with specific tags.*

Description

Single objective functions can be tagged, e.g., as unimodal. Searching for all functions with a specific tag by hand is tedious. The filterFunctionsByTags function helps to filter all single objective smoof functions.

Usage

```
filterFunctionsByTags(tags, or = FALSE)
```

Arguments

tags	[character] Character vector of tags. All available tags can be determined with a call to getAvailableTags.
or	[logical(1)] Should all tags be assigned to the function or are single tags allowed as well? Default is FALSE.

Value

[character] Named vector of function names with the given tags.

Examples

```
# list all functions which are unimodal
filterFunctionsByTags("unimodal")
# list all functions which are both unimodal and separable
filterFunctionsByTags(c("unimodal", "separable"))
# list all functions which are unimodal or separable
filterFunctionsByTags(c("multimodal", "separable"), or = TRUE)
```

getAvailableTags *Returns a character vector of possible function tags.*

Description

Test function are frequently distinguished by characteristic high-level properties, e.g., uni-modal or multi-modal, continuous or discontinuous, separable or non-separable. The **smoof** package offers the possibility to associate a set of properties, termed "tags" to a `smoof_function`. This helper function returns a character vector of all possible tags.

Usage

```
getAvailableTags()
```

Value

[`character`] Character vector of all the possible tags

getDescription *Returns the description of the function.*

Description

Returns the description of the function.

Usage

```
getDescription(fn)
```

Arguments

`fn`	[`smoof_function`] Objective function.

Value

[`character(1)`] A character string representing the description of the function.

getGlobalOptimum *Returns the global optimum and its value.*

Description

Returns the global optimum and its value.

Usage

```
getGlobalOptimum(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[list] List containing the following entries:

- param [list] Named list of parameter value(s).
- value [numeric(1)] Optimal value.
- is.minimum [logical(1)] Is the global optimum a minimum or maximum?

Note

Keep in mind, that this method makes sense only for single-objective target function.

getID *Returns the ID / short name of the function.*

Description

Returns the ID / short name of the function or NA if no ID is set.

Usage

```
getID(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[character(1)] ID / short name or NA

getLocalOptimum *Returns the local optima of a single objective smoof function.*

Description

This function returns the parameters and objective values of all local optima (including the global one).

Usage

```
getLocalOptimum(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[list] List containing the following entries:

- param [list]List of parameter values per local optima.
- value [list]List of objective values per local optima.
- is.minimum [logical(1)]Are the local optima minima or maxima?

Note

Keep in mind, that this method makes sense only for single-objective target functions.

getLoggedValues *Extracts the logged values of a function wrapped by a logging wrapper.*

Description

Extracts the logged values of a function wrapped by a logging wrapper.

Usage

```
getLoggedValues(fn, compact = FALSE)
```

Arguments

fn	[wrapped_smoof_function] Wrapped smoof function.
compact	[logical(1)] Wrap all logged values in a single data frame? Default is FALSE.

Value

[list || data.frame] If compact is TRUE, a single data frame. Otherwise the function returns a list containing the following values:

pars Data frame of parameter values, i.e., x-values or the empty data frame if x-values were not logged.

obj.vals Numeric vector of objective values in the single-objective case respectively a matrix of objective values for multi-objective functions.

See Also

addLoggingWrapper

---

getLowerBoxConstraints *Returns lower box constraints for a Smoof function.*

Description

Returns lower box constraints for a Smoof function.

Usage

```
getLowerBoxConstraints(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[numeric] Numeric vector representing the lower box constraints

---

getMeanFunction *Returns the true mean function in the noisy case.*

Description

Returns the true mean function in the noisy case.

Usage

```
getMeanFunction(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[function] True mean function in the noisy case.

---

getName *Returns the name of the function.*

Description

Returns the name of the function.

Usage

```
getName(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[character(1)] The name of the function.

---

getNumberOfEvaluations *Returns the number of function evaluations performed by the wrapped smoof_function.*

Description

Returns the number of function evaluations performed by the wrapped smoof_function.

Usage

```
getNumberOfEvaluations(fn)
```

Arguments

fn	[smoof_counting_function] Wrapped smoof_function.

Value

[integer(1)] The number of function evaluations.

---

getNumberOfObjectives *Determines the number of objectives.*

Description

Determines the number of objectives.

Usage

```
getNumberOfObjectives(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[integer(1)] The number of objectives.

---

getNumberOfParameters *Determines the number of parameters.*

Description

Determines the number of parameters.

Usage

```
getNumberOfParameters(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[integer(1)] The number of parameters.

getOptimaDf *Get Data Frame of optima.*

Description

This function returns data frame of global / local optima for visualization using ggplot.

Usage

```
getOptimaDf(fn)
```

Arguments

fn	[smoof_function] Smoof function.

Value

[data.frame] A data frame containing information about the optima.

getParamSet *Get parameter set.*

Description

Each smoof function contains a parameter set of type ParamSet assigned to it, which describes types and bounds of the function parameters. This function returns the parameter set.

Arguments

fn	[smoof_function] Objective function.

Value

[ParamSet]

Examples

```
fn = makeSphereFunction(3L)
ps = getParamSet(fn)
print(ps)
```

`getRefPoint` *Returns the reference point of a multi-objective function.*

Description

Returns the reference point of a multi-objective function.

Usage

```
getRefPoint(fn)
```

Arguments

`fn`	[`smoof_function`] Objective function.

Value

[`numeric`] The reference point.

Note

Keep in mind, that this method makes sense only for multi-objective target functions.

`getTags` *Returns the vector of associated tags.*

Description

Returns the vector of associated tags.

Usage

```
getTags(fn)
```

Arguments

`fn`	[`smoof_function`] Objective function.

Value

[`character`] Vector of associated tags.

getUpperBoxConstraints *Return upper box constaints.*

Description

Return upper box constaints.

Usage

```
getUpperBoxConstraints(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[numeric]

getWrappedFunction *Extract wrapped function.*

Description

The **smoof** package offers means to let a function log its evaluations or even store to points and function values it has been evaluated on. This is done by wrapping the function with other functions. This helper function extract the wrapped function.

Usage

```
getWrappedFunction(fn, deepest = FALSE)
```

Arguments

fn	[smoof_wrapped_function] Wrapping function.
deepest	[logical(1)] Function may be wrapped with multiple wrappers. If deepest is set to TRUE the function unwraps recursively until the "deepest" wrapped smoof_function is reached. Default is FALSE.

Value

[function] The extracted wrapped function.

Note

If this function is applied to a simple `smoof_function`, the `smoof_function` itself is returned.

See Also

`addCountingWrapper`, `addLoggingWrapper`

---

hasBoxConstraints *Checks whether the objective function has box constraints.*

Description

Checks whether the objective function has box constraints.

Usage

```
hasBoxConstraints(fn)
```

Arguments

`fn`	[`smoof_function`] Objective function.

Value

[`logical(1)`]

---

hasConstraints *Checks whether the objective function has constraints.*

Description

Checks whether the objective function has constraints.

Usage

```
hasConstraints(fn)
```

Arguments

`fn`	[`smoof_function`] Objective function.

Value

[`logical(1)`] `TRUE` if the function has constraints, `FALSE` otherwise.

hasGlobalOptimum *Checks whether the global optimum is known.*

Description

Checks whether the global optimum is known.

Usage

```
hasGlobalOptimum(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[logical(1)] TRUE if the global optimum is known, FALSE otherwise.

hasLocalOptimum *Checks whether local optima are known.*

Description

Checks whether local optima are known.

Usage

```
hasLocalOptimum(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[logical(1)]

hasOtherConstraints *Checks whether the objective function has other constraints.*

Description

Checks whether the objective function has other constraints.

Usage

```
hasOtherConstraints(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[logical(1)]

hasTags *Checks if the function has assigned special tags.*

Description

Each single-objective smoof function has tags assigned to it (see getAvailableTags). This little helper returns a vector of assigned tags from a smoof function.

Usage

```
hasTags(fn, tags)
```

Arguments

fn	[smoof_function] Function of smoof_function, a smoof_generator or a string.
tags	[character] Vector of tags/properties to check fn for.

Value

[logical(1)] Logical vector indicating the presence of specified tags.

isMultiobjective *Checks whether the given function is multi-objective.*

Description

Checks whether the given function is multi-objective.

Usage

```
isMultiobjective(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[logical(1)] TRUE if function is multi-objective, FALSE otherwise.

isNoisy *Checks whether the given function is noisy.*

Description

Checks whether the given function is noisy.

Usage

```
isNoisy(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[logical(1)] TRUE if the function is noisy, FALSE otherwise.

isSingleobjective *Checks whether the given function is single-objective.*

Description

Checks whether the given function is single-objective.

Usage

```
isSingleobjective(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[logical(1)] TRUE if function is single-objective, FALSE otherwise.

isSmoofFunction *Checks whether the given object is a* smoof_function *or a* smoof_wrapped_function.

Description

Checks whether the given object is a smoof_function or a smoof_wrapped_function.

Usage

```
isSmoofFunction(object)
```

Arguments

object	[any] Arbitrary R object.

Value

[logical(1)] TRUE if the object is a SMOOF function or wrapped function, FALSE otherwise.

See Also

addCountingWrapper, addLoggingWrapper

isVectorized *Checks whether the given function accepts "vectorized" input.*

Description

Checks whether the given function accepts "vectorized" input.

Usage

```
isVectorized(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[logical(1)] TRUE if the function accepts vectorized input, FALSE otherwise.

isWrappedSmoofFunction *Checks whether the function is of type* smoof_wrapped_function.

Description

Checks whether the function is of type smoof_wrapped_function.

Usage

```
isWrappedSmoofFunction(object)
```

Arguments

object	[any] Arbitrary R object.

Value

[logical(1)] TRUE if the object is a SMOOF wrapped function, FALSE otherwise.

makeAckleyFunction *Ackley Function*

Description

Also known as "Ackley's Path Function". Multi-modal test function with its global optimum in the center of the definition space. The implementation is based on the formula

$$f(\mathbf{x}) = -a \cdot \exp\left(-b \cdot \sqrt{\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\right)}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(c \cdot \mathbf{x}_i)\right),$$

with $a = 20$, $b = 0.2$ and $c = 2\pi$. The feasible region is given by the box constraints $\mathbf{x}_i \in [-32.768, 32.768]$.

Usage

```
makeAckleyFunction(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Ackley Function.

[smoof_single_objective_function]

References

Ackley, D. H.: A connectionist machine for genetic hillclimbing. Boston: Kluwer Academic Publishers, 1987.

makeAdjimanFunction *Adjiman function*

Description

This two-dimensional multi-modal test function follows the formula

$$f(\mathbf{x}) = \cos(\mathbf{x}_1)\sin(\mathbf{x}_2) - \frac{\mathbf{x}_1}{(\mathbf{x}_2^2 + 1)}$$

with $\mathbf{x}_1 \in [-1, 2], \mathbf{x}_2 \in [2, 1]$.

Usage

```
makeAdjimanFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Adjiman Function.

[smoof_single_objective_function]

References

C. S. Adjiman, S. Sallwig, C. A. Flouda, A. Neumaier, A Global Optimization Method, aBB for General Twice-Differentiable NLPs-1, Theoretical Advances, Computers Chemical Engineering, vol. 22, no. 9, pp. 1137-1158, 1998.

`makeAlpine01Function` *Alpine01 function*

Description

Highly multi-modal single-objective optimization test function. It is defined as

$$f(\mathbf{x}) = \sum_{i=1}^{n} |\mathbf{x}_i \sin(\mathbf{x}_i) + 0.1\mathbf{x}_i|$$

with box constraints $\mathbf{x}_i \in [-10, 10]$ for $i = 1, \ldots, n$.

Usage

```
makeAlpine01Function(dimensions)
```

Arguments

`dimensions`	[integer(1)] Size of corresponding parameter space.

Value

An object of class `SingleObjectiveFunction`, representing the Alpine01 Function.

[smoof_single_objective_function]

References

S. Rahnamyan, H. R. Tizhoosh, N. M. M. Salama, A Novel Population Initialization Method for Accelerating Evolutionary Algorithms, Computers and Mathematics with Applications, vol. 53, no. 10, pp. 1605-1614, 2007.

makeAlpine02Function *Alpine02 function*

Description

Another multi-modal optimization test function. The implementation is based on the formula

$$f(\mathbf{x}) = \prod_{i=1}^{n} \sqrt{\mathbf{x}_i} \sin(\mathbf{x}_i)$$

with $\mathbf{x}_i \in [0, 10]$ for $i = 1, \ldots, n$.

Usage

```
makeAlpine02Function(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Alpine02 Function.

[smoof_single_objective_function]

References

M. Clerc, The Swarm and the Queen, Towards a Deterministic and Adaptive Particle Swarm Optimization, IEEE Congress on Evolutionary Computation, Washington DC, USA, pp. 1951-1957, 1999.

makeAluffiPentiniFunction
Aluffi-Pentini function.

Description

Two-dimensional test function based on the formula

$$f(\mathbf{x}) = 0.25x_1^4 - 0.5x_1^2 + 0.1x_1 + 0.5x_2^2$$

with $\mathbf{x}_1, \mathbf{x}_2 \in [-10, 10]$.

Usage

```
makeAluffiPentiniFunction()
```

Value

[smoof_single_objective_function]

Note

This functions is also know as the Zirilli function.

References

See https://al-roomi.org/benchmarks/unconstrained/2-dimensions/26-aluffi-pentini-s-or-zirilli-s-fun

---

makeBartelsConnFunction *Bartels Conn Function*

Description

The Bartels Conn Function is defined as

$$f(\mathbf{x}) = |\mathbf{x}_1^2 + \mathbf{x}_2^2 + \mathbf{x}_1\mathbf{x}_2| + |\sin(\mathbf{x}_1)| + |\cos(\mathbf{x})|$$

subject to $\mathbf{x}_i \in [-500, 500]$ for $i = 1, 2$.

Usage

```
makeBartelsConnFunction()
```

Value

An object of class SingleObjectiveFunction, representing the Bartels Conn Function.

[smoof_single_objective_function]

---

makeBBOBFunction *Generator for noiseless function set of the real-parameter BBOB.*

Description

Generator for the noiseless function set of the real-parameter Black-Box Optimization Benchmarking (BBOB).

Usage

```
makeBBOBFunction(dimensions, fid, iid)
```

Arguments

dimensions	[integer(1)] Problem dimension. Integer value between 2 and 40.
fid	[integer(1)] Function identifier. Integer value between 1 and 24.
iid	[integer(1)] Instance identifier. Integer value greater than or equal 1.

Value

[smoof_single_objective_function] Noiseless function from the BBOB benchmark.

Note

It is possible to pass a matrix of parameters to the functions, where each column consists of one parameter setting.

References

See the BBOB website for a detailed description of the BBOB functions.

Examples

```
# get the first instance of the 2D Sphere function
fn = makeBBOBFunction(dimensions = 2L, fid = 1L, iid = 1L)
if (require(plot3D)) {
  plot3D(fn, contour = TRUE)
}
```

---

makeBealeFunction *Beale Function*

Description

Multi-modal single-objective test function for optimization. It is based on the mathematic formula

$$f(\mathbf{x}) = (1.5 - \mathbf{x}_1 + \mathbf{x}_1\mathbf{x}_2)^2 + (2.25 - \mathbf{x}_1 + \mathbf{x}_1\mathbf{x}_2^2)^2 + (2.625 - \mathbf{x}_1 + \mathbf{x}_1\mathbf{x}_2^3)^2$$

usually evaluated within the bounds $\mathbf{x}_i \in [-4.5, 4.5], i = 1, 2$. The function has a flat but multi-modal region around the single global optimum and large peaks in the edges of its definition space.

Usage

```
makeBealeFunction()
```

Value

An object of class SingleObjectiveFunction, representing the Beale Function.

[smoof_single_objective_function]

`makeBentCigarFunction` *Bent-Cigar Function*

Description

Scalable test function f with

$$f(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$$

subject to $-100 \leq \mathbf{x}_i \leq 100$ for $i = 1, \ldots, n$.

Usage

```
makeBentCigarFunction(dimensions)
```

Arguments

`dimensions`	`[integer(1)]` Size of corresponding parameter space.

Value

An object of class `SingleObjectiveFunction`, representing the Bent-Cigar Function.

`[smoof_single_objective_function]`

References

See https://al-roomi.org/benchmarks/unconstrained/n-dimensions/164-bent-cigar-function.

`makeBiObjBBOBFunction` *Generate Bi-Objective Function from the Real-Parameter Bi-Objective Black-Box Optimization Benchmarking (BBOB)*

Description

Generator for the function set of the real-parameter Bi-Objective Black-Box Optimization Benchmarking (BBOB) with Function IDs 1-55, as well as its extended version (bbob-biobj-ext) with Function IDs 1-92.

Usage

```
makeBiObjBBOBFunction(dimensions, fid, iid)
```

Arguments

`dimensions`	[integer(1)] Problem dimensions. Integer value between 2 and 40.
`fid`	[integer(1)] Function identifier. Integer value between 1 and 92.
`iid`	[integer(1)] Instance identifier. Integer value greater than or equal 1.

Value

[smoof_multi_objective_function] Bi-objective function from the BBOB benchmark.

Note

Concatenation of single-objective BBOB functions into a bi-objective problem.

References

See the COCO website for a detailed description of the bi-objective BBOB functions. An overview of which pair of single-objective BBOB functions creates which of the 55 bi-objective BBOB functions can be found in the official documentation of the bi-objective BBOB test suite. And a full description of the extended suite with 92 functions can be found in the official documentation of the extended bi-objective BBOB test suite.

Examples

```
# get the fifth instance of the concatenation of the
# 3D versions of sphere and Rosenbrock
fn = makeBiObjBBOBFunction(dimensions = 3L, fid = 4L, iid = 5L)
fn(c(3, -1, 0))
# compare to the output of its single-objective pendants
f1 = makeBBOBFunction(dimensions = 3L, fid = 1L, iid = 2L * 5L + 1L)
f2 = makeBBOBFunction(dimensions = 3L, fid = 8L, iid = 2L * 5L + 2L)
identical(fn(c(3, -1, 0)), c(f1(c(3, -1, 0)), f2(c(3, -1, 0))))
```

---

makeBirdFunction *Bird Function*

Description

Multi-modal single-objective test function. The implementation is based on the mathematical formulation

$$f(\mathbf{x}) = (\mathbf{x}_1 - \mathbf{x}_2)^2 + \exp((1 - \sin(\mathbf{x}_1))^2)\cos(\mathbf{x}_2) + \exp((1 - \cos(\mathbf{x}_2))^2)\sin(\mathbf{x}_1).$$

The function is restricted to two dimensions with $\mathbf{x}_i \in [-2\pi, 2\pi], i = 1, 2$.

Usage

```
makeBirdFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Bird Function.

[`smoof_single_objective_function`]

References

S. K. Mishra, Global Optimization By Differential Evolution and Particle Swarm Methods: Evaluation On Some Benchmark Functions, Munich Research Papers in Economics.

`makeBiSphereFunction` *Bi-objective Sphere function*

Description

Builds and returns the bi-objective Sphere test problem:

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n} \mathbf{x}_i^2, \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{a})^2 \right)$$

where

$$\mathbf{a} \in R^n$$

.

Usage

```
makeBiSphereFunction(dimensions, a = rep(0, dimensions))
```

Arguments

`dimensions`	[`integer(1)`] Number of decision variables.
`a`	[`numeric(1)`] Shift parameter for the second objective. Default is (0,...,0).

Value

[`smoof_multi_objective_function`] Returns an instance of the sphere function as a `smoof_multi_objective_function` object.

makeBK1Function *BK1 function generator*

Description

Generates the BK1 function, a multi-objective optimization test function. The BK1 function is commonly used in benchmarking studies for evaluating the performance of optimization algorithms.

Usage

```
makeBK1Function()
```

Value

[smoof_multi_objective_function] Returns an instance of the BK1 function as a smoof_multi_objective_function object.

References

...

makeBohachevskyN1Function *Bohachevsky function N. 1*

Description

Highly multi-modal single-objective test function. The mathematical formula is given by

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} (\mathbf{x}_i^2 + 2\mathbf{x}_{i+1}^2 - 0.3\cos(3\pi\mathbf{x}_i) - 0.4\cos(4\pi\mathbf{x}_{i+1}) + 0.7)$$

with box-constraints $\mathbf{x}_i \in [-100, 100]$ for $i = 1, \ldots, n$. The multi-modality will be visible by "zooming in" in the plot.

Usage

```
makeBohachevskyN1Function(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class `SingleObjectiveFunction`, representing the Bohachevsky Function.

[smoof_single_objective_function]

References

I. O. Bohachevsky, M. E. Johnson, M. L. Stein, General Simulated Annealing for Function Optimization, Technometrics, vol. 28, no. 3, pp. 209-217, 1986.

`makeBoothFunction` *Booth Function*

Description

This function is based on the formula

$$f(\mathbf{x}) = (\mathbf{x}_1 + 2\mathbf{x}_2 - 7)^2 + (2\mathbf{x}_1 + \mathbf{x}_2 - 5)^2$$

subject to $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makeBoothFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Booth Function.

[smoof_single_objective_function]

`makeBraninFunction` *Branin RCOS function*

Description

Popular 2-dimensional single-objective test function based on the formula:

$$f(\mathbf{x}) = a\left(\mathbf{x}_2 - b\mathbf{x}_1^2 + c\mathbf{x_1} - d\right)^2 + e\left(1 - f\right)\cos(\mathbf{x}_1) + e,$$

where $a = 1, b = \frac{5.1}{4\pi^2}, c = \frac{5}{\pi}, d = 6, e = 10$ and $f = \frac{1}{8\pi}$. The box constraints are given by $\mathbf{x}_1 \in [-5, 10]$ and $\mathbf{x}_2 \in [0, 15]$. The function has three global minima.

Usage

```
makeBraninFunction()
```

Value

An object of class SingleObjectiveFunction, representing the Brainin RCOS Function.

[smoof_single_objective_function]

References

F. H. Branin. Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. IBM J. Res. Dev. 16, 504-522, 1972.

Examples

```
library(ggplot2)
fn = makeBraninFunction()
print(fn)
print(autoplot(fn, show.optimum = TRUE))
```

---

makeBrentFunction *Brent Function*

Description

Single-objective two-dimensional test function. The formula is given as

$$f(\mathbf{x}) = (\mathbf{x}_1 + 10)^2 + (\mathbf{x}_2 + 10)^2 + exp(-\mathbf{x}_1^2 - \mathbf{x}_2^2)$$

subject to the constraints $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makeBrentFunction()
```

Value

An object of class SingleObjectiveFunction, representing the Brent Function.

[smoof_single_objective_function]

References

F. H. Branin Jr., Widely Convergent Method of Finding Multiple Solutions of Simul- taneous Nonlinear Equations, IBM Journal of Research and Development, vol. 16, no. 5, pp. 504-522, 1972.

makeBrownFunction *Brown Function*

Description

This function belongs the the uni-modal single-objective test functions. The function is forumlated as

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\mathbf{x}_i^2)^{(\mathbf{x}_{i+1}+1)} + (\mathbf{x}_{i+1})^{(\mathbf{x}_i+1)}$$

subject to $\mathbf{x}_i \in [-1, 4]$ for $i = 1, \ldots, n$.

Usage

```
makeBrownFunction(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Brown Function.

[smoof_single_objective_function]

References

O. Begambre, J. E. Laier, A hybrid Particle Swarm Optimization - Simplex Algorithm (PSOS) for Structural Damage Identification, Journal of Advances in Engineering Software, vol. 40, no. 9, pp. 883-891, 2009.

makeBukinN2Function *Bukin function N. 2*

Description

Multi-modal, non-scalable, continuous optimization test function given by:

$$f(\mathbf{x}) = 100(\mathbf{x}_2 - 0.01 * \mathbf{x}_1^2 + 1) + 0.01(\mathbf{x}_1 + 10)^2$$

subject to $\mathbf{x}_1 \in [-15, -5]$ and $\mathbf{x}_2 \in [-3, 3]$.

Usage

```
makeBukinN2Function()
```

Value

An object of class SingleObjectiveFunction, representing the Bukin N.2 Function.

[smoof_single_objective_function]

References

Z. K. Silagadze, Finding Two-Dimensional Peaks, Physics of Particles and Nuclei Letters, vol. 4, no. 1, pp. 73-80, 2007.

See Also

makeBukinN4Function, makeBukinN6Function

---

makeBukinN4Function *Bukin function N. 4*

Description

Second continuous Bukin function test function. The formula is given by

$$f(\mathbf{x}) = 100\mathbf{x}_2^2 + 0.01 * ||\mathbf{x}_1 + 10||$$

and the box constraints $mathbf{x}_1 \in [-15, 5], \mathbf{x}_2 \in [-3, 3]$.

Usage

```
makeBukinN4Function()
```

Value

An object of class SingleObjectiveFunction, representing the Bukin N.4 Function.

[smoof_single_objective_function]

References

Z. K. Silagadze, Finding Two-Dimesnional Peaks, Physics of Particles and Nuclei Letters, vol. 4, no. 1, pp. 73-80, 2007.

See Also

makeBukinN2Function, makeBukinN6Function

makeBukinN6Function *Bukin function N. 6*

Description

Beside Bukin N. 2 and N. 4 this is the last "Bukin family" function. It is given by the formula

$$f(\mathbf{x}) = 100\sqrt{||\mathbf{x}_2 - 0.01\mathbf{x}_1^2||} + 0.01||\mathbf{x}_1 + 10||$$

and the box constraints $mathbf x_1 \in [-15, 5], \mathbf{x}_2 \in [-3, 3]$.

Usage

```
makeBukinN6Function()
```

Value

An object of class `SingleObjectiveFunction`, representing the Bukin N.6 Function.

[smoof_single_objective_function]

References

Z. K. Silagadze, Finding Two-Dimensional Peaks, Physics of Particles and Nuclei Letters, vol. 4, no. 1, pp. 73-80, 2007.

See Also

`makeBukinN2Function`, `makeBukinN4Function`

makeCarromTableFunction *Carrom Table Function*

Description

This function is defined as follows:

$$f(\mathbf{x}) = -\frac{1}{30}\left((\cos(\mathbf{x}_1)\exp(|1 - ((\mathbf{x}_1^2 + \mathbf{x}_2^2)^{0.5}/\pi)^2)|\right).$$

The box-constraints are given by $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makeCarromTableFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Carrom Table Function.

[smoof_single_objective_function]

References

S. K. Mishra, Global Optimization By Differential Evolution and Particle Swarm Methods: Evaluation On Some Benchmark Functions, Munich Research Papers in Economics.

`makeChichinadzeFunction`

Chichinadze Function

Description

Continuous single-objective test function f with

$$f(\mathbf{x}) = \mathbf{x}_1^2 - 12\mathbf{x}_1 + 11 + 10\cos(0.5\pi\mathbf{x}_1) + 8\sin(2.5\pi\mathbf{x}_1) - (0.25)^{0.5}\exp(-0.5(\mathbf{x}_2 - 0.5)^2)$$

with $-30 \leq \mathbf{x}_i \leq 30, i = 1, 2$.

Usage

```
makeChichinadzeFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Chichinadze Function.

[smoof_single_objective_function]

`makeChungReynoldsFunction`

Chung Reynolds Function

Description

The definition is given by

$$f(\mathbf{x}) = \left(\sum_{i=1}^{n} \mathbf{x}_i^2\right)^2$$

with box-constraings $\mathbf{x}_i \in [-100, 100], i = 1, \ldots, n$.

Usage

```
makeChungReynoldsFunction(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Chung Reynolds Function.

[smoof_single_objective_function]

References

C. J. Chung, R. G. Reynolds, CAEP: An Evolution-Based Tool for Real-Valued Function Optimization Using Cultural Algorithms, International Journal on Artificial Intelligence Tool, vol. 7, no. 3, pp. 239-291, 1998.

---

makeComplexFunction *Complex function.*

Description

Two-dimensional test function based on the formula

$$f(\mathbf{x}) = (x_1^3 - 3x_1x_2^2 - 1)^2 + (3x_2x_1^2 - x_2^3)^2$$

with $\mathbf{x}_1, \mathbf{x}_2 \in [-2, 2]$.

Usage

```
makeComplexFunction()
```

Value

An object of class SingleObjectiveFunction, representing the Complex Function.

[smoof_single_objective_function]

References

See https://al-roomi.org/benchmarks/unconstrained/2-dimensions/43-complex-function.

`makeCosineMixtureFunction`

Cosine Mixture Function

Description

Single-objective test function based on the formula

$$f(\mathbf{x}) = -0.1 \sum_{i=1}^{n} \cos(5\pi \mathbf{x}_i) - \sum_{i=1}^{n} \mathbf{x}_i^2$$

subject to $\mathbf{x}_i \in [-1, 1]$ for $i = 1, \ldots, n$.

Usage

```
makeCosineMixtureFunction(dimensions)
```

Arguments

`dimensions`	`[integer(1)]` Size of corresponding parameter space.

Value

An object of class `SingleObjectiveFunction`, representing the Cosine Mixture Function.

`[smoof_single_objective_function]`

References

M. M. Ali, C. Khompatraporn, Z. B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems, Journal of Global Optimization, vol. 31, pp. 635-672, 2005.

`makeCrossInTrayFunction`

Cross-In-Tray Function

Description

Non-scalable, two-dimensional test function for numerical optimization with

$$f(\mathbf{x}) = -0.0001 \left(\left| \sin(\mathbf{x}_1 \mathbf{x}_2 \exp(\left| 100 - [(\mathbf{x}_1^2 + \mathbf{x}_2^2)]^{0.5} / \pi \right|) \right| + 1 \right)^{0.1}$$

subject to $\mathbf{x}_i \in [-15, 15]$ for $i = 1, 2$.

Usage

```
makeCrossInTrayFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Cross-In-Tray Function.

[smoof_single_objective_function]

References

S. K. Mishra, Global Optimization By Differential Evolution and Particle Swarm Methods: Evaluation On Some Benchmark Functions, Munich Research Papers in Economics.

---

makeCubeFunction *Cube Function*

Description

The Cube Function is defined as follows:

$$f(\mathbf{x}) = 100(\mathbf{x}_2 - \mathbf{x}_1^3)^2 + (1 - \mathbf{x}_1)^2.$$

The box-constraints are given by $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makeCubeFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Cube Function.

[smoof_single_objective_function]

References

A. Lavi, T. P. Vogel (eds), Recent Advances in Optimization Techniques, John Wliley & Sons, 1966.

`makeDeckkersAartsFunction`
Deckkers-Aarts Function

Description

This continuous single-objective test function is defined by the formula

$$f(\mathbf{x}) = 10^5\mathbf{x}_1^2 + \mathbf{x}_2^2 - (\mathbf{x}_1^2 + \mathbf{x}_2^2)^2 + 10^{-5}(\mathbf{x}_1^2 + \mathbf{x}_2^2)^4$$

with the bounding box $-20 \leq \mathbf{x}_i \leq 20$ for $i = 1, 2$.

Usage

```
makeDeckkersAartsFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Deckkers-Aarts Function.

[smoof_single_objective_function]

References

M. M. Ali, C. Khompatraporn, Z. B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems, Journal of Global Optimization, vol. 31, pp. 635-672, 2005.

`makeDeflectedCorrugatedSpringFunction`
Deflected Corrugated Spring function

Description

Scalable single-objective test function based on the formula

$$f(\mathbf{x}) = 0.1\sum_{i=1}^{n}(x_i - \alpha)^2 - \cos\left(K\sqrt{\sum_{i=1}^{n}(x_i - \alpha)^2}\right)$$

with $\mathbf{x}_i \in [0, 2\alpha], i = 1, \ldots, n$ and $\alpha = K = 5$ by default.

Usage

```
makeDeflectedCorrugatedSpringFunction(dimensions, K = 5, alpha = 5)
```

Arguments

`dimensions`	[`integer(1)`] Size of corresponding parameter space.
`K`	[`numeric(1)`] Parameter. Default is 5.
`alpha`	[`numeric(1)`] Parameter. Default is 5.

Value

An object of class `SingleObjectiveFunction`, representing the Deflected Corrugated Spring Function.

[`smoof_single_objective_function`]

References

See https://al-roomi.org/benchmarks/unconstrained/n-dimensions/238-deflected-corrugated-spring-func

---

`makeDentFunction` *Dent Function Generator*

Description

Builds and returns the bi-objective Dent test problem, which is defined as follows:

$$f(\mathbf{x}) = (f_1(\mathbf{x}_1), f_2(\mathbf{x}))$$

with

$$f_1(\mathbf{x}_1) = 0.5 \left(\sqrt{(}1 + (x_1 + x_2)^2) + \sqrt{(}1 + (x_1 - x_2)^2) + x_1 - x_2 \right) + d$$

and

$$f_1(\mathbf{x}_1) = 0.5 \left(\sqrt{(}1 + (x_1 + x_2)^2) + \sqrt{(}1 + (x_1 - x_2)^2) - x_1 + x_2 \right) + d$$

where $d = \lambda * \exp(-(x_1 - x_2)^2)$ and $\mathbf{x}_i \in [-1.5, 1.5], i = 1, 2$.

Usage

```
makeDentFunction()
```

Value

[`smoof_multi_objective_function`] Returns an instance of the Dent function as a `smoof_multi_objective_function` object.

makeDixonPriceFunction *Dixon-Price Function*

Description

Dixon and Price defined the function

$$f(\mathbf{x}) = (\mathbf{x}_1 - 1)^2 + \sum_{i=1}^{n} i(2\mathbf{x}_i^2 - \mathbf{x}_{i-1})$$

subject to $\mathbf{x}_i \in [-10, 10]$ for $i = 1, \ldots, n$.

Usage

```
makeDixonPriceFunction(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Dixon-Price Function.

[smoof_single_objective_function]

References

L. C. W. Dixon, R. C. Price, The Truncated Newton Method for Sparse Unconstrained Optimisation Using Automatic Differentiation, Journal of Optimization Theory and Applications, vol. 60, no. 2, pp. 261-275, 1989.

makeDoubleSumFunction *Double-Sum Function*

Description

Also known as the rotated hyper-ellipsoid function. The formula is given by

$$f(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} \mathbf{x}_j \right)^2$$

with $\mathbf{x}_i \in [-65.536, 65.536], i = 1, \ldots, n$.

Usage

```
makeDoubleSumFunction(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Double-Sum Function.

[smoof_single_objective_function]

References

H.-P. Schwefel. Evolution and Optimum Seeking. John Wiley & Sons, New York, 1995.

---

makeDTLZ1Function *DTLZ1 Function (family)*

Description

Builds and returns the multi-objective DTLZ1 test problem.

The DTLZ1 test problem is defined as follows:

Minimize $f_1(\mathbf{x}) = \frac{1}{2} x_1 x_2 \cdots x_{M-1}(1 + g(\mathbf{x}_M))$,

Minimize $f_2(\mathbf{x}) = \frac{1}{2} x_1 x_2 \cdots (1 - x_{M-1})(1 + g(\mathbf{x}_M))$,

$\vdots$

Minimize $f_{M-1}(\mathbf{x}) = \frac{1}{2} x_1 (1 - x_2)(1 + g(\mathbf{x}_M))$,

Minimize $f_M(\mathbf{x}) = \frac{1}{2}(1 - x_1)(1 + g(\mathbf{x}_M))$,

with $0 \leq x_i \leq 1$, for $i = 1, 2, \ldots, n$,

where $g(\mathbf{x}_M) = 100 \left[|\mathbf{x}_M| + \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)) \right]$

Usage

```
makeDTLZ1Function(dimensions, n.objectives)
```

Arguments

dimensions	[integer(1)] Number of decision variables.
n.objectives	[integer(1)] Number of objectives.

Value

[smoof_multi_objective_function] Returns an instance of the DTLZ1 family as a smoof_multi_objective_function object.

References

K. Deb and L. Thiele and M. Laumanns and E. Zitzler. Scalable Multi-Objective Optimization Test Problems. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 112, 2001

---

makeDTLZ2Function *DTLZ2 Function (family)*

Description

Builds and returns the multi-objective DTLZ2 test problem.

The DTLZ2 test problem is defined as follows:

Minimize $f_1(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \cdots \cos(x_{M-2}\pi/2) \cos(x_{M-1}\pi/2)$,

Minimize $f_2(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \cdots \cos(x_{M-2}\pi/2) \sin(x_{M-1}\pi/2)$,

Minimize $f_3(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \cdots \sin(x_{M-2}\pi/2)$,

$\vdots$

Minimize $f_{M-1}(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \sin(x_2\pi/2)$,

Minimize $f_M(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \sin(x_1\pi/2)$,

with $0 \leq x_i \leq 1$, for $i = 1, 2, \ldots, n$,

where $g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2$

Usage

```
makeDTLZ2Function(dimensions, n.objectives)
```

Arguments

dimensions	[integer(1)] Number of decision variables.
n.objectives	[integer(1)] Number of objectives.

Value

[smoof_multi_objective_function] Returns an instance of the DTLZ2 family as a smoof_multi_objective_function object.

Note

Note that in case of a bi-objective scenario (`n.objectives = 2L`) DTLZ2 and DTLZ5 are identical.

References

K. Deb and L. Thiele and M. Laumanns and E. Zitzler. Scalable Multi-Objective Optimization Test Problems. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 112, 2001

`makeDTLZ3Function` *DTLZ3 Function (family)*

Description

Builds and returns the multi-objective DTLZ3 test problem. The formula is very similar to the formula of DTLZ2, but it uses the g function of DTLZ1, which introduces a lot of local Pareto-optimal fronts. Thus, this problems is well suited to check the ability of an optimizer to converge to the global Pareto-optimal front.

The DTLZ3 test problem is defined as follows:

Minimize $f_1(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \cdots \cos(x_{M-2}\pi/2) \cos(x_{M-1}\pi/2)$,

Minimize $f_2(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \cdots \cos(x_{M-2}\pi/2) \sin(x_{M-1}\pi/2)$,

Minimize $f_3(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \cos(x_2\pi/2) \cdots \sin(x_{M-2}\pi/2)$,

$\vdots$

Minimize $f_{M-1}(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1\pi/2) \sin(x_2\pi/2)$,

Minimize $f_M(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \sin(x_1\pi/2)$,

with $0 \leq x_i \leq 1$, for $i = 1, 2, \ldots, n$,

where $g(\mathbf{x}_M) = 100 \left[|\mathbf{x}_M| + \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)) \right]$

Usage

```
makeDTLZ3Function(dimensions, n.objectives)
```

Arguments

`dimensions`	`[integer(1)]` Number of decision variables.
`n.objectives`	`[integer(1)]` Number of objectives.

Value

[smoof_multi_objective_function] Returns an instance of the DTLZ3 family as a smoof_multi_objective_function object.

References

K. Deb and L. Thiele and M. Laumanns and E. Zitzler. Scalable Multi-Objective Optimization Test Problems. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 112, 2001

---

makeDTLZ4Function *DTLZ4 Function (family)*

Description

Builds and returns the multi-objective DTLZ4 test problem. It is a slight modification of the DTLZ2 problems by introducing the parameter α. The parameter is used to map $\mathbf{x}_i \rightarrow \mathbf{x}_i^{\alpha}$.

The DTLZ4 test problem is defined as follows:

Minimize $f_1(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1^{\alpha}\pi/2) \cos(x_2^{\alpha}\pi/2) \cdots \cos(x_{M-2}^{\alpha}\pi/2) \cos(x_{M-1}^{\alpha}\pi/2)$,

Minimize $f_2(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1^{\alpha}\pi/2) \cos(x_2^{\alpha}\pi/2) \cdots \cos(x_{M-2}^{\alpha}\pi/2) \sin(x_{M-1}^{\alpha}\pi/2)$,

Minimize $f_3(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1^{\alpha}\pi/2) \cos(x_2^{\alpha}\pi/2) \cdots \sin(x_{M-2}^{\alpha}\pi/2)$,

$\vdots$

Minimize $f_{M-1}(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(x_1^{\alpha}\pi/2) \sin(x_2^{\alpha}\pi/2)$,

Minimize $f_M(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \sin(x_1^{\alpha}\pi/2)$,

with $0 \leq x_i \leq 1$, for $i = 1, 2, \ldots, n$,

where $g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2$

Usage

```
makeDTLZ4Function(dimensions, n.objectives, alpha = 100)
```

Arguments

dimensions	[integer(1)] Number of decision variables.
n.objectives	[integer(1)] Number of objectives.
alpha	[numeric(1)] Optional parameter. Default is 100, which is recommended by Deb et al.

Value

[smoof_multi_objective_function] Returns an instance of the DTLZ4 family as a smoof_multi_objective_function object.

References

K. Deb and L. Thiele and M. Laumanns and E. Zitzler. Scalable Multi-Objective Optimization Test Problems. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 112, 2001

---

makeDTLZ5Function *DTLZ5 Function (family)*

Description

Builds and returns the multi-objective DTLZ5 test problem. This problem can be characterized by a disconnected Pareto-optimal front in the search space. This introduces a new challenge to evolutionary multi-objective optimizers, i.e., to maintain different sub-populations within the search space to cover the entire Pareto-optimal front.

The DTLZ5 test problem is defined as follows:

Minimize $f_1(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(\theta_1 \pi/2) \cos(\theta_2 \pi/2) \cdots \cos(\theta_{M-2} \pi/2) \cos(\theta_{M-1} \pi/2)$,

Minimize $f_2(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(\theta_1 \pi/2) \cos(\theta_2 \pi/2) \cdots \cos(\theta_{M-2} \pi/2) \sin(\theta_{M-1} \pi/2)$,

Minimize $f_3(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(\theta_1 \pi/2) \cos(\theta_2 \pi/2) \cdots \sin(\theta_{M-2} \pi/2)$,

$\vdots$

Minimize $f_{M-1}(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(\theta_1 \pi/2) \sin(\theta_2 \pi/2)$,

Minimize $f_M((1 + g(\mathbf{x}_M)) \sin(\theta_1 \pi/2)$,

with $0 \leq x_i \leq 1$, for $i = 1, 2, \ldots, n$,

where $\theta_i = \frac{\pi}{4(1+g(\mathbf{x}_M))}(1 + 2g(\mathbf{x}_M) x_i)$, for $i = 2, 3, \ldots, (M-1)$

and $g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2$

Usage

```
makeDTLZ5Function(dimensions, n.objectives)
```

Arguments

dimensions	[integer(1)] Number of decision variables.
n.objectives	[integer(1)] Number of objectives.

Value

[smoof_multi_objective_function] Returns an instance of the DTLZ5 family as a smoof_multi_objective_function object.

Note

This problem definition does not exist in the succeeding work of Deb et al. (K. Deb and L. Thiele and M. Laumanns and E. Zitzler (2002). Scalable multi-objective optimization test problems, Proceedings of the IEEE Congress on Evolutionary Computation, pp. 825-830).
Also, note that in case of a bi-objective scenario (n.objectives = 2L) DTLZ2 and DTLZ5 are identical.

References

K. Deb and L. Thiele and M. Laumanns and E. Zitzler. Scalable Multi-Objective Optimization Test Problems. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 112, 2001

---

makeDTLZ6Function *DTLZ6 Function (family)*

Description

Builds and returns the multi-objective DTLZ6 test problem. This problem can be characterized by a disconnected Pareto-optimal front in the search space. This introduces a new challenge to evolutionary multi-objective optimizers, i.e., to maintain different subpopulations within the search space to cover the entire Pareto-optimal front.

The DTLZ6 test problem is defined as follows:

Minimize $f_1(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(\theta_1 \pi/2) \cos(\theta_2 \pi/2) \cdots \cos(\theta_{M-2} \pi/2) \cos(\theta_{M-1} \pi/2)$,

Minimize $f_2(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(\theta_1 \pi/2) \cos(\theta_2 \pi/2) \cdots \cos(\theta_{M-2} \pi/2) \sin(\theta_{M-1} \pi/2)$,

Minimize $f_3(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(\theta_1 \pi/2) \cos(\theta_2 \pi/2) \cdots \sin(\theta_{M-2} \pi/2)$,

$\vdots$

Minimize $f_{M-1}(\mathbf{x}) = (1 + g(\mathbf{x}_M)) \cos(\theta_1 \pi/2) \sin(\theta_2 \pi/2)$,

Minimize $f_M((1 + g(\mathbf{x}_M)) \sin(\theta_1 \pi/2)$,

with $0 \leq x_i \leq 1$, for $i = 1, 2, \ldots, n$,

where $\theta_i = \frac{\pi}{4(1+g(\mathbf{x}_M))}(1 + 2g(\mathbf{x}_M)x_i)$, for $i = 2, 3, \ldots, (M-1)$

and $g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} x_i^{0.1}$

Usage

```
makeDTLZ6Function(dimensions, n.objectives)
```

Arguments

`dimensions`	[integer(1)] Number of decision variables.
`n.objectives`	[integer(1)] Number of objectives.

Value

[smoof_multi_objective_function] Returns an instance of the DTLZ6 family as a smoof_multi_objective_function object.

Note

Attention: Within the succeeding work of Deb et al. (K. Deb and L. Thiele and M. Laumanns and E. Zitzler (2002). Scalable multi-objective optimization test problems, Proceedings of the IEEE Congress on Evolutionary Computation, pp. 825-830) this problem was called DTLZ5.

References

K. Deb and L. Thiele and M. Laumanns and E. Zitzler. Scalable Multi-Objective Optimization Test Problems. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 112, 2001

---

makeDTLZ7Function *DTLZ7 Function (family)*

Description

Builds and returns the multi-objective DTLZ7 test problem. This problem can be characterized by a disconnected Pareto-optimal front in the search space. This introduces a new challenge to evolutionary multi-objective optimizers, i.e., to maintain different sub-populations within the search space to cover the entire Pareto-optimal front.

The DTLZ7 test problem is defined as follows:

Minimize $f_1(\mathbf{x}) = x_1$,

Minimize $f_2(\mathbf{x}) = x_2$,

$\vdots$

Minimize $f_{M-1}(\mathbf{x}) = x_{M-1}$,

Minimize $f_M(\mathbf{x}) = (1 + g(\mathbf{x}_M))h(f_1, f_2, \cdots, f_{M-1}, g)$,

with $0 \leq x_i \leq 1$, for $i = 1, 2, \ldots, n$,

where $g(\mathbf{x}_M) = 1 + \frac{9}{|\mathbf{x}_M|} \sum_{x_i \in \mathbf{x}_M} x_i$

and $h(f_1, f_2, \cdots, f_{M-1}, g) = M - \sum_{i=1}^{M-1} \left[\frac{f_i}{1+g}(1 + sin(3\pi f_i)) \right]$

Usage

```
makeDTLZ7Function(dimensions, n.objectives)
```

Arguments

dimensions	[integer(1)] Number of decision variables.
n.objectives	[integer(1)] Number of objectives.

Value

[smoof_multi_objective_function] Returns an instance of the DTLZ7 family as a smoof_multi_objective_function object.

Note

Attention: Within the succeeding work of Deb et al. (K. Deb and L. Thiele and M. Laumanns and E. Zitzler (2002). Scalable multi-objective optimization test problems, Proceedings of the IEEE Congress on Evolutionary Computation, pp. 825-830) this problem was called DTLZ6.

References

K. Deb and L. Thiele and M. Laumanns and E. Zitzler. Scalable Multi-Objective Optimization Test Problems. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 112, 2001

---

makeEasomFunction *Easom Function*

Description

Uni-modal function with its global optimum in the center of the search space. The attraction area of the global optimum is very small in relation to the search space:

$$f(\mathbf{x}) = -\cos(\mathbf{x}_1)\cos(\mathbf{x}_2)\exp\left(-\left((\mathbf{x}_1 - \pi)^2 + (\mathbf{x}_2 - pi)^2\right)\right)$$

with $\mathbf{x}_i \in [-100, 100], i = 1, 2$.

Usage

```
makeEasomFunction()
```

Value

An object of class SingleObjectiveFunction, representing the Easom Function.

[smoof_single_objective_function]

References

Easom, E. E.: A survey of global optimization techniques. M. Eng. thesis, University of Louisville, Louisville, KY, 1990.

`makeED1Function` *ED1 Function*

Description

Builds and returns the multi-objective ED1 test problem.

The ED1 test problem is defined as follows:

Minimize $f_j(\mathbf{x}) = \frac{1}{r(\mathbf{x})+1} \cdot \tilde{p}(\Theta(\mathbf{X}))$, for $j = 1, \ldots, m$,

with $\mathbf{x} = (x_1, \ldots, x_n)^T$, where $0 \leq x_i \leq 1$, and $\Theta = (\theta_1, \ldots, \theta_{m-1})$, where $0 \leq \theta_j \leq \frac{\pi}{2}$, for $i = 1, \ldots, n$, and $j = 1, \ldots, m-1$.

Moreover $r(\mathbf{X}) = \sqrt{x_m^2 + \ldots, x_n^2}$,

$\tilde{p}_1(\Theta) = \cos(\theta_1)^{2/\gamma}$,

$\tilde{p}_j(\Theta) = (\sin(\theta_1) \cdot \ldots \cdot \sin(\theta_{j-1}) \cdot \cos(\theta_j))^{2/\gamma}$, for $2 \leq j \leq m-1$,

and $\tilde{p}_m(\Theta) = (\sin(\theta_1) \cdot \ldots \cdot \sin(\theta_{m-1}))^{2/\gamma}$.

Usage

```
makeED1Function(dimensions, n.objectives, gamma = 2, theta)
```

Arguments

`dimensions`	[`integer(1)`] Number of decision variables.
`n.objectives`	[`integer(1)`] Number of objectives.
`gamma`	[`numeric(1)`] Optional parameter. Default is 2, which is recommended by Emmerich and Deutz.
`theta`	[`numeric(dimensions)`] Parameter vector, whose components have to be between `0` and `0.5*pi`. The default is `theta = (pi/2) * x` (with `x` being the point from the decision space) as recommended by Emmerich and Deutz.

Value

[`smoof_multi_objective_function`] Returns an instance of the ED1 function as a `smoof_multi_objective_function` object.

References

M. T. M. Emmerich and A. H. Deutz. Test Problems based on Lame Superspheres. Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007), pp. 922-936, Springer, 2007.

`makeED2Function` *ED2 Function*

Description

Builds and returns the multi-objective ED2 test problem.

The ED2 test problem is defined as follows:

Minimize $f_j(\mathbf{x}) = \frac{1}{F_{natmin}(\mathbf{x})+1} \cdot \tilde{p}(\Theta(\mathbf{X}))$, for $j = 1, \ldots, m$,

with $\mathbf{x} = (x_1, \ldots, x_n)^T$, where $0 \leq x_i \leq 1$, and $\Theta = (\theta_1, \ldots, \theta_{m-1})$, where $0 \leq \theta_j \leq \frac{\pi}{2}$, for $i = 1, \ldots, n$, and $j = 1, \ldots, m-1$.

Moreover $F_{natmin}(\mathbf{x}) = b + (r(\mathbf{x}) - a) + 0.5 + 0.5 \cdot (2\pi \cdot (r(\mathbf{x}) - a) + \pi)$

with $a \approx 0.051373$, $b \approx 0.0253235$, and $r(\mathbf{X}) = \sqrt{x_m^2 + \ldots, x_n^2}$, as well as

$\tilde{p}_1(\Theta) = \cos(\theta_1)^{2/\gamma}$,

$\tilde{p}_j(\Theta) = (\sin(\theta_1) \cdot \ldots \cdot \sin(\theta_{j-1}) \cdot \cos(\theta_j))^{2/\gamma}$, for $2 \leq j \leq m-1$,

and $\tilde{p}_m(\Theta) = (\sin(\theta_1) \cdot \ldots \cdot \sin(\theta_{m-1}))^{2/\gamma}$.

Usage

```
makeED2Function(dimensions, n.objectives, gamma = 2, theta)
```

Arguments

`dimensions`	[`integer(1)`] Number of decision variables.
`n.objectives`	[`integer(1)`] Number of objectives.
`gamma`	[`numeric(1)`] Optional parameter. Default is 2, which is recommended by Emmerich and Deutz.
`theta`	[`numeric(dimensions)`] Parameter vector, whose components have to be between `0` and `0.5*pi`. The default is `theta = (pi/2) * x` (with `x` being the point from the decision space) as recommended by Emmerich and Deutz.

Value

[`smoof_multi_objective_function`] Returns an instance of the ED2 function as a `smoof_multi_objective_function` object.

References

M. T. M. Emmerich and A. H. Deutz. Test Problems based on Lame Superspheres. Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007), pp. 922-936, Springer, 2007.

`makeEggCrateFunction` *Egg Crate Function*

Description

This single-objective function follows the definition

$$f(\mathbf{x}) = \mathbf{x}_1^2 + \mathbf{x}_2^2 + 25(\sin^2(\mathbf{x}_1) + \sin^2(\mathbf{x}_2))$$

with $\mathbf{x}_i \in [-5, 5]$ for $i = 1, 2$.

Usage

```
makeEggCrateFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Egg Crate Function.

[smoof_single_objective_function]

`makeEggholderFunction` *Egg Holder function*

Description

The Egg Holder function is a difficult to optimize function based on the definition

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[-(\mathbf{x}_{i+1} + 47) \sin \sqrt{|\mathbf{x}_{i+1} + 0.5\mathbf{x}_i + 47|} - \mathbf{x}_i \sin(\sqrt{|\mathbf{x}_i - (\mathbf{x}_{i+1} - 47)|}) \right]$$

subject to $-512 \leq \mathbf{x}_i \leq 512$ for $i = 1, \ldots, n$.

Usage

```
makeEggholderFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Egg Holder Function.

[smoof_single_objective_function]

makeElAttarVidyasagarDuttaFunction *El-Attar-Vidyasagar-Dutta Function*

Description

This function is based on the formula

$$f(\mathbf{x}) = (\mathbf{x}_1^2 + \mathbf{x}_2 - 10)^2 + (\mathbf{x}_1 + \mathbf{x}_2^2 - 7)^2 + (\mathbf{x}_1^2 + \mathbf{x}_2^3 - 1)^2$$

subject to $\mathbf{x}_i \in [-500, 500], i = 1, 2$.

Usage

```
makeElAttarVidyasagarDuttaFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the El-Attar-Vidyasagar-Dutta Function.

[smoof_single_objective_function]

References

R. A. El-Attar, M. Vidyasagar, S. R. K. Dutta, An Algorithm for II-norm Minimization With Application to Nonlinear II-approximation, SIAM Journal on Numerical Analysis, vol. 16, no. 1, pp. 70-86, 1979.

makeEngvallFunction *Complex function.*

Description

Two-dimensional test function based on the formula

$$f(\mathbf{x}) = (x_1^4 + x_2^4 + 2x_1^2x_2^2 - 4x_1 + 3$$

with $\mathbf{x}_1, \mathbf{x}_2 \in [-2000, 2000]$.

Usage

```
makeEngvallFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Complex Function.

[smoof_single_objective_function]

References

See https://al-roomi.org/benchmarks/unconstrained/2-dimensions/116-engvall-s-function.

`makeExponentialFunction` *Exponential Function*

Description

This scalable test function is based on the definition

$$f(\mathbf{x}) = -\exp\left(-0.5\sum_{i=1}^{n}\mathbf{x}_i^2\right)$$

with the box-constraints $\mathbf{x}_i \in [-1, 1], i = 1, \ldots, n$.

Usage

```
makeExponentialFunction(dimensions)
```

Arguments

`dimensions`	`[integer(1)]` Size of corresponding parameter space.

Value

An object of class `SingleObjectiveFunction`, representing the Exponential Function.

`[smoof_single_objective_function]`

References

S. Rahnamyan, H. R. Tizhoosh, N. M. M. Salama, Opposition-Based Differential Evolution (ODE) with Variable Jumping Rate, IEEE Sympousim Foundations Com- putation Intelligence, Honolulu, HI, pp. 81-88, 2007.

makeFreudensteinRothFunction *Freudenstein Roth Function*

Description

This test function is based on the formula

$$f(\mathbf{x}) = (\mathbf{x}_1 - 13 + ((5 - \mathbf{x}_2)\mathbf{x}_2 - 2)\mathbf{x}_2)^2 + (\mathbf{x}_1 - 29 + ((\mathbf{x}_2 + 1)\mathbf{x}_2 - 14)\mathbf{x}_2)^2$$

subject to $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makeFreudensteinRothFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Freundstein-Roth Function.

[smoof_single_objective_function]

References

S. S. Rao, Engineering Optimization: Theory and Practice, John Wiley & Sons, 2009.

makeFunctionsByName *Generate smoof function by passing a character vector of generator names.*

Description

This function is especially useful in combination with `filterFunctionsByTags` to generate a test set of functions with certain properties, e.~g., multimodality.

Usage

```
makeFunctionsByName(fun.names, ...)
```

Arguments

`fun.names`	[character] Non empty character vector of generator function names.
`...`	[any] Further arguments passed to generator.

Value

[smoof_function] Smoof function generated based on the specified names and arguments.

See Also

filterFunctionsByTags

Examples

```
# generate a testset of multimodal 2D functions
## Not run:
test.set = makeFunctionsByName(filterFunctionsByTags("multimodal"), dimensions = 2L, m = 5L)

## End(Not run)
```

---

makeGeneralizedDropWaveFunction

Generalized Drop-Wave Function

Description

Multi-modal single-objective function following the formula:

$$\mathbf{x} = -\frac{1 + \cos(\sqrt{\sum_{i=1}^{n} \mathbf{x}_i^2})}{2 + 0.5 \sum_{i=1}^{n} \mathbf{x}_i^2}$$

with $\mathbf{x}_i \in [-5.12, 5.12], i = 1, \ldots, n$.

Usage

```
makeGeneralizedDropWaveFunction(dimensions = 2L)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Generalized Drop-Wave Function.

[smoof_single_objective_function]

makeGiuntaFunction *Giunta Function*

Description

Multi-modal test function based on the definition

$$f(\mathbf{x}) = 0.6 + \sum_{i=1}^{n} \left[\sin(\frac{16}{15}\mathbf{x}_i - 1) + \sin^2(\frac{16}{15}\mathbf{x}_i - 1) + \frac{1}{50}\sin(4(\frac{16}{15}\mathbf{x}_i - 1)) \right]$$

with box-constraints $\mathbf{x}_i \in [-1, 1]$ for $i = 1, \ldots, n$.

Usage

```
makeGiuntaFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Giunta Function.

[smoof_single_objective_function]

References

S. K. Mishra, Global Optimization By Differential Evolution and Particle Swarm Methods: Evaluation On Some Benchmark Functions, Munich Research Papers in Economics.

makeGoldsteinPriceFunction *Goldstein-Price Function*

Description

Two-dimensional test function for global optimization. The implementation follows the formula:

$f(\mathbf{x}) = \big(1 + (\mathbf{x}_1 + \mathbf{x}_2 + 1)^2 \cdot (19 - 14\mathbf{x}_1 + 3\mathbf{x}_1^2 - 14\mathbf{x}_2 + 6\mathbf{x}_1\mathbf{x}_2 + 3\mathbf{x}_2^2)\big) \quad \cdot\big(30 + (2\mathbf{x}_1 - 3\mathbf{x}_2)^2 \cdot (18 - 32\mathbf{x}_1 + 12\mathbf{x}$

with $\mathbf{x}_i \in [-2, 2], i = 1, 2$.

Usage

```
makeGoldsteinPriceFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Goldstein-Price Function.

[smoof_single_objective_function]

References

Goldstein, A. A. and Price, I. F.: On descent from local minima. Math. Comput., Vol. 25, No. 115, 1971.

`makeGOMOPFunction` *GOMOP function generator.*

Description

Construct a multi-objective function by putting together multiple single-objective smoof functions.

Usage

```
makeGOMOPFunction(dimensions = 2L, funs = list())
```

Arguments

`dimensions`	[`integer(1)`] Size of corresponding parameter space.
`funs`	[`list`] List of single-objective smoof functions.

Details

The decision space of the resulting function is restricted to $[0, 1]^d$. Each parameter x is stretched for each objective function. I.e., if $f_1, \ldots, f_n$ are the single objective smoof functions with box constraints $[l_i, u_i], i = 1, \ldots, n$, then

$$f(x) = (f_1(l_1 + x * (u_1 - l_1)), \ldots, f_1(l_1 + x * (u_1 - l_1)))$$

for $x \in [0, 1]^d$ where the additions and multiplication are performed component-wise.

Value

[`smoof_multi_objective_function`] Returns an instance of the GOMOP function as a `smoof_multi_objective_functio` object.

makeGriewankFunction *Griewank Function*

Description

Highly multi-modal function with a lot of regularly distributed local minima. The corresponding formula is:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \frac{\mathbf{x}_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{\mathbf{x}_i}{\sqrt{i}}\right) + 1$$

subject to $\mathbf{x}_i \in [-100, 100], i = 1, \ldots, n$.

Usage

```
makeGriewankFunction(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Griewank Function.

[smoof_single_objective_function]

References

A. O. Griewank, Generalized Descent for Global Optimization, Journal of Optimization Theory and Applications, vol. 34, no. 1, pp. 11-39, 1981.

makeHansenFunction *Hansen Function*

Description

Test function with multiple global optima based on the definition

$$f(\mathbf{x}) = \sum_{i=1}^{4} (i+1)\cos(i\mathbf{x}_1 + i - 1) \sum_{j=1}^{4} (j+1)\cos((j+2)\mathbf{x}_2 + j + 1)$$

subject to $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makeHansenFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Hansen Function.

[smoof_single_objective_function]

References

C. Fraley, Software Performances on Nonlinear lems, Technical Report no. STAN-CS-89-1244, Computer Science, Stanford University, 1989.

`makeHartmannFunction` *Hartmann Function*

Description

Uni-modal single-objective test function with six local minima. The implementation is based on the mathematical formulation

$$f(x) = -\sum_{i=1}^{4} \alpha_i \, exp\left(-\sum_{j=1}^{6} A_{ij}(x_j - P_{ij})^2\right)$$

, where

$$\alpha = (1.0, 1.2, 3.0, 3.2)^T, A = \begin{pmatrix} 10 & 3 & 17 & 3.50 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}, P = 10^{-4} \cdot \begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 \\ 2329 & 4135 & 8307 & 3736 & 1004 \\ 2348 & 1451 & 3522 & 2883 & 3047 \\ 4047 & 8828 & 8732 & 5743 & 1091 \end{pmatrix}$$

The function is restricted to six dimensions with $\mathbf{x}_i \in [0, 1], i = 1, \ldots, 6$. The function is not normalized in contrast to some benchmark applications in the literature.

Usage

```
makeHartmannFunction(dimensions)
```

Arguments

`dimensions`	`[integer(1)]` Size of corresponding parameter space.

Value

An object of class `SingleObjectiveFunction`, representing the Hartmann Function.

[smoof_single_objective_function]

References

Picheny, V., Wagner, T., & Ginsbourger, D. (2012). A benchmark of kriging-based infill criteria for noisy optimization.

---

makeHimmelblauFunction

Himmelblau Function

Description

Two-dimensional test function based on the function definition

$$f(\mathbf{x}) = (\mathbf{x}_1^2 + \mathbf{x}_2 - 11)^2 + (\mathbf{x}_1 + \mathbf{x}_2^2 - 7)^2$$

with box-constraints $\mathbf{x}_i \in [-5, 5], i = 1, 2$.

Usage

```
makeHimmelblauFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Himmelblau Function.

[smoof_single_objective_function]

References

D. M. Himmelblau, Applied Nonlinear Programming, McGraw-Hill, 1972.

---

makeHolderTableN1Function

Holder Table function N. 1

Description

This multi-modal function is defined as

$$f(\mathbf{x}) = -\left|\cos(\mathbf{x}_1)\cos(\mathbf{x}_2)\exp(|1 - \sqrt{\mathbf{x}_1 + \mathbf{x}_2}/\pi|)\right|$$

with box-constraints $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makeHolderTableN1Function()
```

Value

An object of class `SingleObjectiveFunction`, representing the Holder Table function N. 1 Function.

[smoof_single_objective_function]

References

S. K. Mishra, Global Optimization By Differential Evolution and Particle Swarm Methods: Evaluation On Some Benchmark Functions, Munich Research Papers in Economics.

See Also

makeHolderTableN2Function

---

makeHolderTableN2Function

Holder Table function N. 2

Description

This multi-modal function is defined as

$$f(\mathbf{x}) = -\left|\sin(\mathbf{x}_1)\cos(\mathbf{x}_2)\exp(|1 - \sqrt{\mathbf{x}_1 + \mathbf{x}_2}/\pi|)\right|$$

with box-constraints $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makeHolderTableN2Function()
```

Value

An object of class SingleObjectiveFunction, representing the Holder Table function N. 2 Function.

[smoof_single_objective_function]

References

S. K. Mishra, Global Optimization By Differential Evolution and Particle Swarm Methods: Evaluation On Some Benchmark Functions, Munich Research Papers in Economics.

See Also

makeHolderTableN1Function

makeHosakiFunction *Hosaki Function*

Description

Two-dimensional test function f with

$$f(\mathbf{x}) = (1 - 8\mathbf{x}_1 + 7\mathbf{x}_1^2 - 7/3\mathbf{x}_1^3 + 1/4\mathbf{x}_1^4)\mathbf{x}_2^2 e^{-\mathbf{x}_2}$$

subject to $0 \leq \mathbf{x}_1 \leq 5$ and $0 \leq \mathbf{x}_2 \leq 6$.

Usage

```
makeHosakiFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Hosaki Function.

[smoof_single_objective_function]

References

G. A. Bekey, M. T. Ung, A Comparative Evaluation of Two Global Search Algorithms, IEEE Transaction on Systems, Man and Cybernetics, vol. 4, no. 1, pp. 112- 116, 1974.

makeHyperEllipsoidFunction
Hyper-Ellipsoid function

Description

Uni-modal, convex test function similar to the Sphere function (see `makeSphereFunction`). Calculated via the formula:

$$f(\mathbf{x}) = \sum_{i=1}^{n} i \cdot \mathbf{x}_i.$$

Usage

```
makeHyperEllipsoidFunction(dimensions)
```

Arguments

`dimensions`	[integer(1)] Size of corresponding parameter space.

Value

An object of class `SingleObjectiveFunction`, representing the Hyper-Ellipsoid Function.

[smoof_single_objective_function]

`makeInvertedVincentFunction`
Inverted Vincent Function

Description

Single-objective test function based on the formula

$$f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \sin(10 \log(\mathbf{x}_i))$$

subject to $\mathbf{x}_i \in [0.25, 10]$ for $i = 1, \ldots, n$.

Usage

```
makeInvertedVincentFunction(dimensions)
```

Arguments

`dimensions`	[integer(1)] Size of corresponding parameter space.

Value

An object of class `SingleObjectiveFunction`, representing the Inverted Vincent Function.

[smoof_single_objective_function]

References

Xiadong Li, Andries Engelbrecht, and Michael G. Epitropakis. Benchmark functions for CEC2013 special session and competition on niching methods for multi-modal function optimization. Technical report, RMIT University, Evolutionary Computation and Machine Learning Group, Australia, 2013.

---

makeJennrichSampsonFunction *Jennrich-Sampson function.*

Description

Two-dimensional test function based on the formula

$$f(\mathbf{x}) = \sum_{i=1}^{10} \left[2 + 2i - (e^{ix_1} + e^{ix_2})\right]^2$$

with $\mathbf{x}_1, \mathbf{x}_2 \in [-1, 1]$.

Usage

```
makeJennrichSampsonFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Jennrich-Sampson Function.
[`smoof_single_objective_function`]

References

See https://al-roomi.org/benchmarks/unconstrained/2-dimensions/134-jennrich-sampson-s-function.

---

makeJudgeFunction *Judge function.*

Description

Two-dimensional test function based on the formula

$$f(\mathbf{x}) = \sum_{i=1}^{20} \left[(x_1 + B_i x_2 + C_i x_2^2) - A_i\right]^2$$

with $\mathbf{x}_1, \mathbf{x}_2 \in [-10, 10]$. For details on A, B, C see the referenced website.

Usage

```
makeJudgeFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Judge Function.
[`smoof_single_objective_function`]

References

See https://al-roomi.org/benchmarks/unconstrained/2-dimensions/133-judge-s-function.

`makeKeaneFunction` *Keane Function*

Description

Two-dimensional test function based on the definition

$$f(\mathbf{x}) = \frac{\sin^2(\mathbf{x}_1 - \mathbf{x}_2)\sin^2(\mathbf{x}_1 + \mathbf{x}_2)}{\sqrt{\mathbf{x}_1^2 + \mathbf{x}_2^2}}.$$

The domain of definition is bounded by the box constraints $\mathbf{x}_i \in [0, 10], i = 1, 2$.

Usage

```
makeKeaneFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Keane Function.

[smoof_single_objective_function]

`makeKearfottFunction` *Kearfott function.*

Description

Two-dimensional test function based on the formula

$$f(\mathbf{x}) = (x_1^2 + x_2^2 - 2)^2 + (x_1^2 - x_2^2 - 1)^2$$

with $\mathbf{x}_1, \mathbf{x}_2 \in [-3, 4]$.

Usage

```
makeKearfottFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Kearfott Function.

[smoof_single_objective_function]

References

See https://al-roomi.org/benchmarks/unconstrained/2-dimensions/59-kearfott-s-function.

makeKursaweFunction *Kursawe Function*

Description

Builds and returns the bi-objective Kursawe test problem.

The Kursawe test problem is defined as follows:

Minimize $f_1(\mathbf{x}) = \sum_{i=1}^{n-1}(-10 \cdot \exp(-0.2 \cdot \sqrt{x_i^2 + x_{i+1}^2}))$,

Minimize $f_2(\mathbf{x}) = \sum_{i=1}^{n}(|x_i|^{0.8} + 5 \cdot \sin^3(x_i))$,

with $-5 \leq x_i \leq 5$, for $i = 1, 2, \ldots, n$,.

Usage

```
makeKursaweFunction(dimensions)
```

Arguments

`dimensions`	`[integer(1)]` Number of decision variables.

Value

[`smoof_multi_objective_function`] Returns an instance of the Kursawe function as a `smoof_multi_objective_functio` object.

References

F. Kursawe. A Variant of Evolution Strategies for Vector Optimization. Proceedings of the International Conference on Parallel Problem Solving from Nature, pp. 193-197, Springer, 1990.

makeLeonFunction *Leon Function*

Description

The function is based on the definition

$$f(\mathbf{x}) = 100(\mathbf{x}_2 - \mathbf{x}_1^2)^2 + (1 - \mathbf{x}_1)^2$$

. Box-constraints: $\mathbf{x}_i \in [-1.2, 1.2]$ for $i = 1, 2$.

Usage

```
makeLeonFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Leon Function.

[smoof_single_objective_function]

References

A. Lavi, T. P. Vogel (eds), Recent Advances in Optimization Techniques, John Wliley & Sons, 1966.

---

makeMatyasFunction *Matyas Function*

Description

Two-dimensional, uni-modal test function

$$f(\mathbf{x}) = 0.26(\mathbf{x}_1^2 + \mathbf{x}_2^2) - 0.48\mathbf{x}_1\mathbf{x}_2$$

subject to $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makeMatyasFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Matyas Function.

[smoof_single_objective_function]

References

A.-R. Hedar, Global Optimization Test Problems.

---

makeMcCormickFunction *McCormick Function*

Description

Two-dimensional, multi-modal test function. The definition is given by

$$f(\mathbf{x}) = \sin(\mathbf{x}_1 + \mathbf{x}_2) + (\mathbf{x}_1 - \mathbf{x}_2)^2 - 1.5\mathbf{x}_1 + 2.5\mathbf{x}_2 + 1$$

subject to $\mathbf{x}_1 \in [-1.5, 4], \mathbf{x}_2 \in [-3, 3]$.

Usage

```
makeMcCormickFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the McCormick Function.

[smoof_single_objective_function]

References

F. A. Lootsma (ed.), Numerical Methods for Non-Linear Optimization, Academic Press, 1972.

---

makeMichalewiczFunction

Michalewicz Function

Description

Highly multi-modal single-objective test function with $n!$ local minima with the formula:

$$f(\mathbf{x}) = -\sum_{i=1}^{n} \sin(\mathbf{x}_i) \cdot \left(\sin\left(\frac{i \cdot \mathbf{x}_i}{\pi} \right) \right)^{2m}.$$

The recommended value $m = 10$, which is used as a default in the implementation.

Usage

```
makeMichalewiczFunction(dimensions, m = 10)
```

Arguments

`dimensions`	[integer(1)] Size of corresponding parameter space.
`m`	[integer(1)] "Steepness" parameter.

Value

An object of class `SingleObjectiveFunction`, representing the Michalewicz Function.

[smoof_single_objective_function]

Note

The location of the global optimum s varying based on both the dimension and m parameter and is thus not provided in the implementation.

References

Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Berlin, Heidelberg, New York: Springer-Verlag, 1992.

makeMMF10Function *MMF10 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF10Function()
```

Value

[smoof_multi_objective_function] Returns an instance of the MMF10 function as a smoof_multi_objective_functio object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

makeMMF11Function *MMF11 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF11Function(np = 2L)
```

Arguments

np	[integer(1)] Number of global Pareto sets. In the CEC2019 competition, the organizers used np = 2L.

Value

[smoof_multi_objective_function] Returns an instance of the MMF11 function as a smoof_multi_objective_functio object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

`makeMMF12Function` *MMF12 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF12Function(np = 2L, q = 4L)
```

Arguments

`np`	[`integer(1)`] Number of global Pareto sets. In the CEC2019 competition, the organizers used `np = 2L`.
`q`	[`integer(1)`] Number of discontinuous pieces in each Pareto front. In the CEC2019 competition, the organizers used `q = 4L`.

Value

[`smoof_multi_objective_function`] Returns an instance of the MMF12 function as a `smoof_multi_objective_functio` object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

makeMMF13Function *MMF13 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF13Function(np = 2L)
```

Arguments

np	[integer(1)] Number of global Pareto sets. In the CEC2019 competition, the organizers used np = 2L.

Value

[smoof_multi_objective_function] Returns an instance of the MMF13 function as a smoof_multi_objective_functio
object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

makeMMF14aFunction *MMF14a Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF14aFunction(dimensions, n.objectives, np = 2L)
```

Arguments

dimensions	[integer(1)] Number of decision variables.
n.objectives	[integer(1)] Number of objectives.
np	[integer(1)] Number of global Pareto sets. In the CEC2019 competition, the organizers used np = 2L.

Value

[smoof_multi_objective_function] Returns an instance of the MMF14a function as a smoof_multi_objective_functio object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

---

makeMMF14Function *MMF14 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF14Function(dimensions, n.objectives, np = 2L)
```

Arguments

dimensions	[integer(1)] Number of decision variables.
n.objectives	[integer(1)] Number of objectives.
np	[integer(1)] Number of global Pareto sets. In the CEC2019 competition, the organizers used np = 2L.

Value

[smoof_multi_objective_function] Returns an instance of the MMF14 function as a smoof_multi_objective_functio object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

`makeMMF15aFunction` *MMF15a Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF15aFunction(dimensions, n.objectives, np = 2L)
```

Arguments

`dimensions`	[`integer(1)`] Number of decision variables.
`n.objectives`	[`integer(1)`] Number of objectives.
`np`	[`integer(1)`] Number of global Pareto sets. In the CEC2019 competition, the organizers used `np = 2L`.

Value

[`smoof_multi_objective_function`] Returns an instance of the MMF15a function as a `smoof_multi_objective_functio` object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

makeMMF15Function *MMF15 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF15Function(dimensions, n.objectives, np = 2L)
```

Arguments

dimensions	[integer(1)] Number of decision variables.
n.objectives	[integer(1)] Number of objectives.
np	[integer(1)] Number of global Pareto sets. In the CEC2019 competition, the organizers used np = 2L.

Value

[smoof_multi_objective_function] Returns an instance of the MMF15 function as a smoof_multi_objective_functio object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

makeMMF1eFunction *MMF1e Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF1eFunction(a = exp(1L))
```

Arguments

a	[double(1)] Parametrizable factor. In the CEC2019 competition, the organizers used a = exp(1L).

Value

[smoof_multi_objective_function] Returns an instance of the MMF1e function as a smoof_multi_objective_functio object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multimodal multiobjective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

---

makeMMF1Function *MMF1 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF1Function()
```

Value

[smoof_multi_objective_function] Returns an instance of the MMF1 function as a smoof_multi_objective_function object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multimodal multiobjective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

makeMMF1zFunction *MMF1z Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF1zFunction(k = 3)
```

Arguments

k	[double(1)] Parametrizable factor. In the CEC2019 competition, the organizers used k = 3.

Value

[smoof_multi_objective_function] Returns an instance of the MMF1z function as a smoof_multi_objective_functio object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

makeMMF2Function *MMF2 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF2Function()
```

Value

[smoof_multi_objective_function] Returns an instance of the MMF2 function as a smoof_multi_objective_function object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

`makeMMF3Function` *MMF3 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF3Function()
```

Value

[`smoof_multi_objective_function`] Returns an instance of the MMF3 function as a `smoof_multi_objective_function` object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

`makeMMF4Function` *MMF4 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF4Function()
```

Value

[`smoof_multi_objective_function`] Returns an instance of the MMF4 function as a `smoof_multi_objective_function` object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

`makeMMF5Function` *MMF5 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF5Function()
```

Value

[`smoof_multi_objective_function`] Returns an instance of the MMF5 function as a `smoof_multi_objective_function` object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

`makeMMF6Function` *MMF6 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF6Function()
```

Value

[`smoof_multi_objective_function`] Returns an instance of the MMF6 function as a `smoof_multi_objective_function` object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

`makeMMF7Function` *MMF7 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF7Function()
```

Value

[`smoof_multi_objective_function`] Returns an instance of the MMF7 function as a `smoof_multi_objective_function` object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

`makeMMF8Function` *MMF8 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF8Function()
```

Value

[`smoof_multi_objective_function`] Returns an instance of the MMF8 function as a `smoof_multi_objective_function` object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

`makeMMF9Function` *MMF9 Function*

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeMMF9Function(np = 2L)
```

Arguments

`np`	[`integer(1)`] Number of global Pareto sets. In the CEC2019 competition, the organizers used `np = 2L`.

Value

[`smoof_multi_objective_function`] Returns an instance of the MMF9 function as a `smoof_multi_objective_function` object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

`makeMNKFunction` *Generators for (r)MNK-landscapes*

Description

Generators for multi-objective NK-landscapes, i.e., with at least two objectives. Function `makeMNKLandscape(M, N, K)` create NK-landscapes with `M` (≥ 2) objectives, input dimension `N`, and epistatic links specified via parameter `K`. `K` can be a single integer value, a vector of `M` integers or a list of length-`N` integer vectors (see parameter description for details) which allow for maximum flexibility. It is also possible to compose a MNK-landscape by passing a list of single-objective NK-landscapes via argument `funs`.

Usage

```
makeMNKFunction(M, N, K, funs = NULL)

makeRMNKFunction(M, N, K, rho = 0)
```

Arguments

M	[integer(1)] Number of objectives (at least two).
N	[integer(1)] Length of the bit-string (decision space dimension).
K	[integer] Epistatic links. Possible values are a a) single integer which is used for all bits and positions, b) an integer vector of N integers that are used across the objectives, c) a list of M length-N integers to define specific epistatic links for every bit position of every objective.
funs	[list | NULL] Allows for an alternative way to build a MNK-landscape by passing a list of at least two single-objective NK-landscapes. In this case all other parameters M, N, K and rho are ignored. Note that the passed functions must be compatible, i.e., a) the input dimension N needs to match and b) all pased functions need to be multi-objective. Default is NULL.
rho	[numeric(1)] Correlation between objectives (value between -1 and 1).

Details

Function makeRMNKLandscape(M, N, K, rho) generates a MNK-landscape with correlated objective function values. The correlation can be adjusted by setting the rho parameter to a value between minus one and one.

Value

[smoof_multi_objective_function]

References

H. E. Aguirre and K. Tanaka, Insights on properties of multiobjective MNK-landscapes, Proceedings of the 2004 Congress on Evolutionary Computation, Portland, OR, USA, 2004, pp. 196-203 Vol.1, doi: 10.1109/CEC.2004.1330857.

See Also

makeNKFunction

Other nk_landscapes: exportNKFunction(), makeNKFunction()

Examples

```
# generate homogeneous uncorrelated bi-objective MNK-landscape with each
# three epistatic links
M = 2L
N = 20L
K = 3L
fn = makeMNKFunction(M, N, K)

# generate MNK-landscape where the first function has 3 epistatic links
# per bit while the second function has 2
fn = makeMNKFunction(M, N, K = c(3L, 2L))

# sample the number of epistatic links individually from {1, ..., 5} for
# every bit position and every objective
K = lapply(seq_len(M), function(m) sample(1:(N-1), size = N, replace = TRUE))
fn = makeMNKFunction(M, N, K = K)

#' # generate strongly positively correlated objectives
fn = makeRMNKFunction(M, N, K, rho = 0.9)

# alternative constructor: generate two single-objective NK-landscapes
# and combine into bi-objective problem
soofn1 = makeNKFunction(N, K = 2L) # homegeneous in K
K = sample(2:3, size = N, replace = TRUE)
soofn2 = makeNKFunction(N, K = K) # heterogeneous in K
moofn = makeMNKFunction(funs = list(soofn1, soofn2))
getNumberOfObjectives(moofn)
```

---

makeModifiedRastriginFunction

Rastrigin Function

Description

A modified version of the Rastrigin function following the formula:

$$f(\mathbf{x}) = \sum_{i=1}^{n} 10\left(1 + \cos(2\pi k_i \mathbf{x}_i)\right) + 2k_i \mathbf{x}_i^2.$$

The box-constraints are given by $\mathbf{x}_i \in [0, 1]$ for $i = 1, \ldots, n$ and k is a numerical vector. Deb et al. (see references) use, e.g., $k = (2, 2, 3, 4)$ for $n = 4$. See the reference for details.

Usage

```
makeModifiedRastriginFunction(dimensions, k = rep(1, dimensions))
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.
k	[numeric] Vector of numerical values of length dimensions. Default is rep(1, dimensions)

Value

An object of class SingleObjectiveFunction, representing the Rastrigin Function.

[smoof_single_objective_function]

References

Kalyanmoy Deb and Amit Saha. Multi-modal optimization using a bi- objective evolutionary algorithm. Evolutionary Computation, 20(1):27-62, 2012.

---

makeMOP1Function *MOP1 function generator.*

Description

MOP1 function from Van Valedhuizen's test suite.

Usage

```
makeMOP1Function()
```

Value

[smoof_multi_objective_function] Returns an instance of the MOP1 function as a smoof_multi_objective_function object.

References

J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in Proc. 1st Int. Conf. Genetic Algorithms and Their Applications, J. J. Grenfenstett, Ed., 1985, pp. 93-100.

makeMOP2Function *MOP2 function generator.*

Description

MOP2 function from Van Valedhuizen's test suite due to Fonseca and Fleming.

Usage

```
makeMOP2Function(dimensions = 2L)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

[smoof_multi_objective_function] Returns an instance of the MOP2 function as a smoof_multi_objective_function object.

References

C. M. Fonseca and P. J. Fleming, "Multi-objective genetic algorithms made easy: Selection, sharing and mating restriction," Genetic Algorithms in Engineering Systems: Innovations and Applications, pp. 45-52, Sep. 1995. IEE.

makeMOP3Function *MOP3 function generator.*

Description

MOP3 function from Van Valedhuizen's test suite.

Usage

```
makeMOP3Function(dimensions = 2L)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

[smoof_multi_objective_function] Returns an instance of the MOP3 function as a smoof_multi_objective_function object.

References

C. Poloni, G. Mosetti, and S. Contessi, "Multi-objective optimization by GAs: Application to system and component design," in Proc. Comput. Methods in Applied Sciences'96: Invited Lectures and Special Technological Sessions of the 3rd ECCOMAS Comput. Fluid Dynamics Conf. and the 2nd ECCOMAS Conf. Numerical Methods in Engineering, Sep. 1996, pp. 258-264

`makeMOP4Function` *MOP4 function generator.*

Description

MOP4 function from Van Valedhuizen's test suite based on Kursawe.

Usage

```
makeMOP4Function()
```

Value

[smoof_multi_objective_function] Returns an instance of the MOP4 function as a smoof_multi_objective_function object.

References

F. Kursawe, "A variant of evolution strategies for vector optimization," in Lecture Notes in Computer Science, H.-P. Schwefel and R. Maenner, Eds. Berlin, Germany: Springer-Verlag, 1991, vol. 496, Proc. Parallel Problem Solving From Nature. 1st Workshop, PPSN I, pp. 193-197.

`makeMOP5Function` *MOP5 function generator.*

Description

MOP5 function from Van Valedhuizen's test suite.

Usage

```
makeMOP5Function()
```

Value

[smoof_multi_objective_function] Returns an instance of the MOP5 function as a smoof_multi_objective_function object.

Note

Original box constraints where $[-3, 3]$.

References

R. Viennet, C. Fonteix, and I. Marc, "Multi criteria optimization using a genetic algorithm for determining a Pareto set," Int. J. Syst. Sci., vol. 27, no. 2, pp. 255-260, 1996

---

makeMOP6Function *MOP6 function generator.*

Description

MOP6 function from Van Valedhuizen's test suite.

Usage

```
makeMOP6Function()
```

Value

[smoof_multi_objective_function] Returns an instance of the MOP6 function as a smoof_multi_objective_function object.

---

makeMOP7Function *MOP7 function generator.*

Description

MOP7 function from Van Valedhuizen's test suite.

Usage

```
makeMOP7Function()
```

Value

[smoof_multi_objective_function] Returns an instance of the MOP7 function as a smoof_multi_objective_function object.

References

R. Viennet, C. Fonteix, and I. Marc, "Multi-criteria optimization using a genetic algorithm for determining a Pareto set," Int. J. Syst. Sci., vol. 27, no. 2, pp. 255-260, 1996

`makeMPM2Function` *Generator for function with multiple peaks following the multiple peaks model 2.*

Description

Generator for function with multiple peaks following the multiple peaks model 2.

Usage

```
makeMPM2Function(
  n.peaks,
  dimensions,
  topology,
  seed,
  rotated = TRUE,
  peak.shape = "ellipse",
  evaluation.env = "R"
)
```

Arguments

`n.peaks`	[`integer(1)`] Desired number of peaks, i. e., number of (local) optima.
`dimensions`	[`integer(1)`] Size of corresponding parameter space.
`topology`	[`character(1)`] Type of topology. Possible values are "random" and "funnel".
`seed`	[`integer(1)`] Seed for the random numbers generator.
`rotated`	[`logical(1)`] Should the peak shapes be rotated? This parameter is only relevant in case of elliptically shaped peaks.
`peak.shape`	[`character(1)`] Shape of peak(s). Possible values are "ellipse" and "sphere".
`evaluation.env`	[`character(1)`] Evaluation environment after the function was created. Possible (case-insensitive) values are "R" (default) and "Python". The original generation of the problem is always done in the original Python environment. However, evaluation in R is faster, especially if multiple MPM2 functions are used in a multi-objective setting.

Value

[`smoof_single_objective_function`] An object of class `SingleObjectiveFunction`, representing the Multiple peaks model 2 Function.

Author(s)

R interface by Jakob Bossek. Original python code provided by the Simon Wessing.

References

See the technical report of multiple peaks model 2 for an in-depth description of the underlying algorithm.

Examples

```
## Not run:
fn = makeMPM2Function(n.peaks = 10L, dimensions = 2L,
  topology = "funnel", seed = 123, rotated = TRUE, peak.shape = "ellipse")
if (require(plot3D)) {
  plot3D(fn)
}

## End(Not run)
## Not run:
fn = makeMPM2Function(n.peaks = 5L, dimensions = 2L,
  topology = "random", seed = 134, rotated = FALSE)
plot(fn, render.levels = TRUE)

## End(Not run)
```

---

makeMultiObjectiveFunction

Generator for multi-objective target functions.

Description

Generator for multi-objective target functions.

Usage

```
makeMultiObjectiveFunction(
  name = NULL,
  id = NULL,
  description = NULL,
  fn,
  has.simple.signature = TRUE,
  par.set,
  n.objectives = NULL,
  noisy = FALSE,
  fn.mean = NULL,
  minimize = NULL,
  vectorized = FALSE,
```

```
  constraint.fn = NULL,
  ref.point = NULL
)
```

Arguments

name	[character(1)] Function name. Used for the title of plots for example.
id	[character(1) | NULL] Optional short function identifier. If provided, this should be a short name without whitespaces and now special characters beside the underscore. Default is NULL, which means no ID at all.
description	[character(1) | NULL] Optional function description.
fn	[function] Objective function.
has.simple.signature	[logical(1)] Set this to TRUE if the objective function expects a vector as input and FALSE if it expects a named list of values. The latter is needed if the function depends on mixed parameters. Default is TRUE.
par.set	[ParamSet] Parameter set describing different aspects of the objective function parameters, i.~e., names, lower and/or upper bounds, types and so on. See makeParamSet for further information.
n.objectives	[integer(1)] Number of objectives of the multi-objective function.
noisy	[logical(1)] Is the function noisy? Defaults to FALSE.
fn.mean	[function] Optional true mean function in case of a noisy objective function. This functions should have the same mean as fn.
minimize	[logical] Logical vector of length n.objectives indicating if the corresponding objectives shall be minimized or maximized. Default is the vector with all components set to TRUE.
vectorized	[logical(1)] Can the objective function handle "vector" input, i.~e., does it accept matrix of parameters? Default is FALSE.
constraint.fn	[function | NULL] Function which returns a logical vector indicating whether certain conditions are met or not. Default is NULL, which means, that there are no constraints beside possible box constraints defined via the par.set argument.
ref.point	[numeric] Optional reference point in the objective space, e.g., for hyper-volume computation.

Value

[function] Target function with additional stuff attached as attributes.

Examples

```
fn = makeMultiObjectiveFunction(
  name = "My test function",
  fn = function(x) c(sum(x^2), exp(x)),
  n.objectives = 2L,
  par.set = makeNumericParamSet("x", len = 1L, lower = -5L, upper = 5L)
)
print(fn)
```

---

makeNKFunction *Generator for NK-landscapes*

Description

Generate a single-objective NK-landscape. NK-landscapes are combinatorial problems with input space $\{0,1\}^N$ (in their basic definition). The value of each bit position $i \in \{1,\ldots,N\}$ depends on K other bits, the so-called *(epistatic) links / interactions*.

Usage

```
makeNKFunction(N, K)
```

Arguments

N	[integer(1)] Length of the bit-string (decision space dimension).
K	[integer] Integer vector of the number of epistatic interactions. If a single value is passed a homogeneous NK-landscape is generated, i.e. $k_i = k \forall i = 1,\ldots,N$.

Value

[smoof_single_objective_function] NK-landscape function.

References

Kauffman SA, Weinberger ED. The NK model of rugged fitness landscapes and its application to maturation of the immune response. Journal of Theoretical Biology 1989 Nov 21;141(2):211-45. doi: 10.1016/s0022-5193(89)80019-0.

See Also

Other nk_landscapes: exportNKFunction(), makeMNKFunction()

Examples

```
# generate homogeneous NK-landscape with each K=3 epistatic links
N = 20
fn = makeNKFunction(N, 3)

# evaluate function on some random bitstrings
bitstrings = matrix(sample(c(0, 1), size = 10 * N, replace = TRUE), ncol = N)
apply(bitstrings, 1, fn)

# generate heterogeneous NK-landscape where K is sampled from {2,3,4}
# uniformly at random
fn = makeNKFunction(N, K = sample(2:4, size = N, replace = TRUE))
```

---

makeObjectiveFunction *Internal generator for function of smoof type.*

Description

Internal generator for function of smoof type.

Usage

```
makeObjectiveFunction(
  name = NULL,
  id = NULL,
  description = NULL,
  fn,
  has.simple.signature = TRUE,
  par.set,
  n.objectives = NULL,
  noisy = FALSE,
  fn.mean = NULL,
  minimize = NULL,
  vectorized = FALSE,
  constraint.fn = NULL
)
```

Arguments

name	[character(1)] Optional function name used e.g. in plots.
id	[character(1)] Optional identifier for the function
description	[character(1) | NULL] Optional function description.

fn [function]
Target function.

has.simple.signature
[logical(1)]
Set this to TRUE if the target function expects a vector as input and FALSE if it expects a named list of values. The latter is needed if the function depends on mixed parameters. Default is TRUE.

par.set [ParamSet]
Parameter set describing different aspects of the target function parameters, i. e., names, lower and/or upper bounds, types and so on. See makeParamSet for further information.

n.objectives [integer(1)]
Number of objectives of the multi-objective function.

noisy [logical(1)]
Is the function noisy? Defaults to FALSE.

fn.mean [function]
Optional true mean function in case of a noisy objective function. This functions should have the same mean as fn.

minimize [logical]
Logical vector of length n.objectives indicating which objectives shall be minimized/maximized. The default is TRUE n.objectives times.

vectorized [logical(1)]
Can the handle "vector" input, i. e., does it accept matrix of parameters? Default is FALSE.

constraint.fn [function | NULL]
Function which returns a logical vector indicating which indicates whether certain conditions are met or not. Default is NULL, which means, that there are no constraints (beside possible) box constraints defined via the par.set argument.

Value

[function] Target function with additional stuff attached as attributes.

---

makeOmniTestFunction *MMF13 Function*

Description

Test problem from the set of "multimodal multiobjective functions" as for instance used in the CEC2019 competition.

Usage

```
makeOmniTestFunction()
```

Value

[smoof_multi_objective_function] Returns an instance of the Omni function as a smoof_multi_objective_function object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multimodal multiobjective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

---

makePeriodicFunction *Periodic Function*

Description

Single-objective two-dimensional test function. The formula is given as

$$f(\mathbf{x}) = 1 + \sin^2(\mathbf{x}_1) + \sin^2(\mathbf{x}_2) - 0.1e^{-(\mathbf{x}_1^2 + \mathbf{x}_2^2)}$$

subject to the constraints $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makePeriodicFunction()
```

Value

An object of class SingleObjectiveFunction, representing the Periodic Function.

[smoof_single_objective_function]

References

M. M. Ali, C. Khompatraporn, Z. B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems, Journal of Global Optimization, vol. 31, pp. 635-672, 2005.

makePowellSumFunction *Powell-Sum Function*

Description

The formula that underlies the implementation is given by

$$f(\mathbf{x}) = \sum_{i=1}^{n} |\mathbf{x}_i|^{i+1}$$

with $\mathbf{x}_i \in [-1, 1], i = 1, \ldots, n$.

Usage

```
makePowellSumFunction(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Powell-Sum Function.

[smoof_single_objective_function]

References

S. Rahnamyan, H. R. Tizhoosh, N. M. M. Salama, A Novel Population Initialization Method for Accelerating Evolutionary Algorithms, Computers and Mathematics with Applications, vol. 53, no. 10, pp. 1605-1614, 2007.

makePriceN1Function *Price Function N. 1*

Description

Second function by Price. The implementation is based on the definition

$$f(\mathbf{x}) = (|\mathbf{x}_1| - 5)^2 + (|\mathbf{x}_2 - 5)^2$$

subject to $\mathbf{x}_i \in [-500, 500], i = 1, 2$.

Usage

```
makePriceN1Function()
```

Value

An object of class `SingleObjectiveFunction`, representing the Price N. 1 Function.

[smoof_single_objective_function]

References

W. L. Price, A Controlled Random Search Procedure for Global Optimization, Computer journal, vol. 20, no. 4, pp. 367-370, 1977.

See Also

`makePriceN2Function`, `makePriceN4Function`

---

makePriceN2Function *Price Function N. 2*

Description

Second function by Price. The implementation is based on the defintion

$$f(\mathbf{x}) = 1 + \sin^2(\mathbf{x}_1) + \sin^2(\mathbf{x}_2) - 0.1 \exp(-\mathbf{x}^2 - \mathbf{x}_2^2)$$

subject to $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makePriceN2Function()
```

Value

An object of class `SingleObjectiveFunction`, representing the Price N. 2 Function.

[smoof_single_objective_function]

References

W. L. Price, A Controlled Random Search Procedure for Global Optimisation, Computer journal, vol. 20, no. 4, pp. 367-370, 1977.

See Also

`makePriceN1Function`, `makePriceN4Function`

makePriceN4Function *Price Function N. 4*

Description

Fourth function by Price. The implementation is based on the definition

$$f(\mathbf{x}) = (2\mathbf{x}_1^3\mathbf{x}_2 - \mathbf{x}_2^3)^2 + (6\mathbf{x}_1 - \mathbf{x}_2^2 + \mathbf{x}_2)^2$$

subject to $\mathbf{x}_i \in [-500, 500]$.

Usage

```
makePriceN4Function()
```

Value

An object of class `SingleObjectiveFunction`, representing the Price N. 4 Function.

[smoof_single_objective_function]

References

W. L. Price, A Controlled Random Search Procedure for Global Optimisation, Computer journal, vol. 20, no. 4, pp. 367-370, 1977.

See Also

`makePriceN1Function`, `makePriceN2Function`

makeRastriginFunction *Rastrigin Function*

Description

One of the most popular single-objective test functions consists of many local optima and is thus highly multi-modal with a global structure. The implementation follows the formula

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(\mathbf{x}_i^2 - 10\cos(2\pi\mathbf{x}_i)\right).$$

The box-constraints are given by $\mathbf{x}_i \in [-5.12, 5.12]$ for $i = 1, \ldots, n$.

Usage

```
makeRastriginFunction(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Rastrigin Function.

[smoof_single_objective_function]

References

L. A. Rastrigin. Extremal control systems. Theoretical Foundations of Engineering Cybernetics Series. Nauka, Moscow, 1974.

---

makeRosenbrockFunction

Rosenbrock Function

Description

Also known as the "De Jong's function 2" or the "(Rosenbrock) banana/valley function" due to its shape. The global optimum is located within a large flat valley and thus it is hard for optimization algorithms to find it. The following formula underlies the implementation:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} 100 \cdot (\mathbf{x}_{i+1} - \mathbf{x}_i^2)^2 + (1 - \mathbf{x}_i)^2.$$

The domain is given by the constraints $\mathbf{x}_i \in [-30, 30], i = 1, \ldots, n$.

Usage

```
makeRosenbrockFunction(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Rosenbrock Function.

[smoof_single_objective_function]

References

H. H. Rosenbrock, An Automatic Method for Finding the Greatest or least Value of a Function, Computer Journal, vol. 3, no. 3, pp. 175-184, 1960.

`makeSchafferN2Function`

Modified Schaffer Function N. 2

Description

Second function by Schaffer. The definition is given by the formula

$$f(\mathbf{x}) = 0.5 + \frac{\sin^2(\mathbf{x}_1^2 - \mathbf{x}_2^2) - 0.5}{(1 + 0.001(\mathbf{x}_1^2 + \mathbf{x}_2^2))^2}$$

subject to $\mathbf{x}_i \in [-100, 100], i = 1, 2$.

Usage

```
makeSchafferN2Function()
```

Value

An object of class `SingleObjectiveFunction`, representing the Modified Schaffer N. 2 Function.

[smoof_single_objective_function]

References

S. K. Mishra, Some New Test Functions For Global Optimization And Performance of Repulsive Particle Swarm Method.

`makeSchafferN4Function`

Schaffer Function N. 4

Description

Second function by Schaffer. The definition is given by the formula

$$f(\mathbf{x}) = 0.5 + \frac{\cos^2(sin(|\mathbf{x}_1^2 - \mathbf{x}_2^2|)) - 0.5}{(1 + 0.001(\mathbf{x}_1^2 + \mathbf{x}_2^2))^2}$$

subject to $\mathbf{x}_i \in [-100, 100], i = 1, 2$.

Usage

```
makeSchafferN4Function()
```

Value

An object of class `SingleObjectiveFunction`, representing the Schaffer N. 4 Function.

[smoof_single_objective_function]

References

S. K. Mishra, Some New Test Functions For Global Optimization And Performance of Repulsive Particle Swarm Method.

---

makeSchwefelFunction *Schwefel function*

Description

Highly multi-modal test function. The crucial thing about this function is, that the global optimum is far away from the next best local optimum. The function is computed via:

$$f(\mathbf{x}) = \sum_{i=1}^{n} -\mathbf{x}_i \sin\left(\sqrt{(}|\mathbf{x}_i|)\right)$$

with $\mathbf{x}_i \in [-500, 500], i = 1, \ldots, n$.

Usage

```
makeSchwefelFunction(dimensions)
```

Arguments

`dimensions`	[integer(1)] Size of corresponding parameter space.

Value

An object of class `SingleObjectiveFunction`, representing the Schwefel Function.

[smoof_single_objective_function]

References

Schwefel, H.-P.: Numerical optimization of computer models. Chichester: Wiley & Sons, 1981.

makeShekelFunction *Shekel functions*

Description

Single-objective test function based on the formula

$$f(\mathbf{x}) = -\sum_{i=1}^{m} \left(\sum_{j=1}^{4} (x_j - C_{ji})^2 + \beta_i \right)^{-1}$$

. Here, $m \in \{5, 7, 10\}$ defines the number of local optima, C is a $4x10$ matrix and $\beta = \frac{1}{10}(1, 1, 2, 2, 4, 4, 6, 3, 7, 5, 5)$ is a vector. See https://www.sfu.ca/~ssurjano/shekel.html for a definition of C.

Usage

```
makeShekelFunction(m)
```

Arguments

m	[numeric(1)] Integer parameter (defines the number of local optima). Possible values are 5, 7 or 10.

Value

An object of class SingleObjectiveFunction, representing the Shekel Functions.

[smoof_single_objective_function]

makeShubertFunction *Shubert Function*

Description

The definition of this two-dimensional function is given by

$$f(\mathbf{x}) = \prod_{i=1}^{2} \left(\sum_{j=1}^{5} \cos((j+1)\mathbf{x}_i + j \right)$$

subject to $\mathbf{x}_i \in [-10, 10], i = 1, 2$.

Usage

```
makeShubertFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Shubert Function.

[smoof_single_objective_function]

References

J. P. Hennart (ed.), Numerical Analysis, Proc. 3rd AS Workshop, Lecture Notes in Mathematics, vol. 90, Springer, 1982.

`makeSingleObjectiveFunction`

Generator for single-objective target functions.

Description

Generator for single-objective target functions.

Usage

```
makeSingleObjectiveFunction(
  name = NULL,
  id = NULL,
  description = NULL,
  fn,
  has.simple.signature = TRUE,
  vectorized = FALSE,
  par.set,
  noisy = FALSE,
  fn.mean = NULL,
  minimize = TRUE,
  constraint.fn = NULL,
  tags = character(0),
  global.opt.params = NULL,
  global.opt.value = NULL,
  local.opt.params = NULL,
  local.opt.values = NULL
)
```

Arguments

`name`	[`character(1)`] Function name. Used for the title of plots for example.
`id`	[`character(1)` | `NULL`] Optional short function identifier. If provided, this should be a short name without whitespaces and now special characters beside the underscore. Default is `NULL`, which means no ID at all.

description [character(1) | NULL]
Optional function description.

fn [function]
Objective function.

has.simple.signature
[logical(1)]
Set this to TRUE if the objective function expects a vector as input and FALSE if it expects a named list of values. The latter is needed if the function depends on mixed parameters. Default is TRUE.

vectorized [logical(1)]
Can the objective function handle "vector" input, i.~e., does it accept matrix of parameters? Default is FALSE.

par.set [ParamSet]
Parameter set describing different aspects of the objective function parameters, i.~e., names, lower and/or upper bounds, types and so on. See makeParamSet for further information.

noisy [logical(1)]
Is the function noisy? Defaults to FALSE.

fn.mean [function]
Optional true mean function in case of a noisy objective function. This functions should have the same mean as fn.

minimize [logical(1)]
Set this to TRUE if the function should be minimized and to FALSE otherwise. The default is TRUE.

constraint.fn [function | NULL]
Function which returns a logical vector indicating whether certain conditions are met or not. Default is NULL, which means, that there are no constraints beside possible box constraints defined via the par.set argument.

tags [character]
Optional character vector of tags or keywords which characterize the function, e.~g. "unimodal", "separable". See getAvailableTags for a character vector of allowed tags.

global.opt.params
[list | numeric | data.frame | matrix | NULL]
Default is NULL which means unknown. Passing a numeric vector will be the most frequent case (numeric only functions). In this case there is only a single global optimum. If there are multiple global optima, passing a numeric matrix is the best choice. Passing a list or a data.frame is necessary if your function is mixed, e.g., it expects both numeric and discrete parameters. Internally, however, each representation is casted to a data.frame for reasons of consistency.

global.opt.value
[numeric(1) | NULL]
Global optimum value if known. Default is NULL, which means unknown. If only the global.opt.params are passed, the value is computed automatically.

local.opt.params
: [list | numeric | data.frame | matrix | NULL]
Default is NULL, which means the function has no local optima or they are unknown. For details see the description of global.opt.params.

local.opt.values
: [numeric | NULL]
Value(s) of local optima. Default is NULL, which means unknown. If only the local.opt.params are passed, the values are computed automatically.

Value

[function] Objective function with additional stuff attached as attributes.

Examples

```
library(ggplot2)

fn = makeSingleObjectiveFunction(
  name = "Sphere Function",
  fn = function(x) sum(x^2),
  par.set = makeNumericParamSet("x", len = 1L, lower = -5L, upper = 5L),
  global.opt.params = list(x = 0)
)
print(fn)
print(autoplot(fn))

fn.num2 = makeSingleObjectiveFunction(
  name = "Numeric 2D",
  fn = function(x) sum(x^2),
  par.set = makeParamSet(
    makeNumericParam("x1", lower = -5, upper = 5),
    makeNumericParam("x2", lower = -10, upper = 20)
  )
)
print(fn.num2)
print(autoplot(fn.num2))

fn.mixed = makeSingleObjectiveFunction(
  name = "Mixed 2D",
  fn = function(x) x$num1^2 + as.integer(as.character(x$disc1) == "a"),
  has.simple.signature = FALSE,
  par.set = makeParamSet(
    makeNumericParam("num1", lower = -5, upper = 5),
    makeDiscreteParam("disc1", values = c("a", "b"))
  ),
  global.opt.params = list(num1 = 0, disc1 = "b")
)
print(fn.mixed)
print(autoplot(fn.mixed))
```

makeSixHumpCamelFunction *Three-Hump Camel Function*

Description

Two dimensional single-objective test function with six local minima oh which two are global. The surface is similar to the back of a camel. That is why it is called Camel function. The implementation is based on the formula:

$$f(\mathbf{x}) = \left(4 - 2.1\mathbf{x}_1^2 + \mathbf{x}_1^{0.75}\right)\mathbf{x}_1^2 + \mathbf{x}_1\mathbf{x}_2 + \left(-4 + 4\mathbf{x}_2^2\right)\mathbf{x}_2^2$$

with box constraints $\mathbf{x}_1 \in [-3, 3]$ and $\mathbf{x}_2 \in [-2, 2]$.

Usage

```
makeSixHumpCamelFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Three-Hump Camel Function.

[smoof_single_objective_function]

References

Dixon, L. C. W. and Szego, G. P.: The optimization problem: An introduction. In: Towards Global Optimization II, New York: North Holland, 1978.

makeSphereFunction *Sphere Function*

Description

Also known as the the "De Jong function 1". Convex, continous function calculated via the formula

$$f(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{x}_i^2$$

with box-constraints $\mathbf{x}_i \in [-5.12, 5.12], i = 1, \ldots, n$.

Usage

```
makeSphereFunction(dimensions)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.

Value

An object of class SingleObjectiveFunction, representing the Sphere Function.

[smoof_single_objective_function]

References

M. A. Schumer, K. Steiglitz, Adaptive Step Size Random Search, IEEE Transactions on Automatic Control. vol. 13, no. 3, pp. 270-276, 1968.

---

makeStyblinkskiTangFunction

Styblinkski-Tang function

Description

This function is based on the definition

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{2}(\mathbf{x}_i^4 - 16\mathbf{x}_i^2 + 5\mathbf{x}_i)$$

with box-constraints given by $\mathbf{x}_i \in [-5, 5], i = 1, 2$.

Usage

```
makeStyblinkskiTangFunction()
```

Value

An object of class SingleObjectiveFunction, representing the Styblinkski-Tang Function.

[smoof_single_objective_function]

References

Z. K. Silagadze, Finding Two-Dimesnional Peaks, Physics of Particles and Nuclei Letters, vol. 4, no. 1, pp. 73-80, 2007.

`makeSumOfDifferentSquaresFunction`
Sum of Different Squares Function

Description

Simple uni-modal test function similar to the Sphere and Hyper-Ellipsoidal functions. Formula:

$$f(\mathbf{x}) = \sum_{i=1}^{n} |\mathbf{x}_i|^{i+1}.$$

Usage

```
makeSumOfDifferentSquaresFunction(dimensions)
```

Arguments

`dimensions`	`[integer(1)]` Size of corresponding parameter space.

Value

[smoof_single_objective_function]

`makeSwiler2014Function`
Swiler2014 function.

Description

Mixed parameter space with one discrete parameter $x_1 \in \{1, 2, 3, 4, 5\}$ and two numerical parameters $x_1, x_2 \in [0, 1]$. The function is defined as follows:

$f(\mathbf{x}) = \sin(2\pi x_3 - \pi) + 7\sin^2(2\pi x_2 - \pi)\, if\, x_1 = 1 \sin(2\pi x_3 - \pi) + 7\sin^2(2\pi x_2 - \pi) + 12\sin(2\pi x_3 - \pi)\, if\, x_1 = 2 \sin(2\pi x_3 -$

Usage

```
makeSwiler2014Function()
```

Value

[smoof_single_objective_function]

makeSYMPARTrotatedFunction

SYM-PART Rotated Function

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeSYMPARTrotatedFunction(w = pi/4, a = 1, b = 10, c = 8)
```

Arguments

w	[double(1)] Parametrizable factor. In the CEC2019 competition, the organizers used w = pi / 4.
a	[double(1)] Parametrizable factor. In the CEC2019 competition, the organizers used a = 1.
b	[double(1)] Parametrizable factor. In the CEC2019 competition, the organizers used b = 10.
c	[double(1)] Parametrizable factor. In the CEC2019 competition, the organizers used c = 8.

Value

[smoof_multi_objective_function] Returns an instance of the SYM-PART Rotated function as a smoof_multi_objective_function object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

makeSYMPARTsimpleFunction

SYM-PART Simple Function

Description

Test problem from the set of "multi-modal multi-objective functions" as for instance used in the CEC2019 competition.

Usage

```
makeSYMPARTsimpleFunction(a = 1, b = 10, c = 8)
```

Arguments

a	[double(1)] Parametrizable factor. In the CEC2019 competition, the organizers used a = 1.
b	[double(1)] Parametrizable factor. In the CEC2019 competition, the organizers used b = 10.
c	[double(1)] Parametrizable factor. In the CEC2019 competition, the organizers used c = 8.

Value

[smoof_multi_objective_function] Returns an instance of the SYM-PART Simple function as a smoof_multi_objective_function object.

References

Caitong Yue, Boyang Qu, Kunjie Yu, Jing Liang, and Xiaodong Li, "A novel scalable test problem suite for multi-modal multi-objective optimization," in Swarm and Evolutionary Computation, Volume 48, August 2019, pp. 62–71, Elsevier.

---

makeThreeHumpCamelFunction

Three-Hump Camel Function

Description

This two-dimensional function is based on the defintion

$$f(\mathbf{x}) = 2\mathbf{x}_1^2 - 1.05\mathbf{x}_1^4 + \frac{\mathbf{x}_1^6}{6} + \mathbf{x}_1\mathbf{x}_2 + \mathbf{x}_2^2$$

subject to $-5 \leq \mathbf{x}_i \leq 5$.

Usage

```
makeThreeHumpCamelFunction()
```

Value

An object of class SingleObjectiveFunction, representing the Three-Hump Camel Function.

[smoof_single_objective_function]

References

F. H. Branin Jr., Widely Convergent Method of Finding Multiple Solutions of Simultaneous Nonlinear Equations, IBM Journal of Research and Development, vol. 16, no. 5, pp. 504-522, 1972.

makeTrecanniFunction *Trecanni Function*

Description

The Trecanni function belongs to the uni-modal test functions. It is based on the formula

$$f((x)) = (x)_1^4 - 4(x)_1^3 + 4(x)_1 + (x)_2^2.$$

The box-constraints $\mathbf{x}_i \in [-5, 5], i = 1, 2$ define the domain of definition.

Usage

```
makeTrecanniFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Trecanni Function.

[smoof_single_objective_function]

References

L. C. W. Dixon, G. P. Szego (eds.), Towards Global Optimization 2, Elsevier, 1978.

makeUFFunction *Generator for the functions UF1, ..., UF10 of the CEC 2009.*

Description

Generator for the functions UF1, ..., UF10 of the CEC 2009.

Usage

```
makeUFFunction(dimensions, id)
```

Arguments

dimensions	[integer(1)] Size of corresponding parameter space.
id	[integer(1)] Instance identifier. Integer value between 1 and 10.

Value

[smoof_single_objective_function]

Note

The implementation is based on the original CPP implementation by Qingfu Zhang, Aimin Zhou, Shizheng Zhaoy, Ponnuthurai Nagaratnam Suganthany, Wudong Liu and Santosh Tiwar.

Author(s)

Jakob Bossek <j.bossek@gmail.com>

---

makeViennetFunction *Viennet function generator*

Description

The Viennet test problem VNT is designed for three objectives only. It has a discrete set of Pareto fronts. It is defined by the following formulae.

$$f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}, f_3(\mathbf{x})$$

with

$$f_1(\mathbf{x}) = 0.5(\mathbf{x}_1^2 + \mathbf{x}_2^2) + \sin(\mathbf{x}_1^2 + \mathbf{x}_2^2)$$

$$f_2(\mathbf{x}) = \frac{(3\mathbf{x}_1 + 2\mathbf{x}_2 + 4)^2}{8} + \frac{(\mathbf{x}_1 - \mathbf{x}_2 + 1)^2}{27} + 15$$

$$f_3(\mathbf{x}) = \frac{1}{\mathbf{x}_1^2 + \mathbf{x}_2^2 + 1} - 1.1\exp(-(\mathbf{x}_1^1 + \mathbf{x}_2^2))$$

with box constraints $-3 \leq \mathbf{x}_1, \mathbf{x}_2 \leq 3$.

Usage

```
makeViennetFunction()
```

Value

[smoof_multi_objective_function] Returns an instance of the Viennet function as a smoof_multi_objective_function object.

References

Viennet, R. (1996). Multi-criteria optimization using a genetic algorithm for determining the Pareto set. International Journal of Systems Science 27 (2), 255-260.

makeWFG1Function *WFG1 Function*

Description

First test problem from the "Walking Fish Group" problem generator toolkit.

Usage

```
makeWFG1Function(n.objectives, k, l)
```

Arguments

n.objectives	[integer(1)] Number of objectives.
k	[integer(1)] Number of position-related parameters. These will automatically be the first k elements from the input vector. This value has to be a multiple of n.objectives - 1.
l	[integer(1)] Number of distance-related parameters. These will automatically be the last l elements from the input vector.

Details

Huband et al. recommend a value of k = 4L position-related parameters for bi-objective problems and k = 2L * (n.objectives - 1L) for many-objective problems. Furthermore the authors recommend a value of l = 20 distance-related parameters. Therefore, if k and/or l are not explicitly defined by the user, their values will be set to the recommended values per default.

Value

[smoof_multi_objective_function] Returns an instance of the WGF1 function as a smoof_multi_objective_function object.

References

S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit," in IEEE Transactions on Evolutionary Computation, Volume 10, No 5, October 2006, pp. 477-506. IEEE.

makeWFG2Function *WFG2 Function*

Description

Second test problem from the "Walking Fish Group" problem generator toolkit.

Usage

```
makeWFG2Function(n.objectives, k, l)
```

Arguments

n.objectives	[integer(1)] Number of objectives.
k	[integer(1)] Number of position-related parameters. These will automatically be the first k elements from the input vector. This value has to be a multiple of n.objectives - 1.
l	[integer(1)] Number of distance-related parameters. These will automatically be the last l elements from the input vector. This value has to be a multiple of 2.

Details

Huband et al. recommend a value of k = 4L position-related parameters for bi-objective problems and k = 2L * (n.objectives - 1L) for many-objective problems. Furthermore the authors recommend a value of l = 20 distance-related parameters. Therefore, if k and/or l are not explicitly defined by the user, their values will be set to the recommended values per default.

Value

[smoof_multi_objective_function] Returns an instance of the WFG2 function as a smoof_multi_objective_function object.

References

S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit," in IEEE Transactions on Evolutionary Computation, Volume 10, No 5, October 2006, pp. 477-506. IEEE.

makeWFG3Function *WFG3 Function*

Description

Third test problem from the "Walking Fish Group" problem generator toolkit.

Usage

```
makeWFG3Function(n.objectives, k, l)
```

Arguments

n.objectives	[integer(1)] Number of objectives.
k	[integer(1)] Number of position-related parameters. These will automatically be the first k elements from the input vector. This value has to be a multiple of n.objectives - 1.
l	[integer(1)] Number of distance-related parameters. These will automatically be the last l elements from the input vector. This value has to be a multiple of 2.

Details

Huband et al. recommend a value of k = 4L position-related parameters for bi-objective problems and k = 2L * (n.objectives - 1L) for many-objective problems. Furthermore the authors recommend a value of l = 20 distance-related parameters. Therefore, if k and/or l are not explicitly defined by the user, their values will be set to the recommended values per default.

Value

[smoof_multi_objective_function] Returns an instance of the WFG3 function as a smoof_multi_objective_function object.

References

S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit," in IEEE Transactions on Evolutionary Computation, Volume 10, No 5, October 2006, pp. 477-506. IEEE.

makeWFG4Function *WFG4 Function*

Description

Fourth test problem from the "Walking Fish Group" problem generator toolkit.

Usage

```
makeWFG4Function(n.objectives, k, l)
```

Arguments

n.objectives	[integer(1)] Number of objectives.
k	[integer(1)] Number of position-related parameters. These will automatically be the first k elements from the input vector. This value has to be a multiple of n.objectives - 1.
l	[integer(1)] Number of distance-related parameters. These will automatically be the last l elements from the input vector.

Details

Huband et al. recommend a value of k = 4L position-related parameters for bi-objective problems and k = 2L * (n.objectives - 1L) for many-objective problems. Furthermore the authors recommend a value of l = 20 distance-related parameters. Therefore, if k and/or l are not explicitly defined by the user, their values will be set to the recommended values per default.

Value

[smoof_multi_objective_function] Returns an instance of the WFG4 function as a smoof_multi_objective_function object.

References

S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit," in IEEE Transactions on Evolutionary Computation, Volume 10, No 5, October 2006, pp. 477-506. IEEE.

makeWFG5Function *WFG5 Function*

Description

Fifth test problem from the "Walking Fish Group" problem generator toolkit.

Usage

```
makeWFG5Function(n.objectives, k, l)
```

Arguments

n.objectives	[integer(1)] Number of objectives.
k	[integer(1)] Number of position-related parameters. These will automatically be the first k elements from the input vector. This value has to be a multiple of n.objectives - 1.
l	[integer(1)] Number of distance-related parameters. These will automatically be the last l elements from the input vector.

Details

Huband et al. recommend a value of k = 4L position-related parameters for bi-objective problems and k = 2L * (n.objectives - 1L) for many-objective problems. Furthermore the authors recommend a value of l = 20 distance-related parameters. Therefore, if k and/or l are not explicitly defined by the user, their values will be set to the recommended values per default.

Value

[smoof_multi_objective_function] Returns an instance of the WFG5 function as a smoof_multi_objective_function object.

References

S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit," in IEEE Transactions on Evolutionary Computation, Volume 10, No 5, October 2006, pp. 477-506. IEEE.

makeWFG6Function *WFG6 Function*

Description

Sixth test problem from the "Walking Fish Group" problem generator toolkit.

Usage

```
makeWFG6Function(n.objectives, k, l)
```

Arguments

n.objectives	[integer(1)] Number of objectives.
k	[integer(1)] Number of position-related parameters. These will automatically be the first k elements from the input vector. This value has to be a multiple of n.objectives - 1.
l	[integer(1)] Number of distance-related parameters. These will automatically be the last l elements from the input vector.

Details

Huband et al. recommend a value of k = 4L position-related parameters for bi-objective problems and k = 2L * (n.objectives - 1L) for many-objective problems. Furthermore the authors recommend a value of l = 20 distance-related parameters. Therefore, if k and/or l are not explicitly defined by the user, their values will be set to the recommended values per default.

Value

[smoof_multi_objective_function] Returns an instance of the WFG6 function as a smoof_multi_objective_function object.

References

S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit," in IEEE Transactions on Evolutionary Computation, Volume 10, No 5, October 2006, pp. 477-506. IEEE.

makeWFG7Function *WFG7 Function*

Description

Seventh test problem from the "Walking Fish Group" problem generator toolkit.

Usage

```
makeWFG7Function(n.objectives, k, l)
```

Arguments

n.objectives	[integer(1)] Number of objectives.
k	[integer(1)] Number of position-related parameters. These will automatically be the first k elements from the input vector. This value has to be a multiple of n.objectives - 1.
l	[integer(1)] Number of distance-related parameters. These will automatically be the last l elements from the input vector.

Details

Huband et al. recommend a value of k = 4L position-related parameters for bi-objective problems and k = 2L * (n.objectives - 1L) for many-objective problems. Furthermore the authors recommend a value of l = 20 distance-related parameters. Therefore, if k and/or l are not explicitly defined by the user, their values will be set to the recommended values per default.

Value

[smoof_multi_objective_function] Returns an instance of the WFG7 function as a smoof_multi_objective_function object.

References

S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit," in IEEE Transactions on Evolutionary Computation, Volume 10, No 5, October 2006, pp. 477-506. IEEE.

makeWFG8Function *WFG8 Function*

Description

Eighth test problem from the "Walking Fish Group" problem generator toolkit.

Usage

```
makeWFG8Function(n.objectives, k, l)
```

Arguments

n.objectives	[integer(1)] Number of objectives.
k	[integer(1)] Number of position-related parameters. These will automatically be the first k elements from the input vector. This value has to be a multiple of n.objectives - 1.
l	[integer(1)] Number of distance-related parameters. These will automatically be the last l elements from the input vector.

Details

Huband et al. recommend a value of k = 4L position-related parameters for bi-objective problems and k = 2L * (n.objectives - 1L) for many-objective problems. Furthermore the authors recommend a value of l = 20 distance-related parameters. Therefore, if k and/or l are not explicitly defined by the user, their values will be set to the recommended values per default.

Value

[smoof_multi_objective_function] Returns an instance of the WFG8 function as a smoof_multi_objective_function object.

References

S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit," in IEEE Transactions on Evolutionary Computation, Volume 10, No 5, October 2006, pp. 477-506. IEEE.

makeWFG9Function *WFG9 Function*

Description

Ninth test problem from the "Walking Fish Group" problem generator toolkit.

Usage

```
makeWFG9Function(n.objectives, k, l)
```

Arguments

n.objectives	[integer(1)] Number of objectives.
k	[integer(1)] Number of position-related parameters. These will automatically be the first k elements from the input vector. This value has to be a multiple of n.objectives - 1.
l	[integer(1)] Number of distance-related parameters. These will automatically be the last l elements from the input vector.

Details

Huband et al. recommend a value of k = 4L position-related parameters for bi-objective problems and k = 2L * (n.objectives - 1L) for many-objective problems. Furthermore the authors recommend a value of l = 20 distance-related parameters. Therefore, if k and/or l are not explicitly defined by the user, their values will be set to the recommended values per default.

Value

[smoof_multi_objective_function] Returns an instance of the WFG9 function as a smoof_multi_objective_function object.

References

S. Huband, P. Hingston, L. Barone, and L. While, "A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit," in IEEE Transactions on Evolutionary Computation, Volume 10, No 5, October 2006, pp. 477-506. IEEE.

makeZDT1Function *ZDT1 Function*

Description

Builds and returns the two-objective ZDT1 test problem. For m objective it is defined as follows:

$$f(\mathbf{x}) = (f_1(\mathbf{x}_1), f_2(\mathbf{x}))$$

with

$$f_1(\mathbf{x}_1) = \mathbf{x}_1, f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}_1), g(\mathbf{x}))$$

where

$$g(\mathbf{x}) = 1 + \frac{9}{m-1}\sum_{i=2}^{m}\mathbf{x}_i, h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$$

and $\mathbf{x}_i \in [0, 1], i = 1, \ldots, m$

Usage

```
makeZDT1Function(dimensions)
```

Arguments

dimensions	[integer(1)] Number of decision variables.

Value

[smoof_multi_objective_function] Returns an instance of the ZDT1 function as a smoof_multi_objective_function object.

References

E. Zitzler, K. Deb, and L. Thiele. Comparison of Multi-objective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8(2):173-195, 2000

`makeZDT2Function` *ZDT2 Function*

Description

Builds and returns the two-objective ZDT2 test problem. The function is non-convex and resembles the ZDT1 function. For m objective it is defined as follows

$$f(\mathbf{x}) = (f_1(\mathbf{x}_1), f_2(\mathbf{x}))$$

with

$$f_1(\mathbf{x}_1) = \mathbf{x}_1, f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}_1), g(\mathbf{x}))$$

where

$$g(\mathbf{x}) = 1 + \frac{9}{m-1}\sum_{i=2}^{m}\mathbf{x}_i, h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^2$$

and $\mathbf{x}_i \in [0, 1], i = 1, \ldots, m$

Usage

```
makeZDT2Function(dimensions)
```

Arguments

`dimensions`	`[integer(1)]` Number of decision variables.

Value

[`smoof_multi_objective_function`] Returns an instance of the ZDT2 function as a `smoof_multi_objective_function` object.

References

E. Zitzler, K. Deb, and L. Thiele. Comparison of Multi-objective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8(2):173-195, 2000

`makeZDT3Function` *ZDT3 Function*

Description

Builds and returns the two-objective ZDT3 test problem. For m objective it is defined as follows

$$f(\mathbf{x}) = (f_1(\mathbf{x}_1), f_2(\mathbf{x}))$$

with

$$f_1(\mathbf{x}_1) = \mathbf{x}_1, f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}_1), g(\mathbf{x}))$$

where

$$g(\mathbf{x}) = 1 + \frac{9}{m-1}\sum_{i=2}^{m}\mathbf{x}_i, h(f_1, g) = 1 - \sqrt{\frac{f_1(\mathbf{x})}{g(\mathbf{x})}} - \left(\frac{f_1(\mathbf{x})}{g(\mathbf{x})}\right)\sin(10\pi f_1(\mathbf{x}))$$

and $\mathbf{x}_i \in [0, 1], i = 1, \ldots, m$. This function has some discontinuities in the Pareto-optimal front introduced by the sine term in the h function (see above). The front consists of multiple convex parts.

Usage

```
makeZDT3Function(dimensions)
```

Arguments

`dimensions`	`[integer(1)]` Number of decision variables.

Value

[`smoof_multi_objective_function`] Returns an instance of the ZDT3 function as a `smoof_multi_objective_function` object.

References

E. Zitzler, K. Deb, and L. Thiele. Comparison of Multi-objective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8(2):173-195, 2000

`makeZDT4Function` *ZDT4 Function*

Description

Builds and returns the two-objective ZDT4 test problem. For m objective it is defined as follows

$$f(\mathbf{x}) = (f_1(\mathbf{x}_1), f_2(\mathbf{x}))$$

with

$$f_1(\mathbf{x}_1) = \mathbf{x}_1, f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}_1), g(\mathbf{x}))$$

where

$$g(\mathbf{x}) = 1 + 10(m-1) + \sum_{i=2}^{m}(\mathbf{x}_i^2 - 10\cos(4\pi\mathbf{x}_i)), h(f_1, g) = 1 - \sqrt{\frac{f_1(\mathbf{x})}{g(\mathbf{x})}}$$

and $\mathbf{x}_i \in [0, 1], i = 1, \ldots, m$. This function has many Pareto-optimal fronts and is thus suited to test the algorithms ability to tackle multi-modal problems.

Usage

```
makeZDT4Function(dimensions)
```

Arguments

`dimensions`	`[integer(1)]` Number of decision variables.

Value

[`smoof_multi_objective_function`] Returns an instance of the ZDT4 function as a `smoof_multi_objective_function` object.

References

E. Zitzler, K. Deb, and L. Thiele. Comparison of Multi-objective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8(2):173-195, 2000

makeZDT6Function *ZDT6 Function*

Description

Builds and returns the two-objective ZDT6 test problem. For m objective it is defined as follows

$$f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))$$

with

$$f_1(\mathbf{x}) = 1 - \exp(-4\mathbf{x}_1)\sin^6(6\pi\mathbf{x}_1), f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}_1), g(\mathbf{x}))$$

where

$$g(\mathbf{x}) = 1 + 9\left(\frac{\sum_{i=2}^{m}\mathbf{x}_i}{m-1}\right)^{0.25}, h(f_1, g) = 1 - \left(\frac{f_1(\mathbf{x})}{g(\mathbf{x})}\right)^2$$

and $\mathbf{x}_i \in [0,1], i = 1,\ldots,m$. This function introduced two difficulties (see reference): 1. the density of solutions decreases with the closeness to the Pareto-optimal front and 2. the Pareto-optimal solutions are non-uniformly distributed along the front.

Usage

```
makeZDT6Function(dimensions)
```

Arguments

dimensions	[integer(1)] Number of decision variables.

Value

[smoof_multi_objective_function] Returns an instance of the ZDT6 function as a smoof_multi_objective_function object.

References

E. Zitzler, K. Deb, and L. Thiele. Comparison of Multi-objective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8(2):173-195, 2000

makeZettlFunction *Zettl Function*

Description

The uni-modal Zettl Function is based on the definition

$$f(\mathbf{x}) = (\mathbf{x}_1^2 + \mathbf{x}_2^2 - 2\mathbf{x}_1)^2 + 0.25\mathbf{x}_1$$

with box-constraints $\mathbf{x}_i \in [-5, 10], i = 1, 2$.

Usage

```
makeZettlFunction()
```

Value

An object of class `SingleObjectiveFunction`, representing the Zettl Function.

[smoof_single_objective_function]

References

H. P. Schwefel, Evolution and Optimum Seeking, John Wiley Sons, 1995.

mnof *Helper function to create a numeric multi-objective optimization test function.*

Description

This is a simplifying wrapper around `makeMultiObjectiveFunction`. It can be used if the function to generate is purely numeric to save some lines of code.

Usage

```
mnof(
  name = NULL,
  id = NULL,
  par.len = NULL,
  par.id = "x",
  par.lower = NULL,
  par.upper = NULL,
  n.objectives,
  description = NULL,
  fn,
  vectorized = FALSE,
```

```
  noisy = FALSE,
  fn.mean = NULL,
  minimize = rep(TRUE, n.objectives),
  constraint.fn = NULL,
  ref.point = NULL
)
```

Arguments

name	[character(1)] Function name. Used for the title of plots for example.
id	[character(1) | NULL] Optional short function identifier. If provided, this should be a short name without whitespaces and now special characters beside the underscore. Default is NULL, which means no ID at all.
par.len	[integer(1)] Length of parameter vector.
par.id	[character(1)] Optional name of parameter vector. Default is "x".
par.lower	[numeric] Vector of lower bounds. A single value of length 1 is automatically replicated to n.pars. Default is -Inf.
par.upper	[numeric] Vector of upper bounds. A singe value of length 1 is automatically replicated to n.pars. Default is Inf.
n.objectives	[integer(1)] Number of objectives of the multi-objective function.
description	[character(1) | NULL] Optional function description.
fn	[function] Objective function.
vectorized	[logical(1)] Can the objective function handle "vector" input, i.~e., does it accept matrix of parameters? Default is FALSE.
noisy	[logical(1)] Is the function noisy? Defaults to FALSE.
fn.mean	[function] Optional true mean function in case of a noisy objective function. This functions should have the same mean as fn.
minimize	[logical] Logical vector of length n.objectives indicating if the corresponding objectives shall be minimized or maximized. Default is the vector with all components set to TRUE.

constraint.fn [function | NULL]
Function which returns a logical vector indicating whether certain conditions are met or not. Default is NULL, which means, that there are no constraints beside possible box constraints defined via the par.set argument.

ref.point [numeric]
Optional reference point in the objective space, e.g., for hyper-volume computation.

Value

[smoof_multi_objective_function] Returns a numeric multi-objective optimization test function created using the provided parameters.

Examples

```
# first we generate the 10d sphere function the long way
fn = makeMultiObjectiveFunction(
  name = "Testfun",
  fn = function(x) c(sum(x^2), exp(sum(x^2))),
  par.set = makeNumericParamSet(
    len = 10L, id = "a",
    lower = rep(-1.5, 10L), upper = rep(1.5, 10L)
  ),
  n.objectives = 2L
)

# ... and now the short way
fn = mnof(
 name = "Testfun",
 fn = function(x) c(sum(x^2), exp(sum(x^2))),
 par.len = 10L, par.id = "a", par.lower = -1.5, par.upper = 1.5,
 n.objectives = 2L
)
```

---

plot.smoof_function *Generate ggplot2 object.*

Description

This function generates a ggplot2 object for visualization.

Usage

```
## S3 method for class 'smoof_function'
plot(x, ...)
```

Arguments

x	[smoof_function] Function.
...	[any] Further parameters passed to the corresponding plot functions.

Value

Nothing

---

plot1DNumeric *Plot an one-dimensional function.*

Description

Plot an one-dimensional function.

Usage

```
plot1DNumeric(
  x,
  show.optimum = FALSE,
  main = getName(x),
  n.samples = 500L,
  ...
)
```

Arguments

x	[smoof_function] Function.
show.optimum	[logical(1)] If the function has a known global optimum, should its location be plotted by a point or multiple points in case of multiple global optima? Default is FALSE.
main	[character(1L)] Plot title. Default is the name of the smoof function.
n.samples	[integer(1)] Number of locations to be sampled. Default is 500.
...	[any] Further paramerters passed to plot function.

Value

Nothing

plot2DNumeric *Plot a two-dimensional numeric function.*

Description

Either a contour-plot or a level-plot.

Usage

```
plot2DNumeric(
  x,
  show.optimum = FALSE,
  main = getName(x),
  render.levels = FALSE,
  render.contours = TRUE,
  n.samples = 100L,
  ...
)
```

Arguments

x	[smoof_function] Function.
show.optimum	[logical(1)] If the function has a known global optimum, should its location be plotted by a point or multiple points in case of multiple global optima? Default is FALSE.
main	[character(1L)] Plot title. Default is the name of the smoof function.
render.levels	[logical(1)] Show a level-plot? Default is FALSE.
render.contours	[logical(1)] Render contours? Default is TRUE.
n.samples	[integer(1)] Number of locations per dimension to be sampled. Default is 100.
...	[any] Further paramerters passed to image respectively contour function.

Value

Nothing

plot3D *Surface plot of two-dimensional test function.*

Description

This function generates a surface plot of a two-dimensional smoof function.

Usage

```
plot3D(x, length.out = 100L, package = "plot3D", ...)
```

Arguments

x	[smoof_function] Two-dimensional snoof function.
length.out	[integer(1)] Determines the "smoothness" of the grid. The higher the value, the smoother the function landscape looks like. However, you should avoid setting this parameter to high, since with the contour option set to TRUE the drawing can take quite a lot of time. Default is 100.
package	[character(1)] String describing the package to use for 3D visualization. At the moment "plot3D" (package **plot3D**) and "plotly" (package **plotly**) are supported. The latter opens a highly interactive plot in a web browser and is thus suited well to explore a function by hand. Default is "plot3D".
...	[any] Further parameters passed to method used for visualization (which is determined by the package argument.

Examples

```
library(plot3D)
fn = makeRastriginFunction(dimensions = 2L)
## Not run:
# use the plot3D::persp3D method (default behaviour)
plot3D(fn)
plot3D(fn, contour = TRUE)
plot3D(fn, image = TRUE, phi = 30)

# use plotly::plot_ly for interactive plot
plot3D(fn, package = "plotly")

## End(Not run)
```

resetEvaluationCounter *Reset evaluation counter.*

Description

Reset evaluation counter.

Usage

```
resetEvaluationCounter(fn)
```

Arguments

fn	[smoof_counting_function] Wrapped smoof_function.

shouldBeMinimized *Check if function should be minimized.*

Description

Functions can have an associated global optimum. In this case one needs to know whether the optimum is a minimum or a maximum.

Usage

```
shouldBeMinimized(fn)
```

Arguments

fn	[smoof_function] Objective function.

Value

[logical] Each component indicates whether the corresponding objective should be minimized.

`smoof_function` *Smoof function*

Description

Regular R function with additional classes `smoof_function` and one of `smoof_single_objective_function` or codesmoof_multi_objective_function. Both single- and multi-objective functions share the following attributes.

name [`character(1)`] Optional function name.

id [`character(1)`] Short identifier.

description [`character(1)`] Optional function description.

has.simple.signature `TRUE` if the target function expects a vector as input and `FALSE` if it expects a named list of values.

par.set [`ParamSet`] Parameter set describing different ascpects of the target function parameters, i. e., names, lower and/or upper bounds, types and so on.

n.objectives [`integer(1)`] Number of objectives.

noisy [`logical(1)`] Boolean indicating whether the function is noisy or not.

fn.mean [`function`] Optional true mean function in case of a noisy objective function.

minimize [`logical(1)`] Logical vector of length `n.objectives` indicating which objectives shall be minimized/maximized.

vectorized [`logical(1)`] Can the handle "vector" input, i. e., does it accept matrix of parameters?

constraint.fn [`function`] Optional function which returns a logical vector with each component indicating whether the corresponding constraint is violated.

Furthermore, single-objective function may contain additional parameters with information on local and/or global optima as well as characterizing tags.

tags [`character`] Optional character vector of tags or keywords.

global.opt.params [`data.frame`] Data frame of parameter values of global optima.

global.opt.value [`numeric(1)`] Function value of global optima.

local.opt.params [`data.frame`] Data frame of parameter values of local optima.

global.opt.value [`numeric`] Function values of local optima.

Currently tagging is not possible for multi-objective functions. The only additional attribute may be a reference point:

ref.point [`numeric`] Optional reference point of length `n.objectives`.

snof *Helper function to create numeric single-objective optimization test function.*

Description

This is a simplifying wrapper around makeSingleObjectiveFunction. It can be used if the function to generte is purely numeric to save some lines of code.

Usage

```
snof(
  name = NULL,
  id = NULL,
  par.len = NULL,
  par.id = "x",
  par.lower = NULL,
  par.upper = NULL,
  description = NULL,
  fn,
  vectorized = FALSE,
  noisy = FALSE,
  fn.mean = NULL,
  minimize = TRUE,
  constraint.fn = NULL,
  tags = character(0),
  global.opt.params = NULL,
  global.opt.value = NULL,
  local.opt.params = NULL,
  local.opt.values = NULL
)
```

Arguments

name	[character(1)] Function name. Used for the title of plots for example.
id	[character(1) | NULL] Optional short function identifier. If provided, this should be a short name without whitespaces and now special characters beside the underscore. Default is NULL, which means no ID at all.
par.len	[integer(1)] Length of parameter vector.
par.id	[character(1)] Optional name of parameter vector. Default is "x".

par.lower [numeric]
Vector of lower bounds. A single value of length 1 is automatically replicated to n.pars. Default is -Inf.

par.upper [numeric]
Vector of upper bounds. A singe value of length 1 is automatically replicated to n.pars. Default is Inf.

description [character(1) | NULL]
Optional function description.

fn [function]
Objective function.

vectorized [logical(1)]
Can the objective function handle "vector" input, i.~e., does it accept matrix of parameters? Default is FALSE.

noisy [logical(1)]
Is the function noisy? Defaults to FALSE.

fn.mean [function]
Optional true mean function in case of a noisy objective function. This functions should have the same mean as fn.

minimize [logical(1)]
Set this to TRUE if the function should be minimized and to FALSE otherwise. The default is TRUE.

constraint.fn [function | NULL]
Function which returns a logical vector indicating whether certain conditions are met or not. Default is NULL, which means, that there are no constraints beside possible box constraints defined via the par.set argument.

tags [character]
Optional character vector of tags or keywords which characterize the function, e.~g. "unimodal", "separable". See getAvailableTags for a character vector of allowed tags.

global.opt.params
[list | numeric | data.frame | matrix | NULL]
Default is NULL which means unknown. Passing a numeric vector will be the most frequent case (numeric only functions). In this case there is only a single global optimum. If there are multiple global optima, passing a numeric matrix is the best choice. Passing a list or a data.frame is necessary if your function is mixed, e.g., it expects both numeric and discrete parameters. Internally, however, each representation is casted to a data.frame for reasons of consistency.

global.opt.value
[numeric(1) | NULL]
Global optimum value if known. Default is NULL, which means unknown. If only the global.opt.params are passed, the value is computed automatically.

local.opt.params
[list | numeric | data.frame | matrix | NULL]
Default is NULL, which means the function has no local optima or they are unknown. For details see the description of global.opt.params.

local.opt.values
: [numeric | NULL]
Value(s) of local optima. Default is NULL, which means unknown. If only the local.opt.params are passed, the values are computed automatically.

Examples

```
# first we generate the 10d sphere function the long way
fn = makeSingleObjectiveFunction(
  name = "Testfun",
  fn = function(x) sum(x^2),
  par.set = makeNumericParamSet(
    len = 10L, id = "a",
    lower = rep(-1.5, 10L), upper = rep(1.5, 10L)
  )
)

# ... and now the short way
fn = snof(
 name = "Testfun",
 fn = function(x) sum(x^2),
 par.len = 10L, par.id = "a", par.lower = -1.5, par.upper = 1.5
)
```

---

violatesConstraints *Checks whether constraints are violated.*

Description

Checks whether constraints are violated.

Usage

```
violatesConstraints(fn, values)
```

Arguments

fn
: [smoof_function]
Objective function.

values
: [numeric]
List of values.

Value

[logical(1)]

`visualizeParetoOptimalFront` *Pareto-optimal front visualization.*

Description

Quickly visualize the Pareto-optimal front of a bi-criteria objective function by calling the EMOA `nsga2` and extracting the approximated Pareto-optimal front.

Usage

```
visualizeParetoOptimalFront(fn, ...)
```

Arguments

`fn`	[`smoof_multi_objective_function`] Multi-objective smoof function.
`...`	[any] Arguments passed to `nsga2`.

Value

[`ggplot`] Returns a ggplot object representing the Pareto-optimal front visualization.

Examples

```
# Here we visualize the Pareto-optimal front of the bi-objective ZDT3 function
fn = makeZDT3Function(dimensions = 3L)
vis = visualizeParetoOptimalFront(fn)

# Alternatively we can pass some more algorithm parameters to the NSGA2 algorithm
vis = visualizeParetoOptimalFront(fn, popsize = 1000L)
```

Index