Package 'soobench'

July 2, 2014

Title Single Objective Optimization Benchmark Functions

Description Collection of different single objective test functions useful for benchmarks and algorithm development.

Author Olaf Mersmann <olafm@p-value.net> Bernd Bischl <bischl@statistik.tu-dortmund.de>

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

License BSD

LazyData yes

Version 1.0-73

Suggests rgl

Collate 'counting_function.R' 'f_ackley.R' 'f_bbob2009.R' 'f_branin.R' 'f_discus.R' 'f_double_sum.R' 'f_ellipsoidal.R' 'f_goldstein_price.R' 'f_griewank.R' 'f_kotancheck.R' 'f_mexican_hat.R' 'f_rastrigin.R' 'f_rosenbrock.R' 'f_sphere.R' 'f_weierstrass.R' 'generics.R' 'is_soo_function.R' 'persp3d_soo_function.R' 'plot_soo_function.R' 'print_soo_function.R' 'rotate_parameter_space.R'

Repository CRAN

Date/Publication 2012-03-05 16:44:07

NeedsCompilation yes

R topics documented:

`ackley_function` *Generator for the Ackley function.*

Description

Generator for the Ackley function.

Usage

```
ackley_function(dimensions)
```

Arguments

`dimensions`	Size of parameter space.

Value

A `soo_function`.

Examples

```
f <- ackley_function(2)
plot(f, rank=TRUE)
```

---

bbob2009_function *(Noisy) BBOB 2009 test function generator.*

Description

(Noisy) BBOB 2009 test function generator.

Usage

```
bbob2009_function(dimensions, fid, iid)

noisy_bbob2009_function(dimensions, fid, iid,
  noiseSeed = 1L)
```

Arguments

dimensions	Size of parameter space.
fid	Function ID, valid values are 1 to 24.
iid	Instance ID, defaults to 1L.
noiseSeed	Seed for the noise random number generator, defaults to 1L.

Value

A soo_function.

Note

Due to the way the instances are generated, the function value at the optimal parameter settings (as returned by global_minimum) may differ slightly from the optimal function value. These differences are in the order of 10^{-16}.

Also note that the random number generator used for the noisy test functions is shared by all instanciated test functions. This means that if you run multiple trials in parallel within the same interpreter, your results will not necessarily be repeatable.

Author(s)

R interface by Olaf Mersmann. Original C code graciously provided by the BBOB team (Anne Auger, Hans-Georg Beyer, Nikolaus Hansen, Steffen Finck, Raymond Ros, Marc Schoenauer, Darrell Whitley)

References

For a complete description of the 24 test functions and much more background information please see the BBOB homepage

branin_function *Generatore for the Branin test function.*

Description

This function is a 2D test function. The generator does not take any parameters. The only exists so that the interface is consitent with all other test functions.

Usage

```
branin_function()
```

Value

A soo_function.

Examples

```
f <- branin_function()
plot(f, rank=TRUE)
```

counting_function *Return a new function which is identical to the* soofunction *passed in except that all function evaluations are counted.*

Description

Return a new function which is identical to the soofunction passed in except that all function evaluations are counted.

Usage

```
counting_function(fun)
```

Arguments

fun	A test function (class soo_function).

See Also

number_of_evaluations, reset_evaluation_counter

Examples

```
f <- counting_function(double_sum_function(5))
number_of_evaluations(f)
y <- f(random_parameters(1, f))
number_of_evaluations(f)
reset_evaluation_counter(f)
number_of_evaluations(f)
y <- f(random_parameters(21, f))
number_of_evaluations(f)
```

---

discus_function *Discus test function generator.*

Description

The discus test function is similar to a high condition ellipsoid. It is defined as

Usage

```
discus_function(dimensions)
```

Arguments

dimensions	Size of parameter space.

Details

$$f(x) = 10^6 x_1^2 + x'x.$$

Value

A soo_function.

---

double_sum_function *Double sum test function generator.*

Description

Double sum test function generator.

Usage

```
double_sum_function(dimensions)
```

Arguments

dimensions	Size of parameter space.

Value

A soo_function.

---

ellipsoidal_function *Generator for ellipsoidal test functions.*

Description

The ellipsoidal test function is a badly conditioned variant of the sphere function.

Usage

```
ellipsoidal_function(dimensions)
```

Arguments

dimensions	Size of parameter space.

Value

A soo_function.

Examples

```
f <- ellipsoidal_function(2)
plot(f, rank=TRUE)

##'
```

---

function_id *Get a short id for the function that can be used in filenames and such.*

Description

Get a short id for the function that can be used in filenames and such.

Usage

```
function_id(fn)

## S3 method for class 'soo_function'
function_id(fn)
```

Arguments

fn	Function to name.

Value

ID of function. Guaranteed to be unique among all test functions.

---

function_name *Get a pretty function name for a benchmark function.*

Description

Get a pretty function name for a benchmark function.

Usage

```
function_name(fn)

## S3 method for class 'soo_function'
function_name(fn)
```

Arguments

fn	Function to name.

Value

Name of function.

---

global_minimum *Retrieve the global minimum of a function.*

Description

Retrieve the global minimum of a function.

Usage

```
global_minimum(fn)

## S3 method for class 'soo_function'
global_minimum(fn)
```

Arguments

fn	Function to query.

Value

List with two elements. `par` contains the location of the global minimum in the parameter space (possibly as a list if there are multiple global minima) and `value` the function value of the global minimum.

`goldstein_price_function` *Generator for the Goldstein-Price function.*

Description

Generator for the Goldstein-Price function.

Usage

```
goldstein_price_function()
```

Value

A `soo_function`.

Examples

```
f <- ackley_function(2)
plot(f, rank=TRUE)
```

`griewank_function` *Griewank test function generator.*

Description

Griewank test function generator.

Usage

```
griewank_function(dimensions)
```

Arguments

`dimensions`	Size of parameter space.

Value

A `soo_function`.

is_soo_function *Check if a function is a SOO function.*

Description

Check if a function is a SOO function.

Usage

```
is_soo_function(fn)
```

Arguments

fn	Function to check.

Value

TRUE if fn is a proper SOO function, else FALSE.

kotancheck_function *Kotancheck test function generator.*

Description

Kotancheck test function generator.

Usage

```
kotancheck_function(dimensions)
```

Arguments

dimensions	Size of parameter space.

Value

A soo_function.

lower_bounds *Retrieve the lower or upper bounds of a test function.*

Description

Retrieve the lower or upper bounds of a test function.

Usage

```
lower_bounds(fn)

upper_bounds(fn)

## S3 method for class 'soo_function'
lower_bounds(fn)

## S3 method for class 'soo_function'
upper_bounds(fn)
```

Arguments

fn	Function to query.

Value

Vector of lower or upper bounds of test function.

mexican_hat_function *Mexican hat test function generator.*

Description

The Mexican hat function is defined slightly differently by different people. The definition used here is

Usage

```
mexican_hat_function(dimensions)
```

Arguments

dimensions	Size of parameter space.

Details

$$f(x) = -(1 - x'x) * \exp(-\frac{x'x}{2})$$

Note that we have flipped the sign of the function so that it is a minimization problem like all other SOO functions.

Value

A soo_function.

---

number_of_evaluations *Return the number of times a test function has been evaluated.*

Description

The test function must be wrapped by counting_function for this to work.

Usage

```
number_of_evaluations(fun)
```

Arguments

fun	A counting_function as returned by counting_function.

Value

The current value of the evaluation counter.

---

number_of_parameters *Return the parameter space size of a function.*

Description

Return the parameter space size of a function.

Usage

```
number_of_parameters(fn)

## S3 method for class 'soo_function'
number_of_parameters(fn)
```

Arguments

fn	Function.

Value

Expected length of first argument. I.e. the size of the parameter space of the function fn.

---

plot.soo_function *Plot a test function in 2D.*

Description

Plot a test function in 2D.

Usage

```
 ## S3 method for class 'soo_function'
plot(x, lower = lower_bounds(x),
   upper = upper_bounds(x), n = 10000L,
   main = function_name(x), xlab = expression(x[1]),
   ylab = expression(x[2]), log = FALSE, rank = FALSE,
   show = c("image", "contour"), ...,
   image_args = list(useRaster = TRUE),
   contour_args = list())
```

Arguments

x	Function to plot.
lower	Lower bounds of x1 and x2.
upper	Upper bounds of x1 and x2.
n	Number of locations at which to sample the function.
xlab	Label of x (x1) axes.
ylab	Label of y (x2) axes.
main	Main title of plot.
log	If TRUE, the z axes is plotted on log scale.
rank	If TRUE, instead of the y values, their ranks are drawn.
show	A vector of parts to plot. Defaults to c("image", "contour") and can be any subset.
...	Ignored.
image_args	List of further arguments passed to image().
contour_args	List of further arguments passed to contour().

Author(s)

Olaf Mersmann <olafm@datensplitter.net>

Examples

```
par(mfrow=c(2, 2))
f <- sphere_function(2)
plot(f)
plot(f, show="contour")
plot(f, rank=TRUE)
plot(f, log=TRUE)
```

---

plot3d *Plot a test function in 3D.*

Description

Plot a test function in 3D.

Usage

```
plot3d(x, lower = lower_bounds(x),
  upper = upper_bounds(x), n = 10000L,
  main = function_name(x), xlab = expression(x[1]),
  ylab = expression(x[2]), log = FALSE, rank = FALSE,
  ...)
```

Arguments

x	Function to plot.
lower	Lower bounds of x1 and x2.
upper	Upper bounds of x1 and x2.
n	Number of locations at which to sample the function.
xlab	Label of x (x1) axes.
ylab	Label of y (x2) axes.
main	Main title of plot.
log	If TRUE, the z axes is plotted on log scale.
rank	If TRUE, instead of the y values, their ranks are drawn.
...	Passed to persp3d.default.

Author(s)

Olaf Mersmann <olafm@datensplitter.net>

print.soo_function *Print a SOO function.*

Description

Print a SOO function.

Usage

```
## S3 method for class 'soo_function'
print(x, ...)
```

Arguments

x	A soo_function object.
...	Ignored.

random_parameters *Generate random parameters for a given function. Given a test function* fn*, generate* n *random parameter settings for that function.*

Description

Generate random parameters for a given function.

Given a test function fn, generate n random parameter settings for that function.

Usage

```
random_parameters(n, fn)
```

Arguments

n	Number of parameters to generate.
fn	Test function.

Value

A matrix containing the parameter settings in the *columns* of the matrix.

Examples

```
fn <- ackley_function(10)
X <- random_parameters(100, fn)
str(X)
y <- fn(X)
```

random_rotation_matrix *Generate a random* d*-dimensional rotation matrix.*

Description

The algorithm used to randomly create the rotation matrix is due to R Salomon (see reference). No guarantee is given that the generated rotation matricies are uniformly distributed in any sense.

Usage

```
random_rotation_matrix(d)
```

Arguments

d	Dimension of desired rotation matrix.

Value

A random $d \times d$ rotation matrix.

References

Salomon R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. Biosystems. 1996;39(3):263-78.

rastrigin_function *Rastrigin test function generator.*

Description

Rastrigin test function generator.

Usage

```
rastrigin_function(dimensions)
```

Arguments

dimensions	Size of parameter space.

Value

A soo_function.

reset_evaluation_counter

Reset the evaluation counter of a test function.

Description

The test function must be wrapped by `counting_function` for this to work.

Usage

```
reset_evaluation_counter(fun)
```

Arguments

`fun`	A `counting_function` as returned by `counting_function`.

Value

The current value of the evaluation counter.

rosenbrock_function

Rosenbrock test function generator.

Description

Rosenbrock test function generator.

Usage

```
rosenbrock_function(dimensions)
```

Arguments

`dimensions`	Size of parameter space.

Value

A `soo_function`.

rotate_parameter_space *Rotate the parameter space of a SOO function.*

Description

This function is a simple parameter space transformation. Given a function $f(x)$ it retuns a new function $f_r(x) = f(Rx)$, where R is a random rotation matrix.

Usage

```
rotate_parameter_space(fn)
```

Arguments

fn	A soo_function object.

Details

If you want repeatable results, make sure you explicitly set a seed before calling rotate_parameter_space.

Value

A new soo_function object where the parameter space has been randomly rotated.

Examples

```
f <- ackley_function(2)
f_r <- rotate_parameter_space(f)
par(mfrow=c(1, 2))
plot(f)
plot(f_r)
```

soo_function *Define a new* soo_function *object.*

Description

Define a new soo_function object.

Usage

```
soo_function(name, id, fun, dimensions, lower_bounds,
  upper_bounds, best_value, best_par)
```

Arguments

name	Name of function.
id	Short id for the function. Must be unique to the function instance and should not contain any other characters than [a-z0-9] and '-'.
fun	Function definition.
dimensions	Size of parameter space.
lower_bounds	Lower bounds of the parameter space.
upper_bounds	Upper bounds of the parameter space.
best_value	Best known function value.
best_par	Parameter settings that correspond to best_value. If there are multiple global minima, this should be a list with one entry for each minimum.

Value

A soo_function object.

Examples

```
## Given the following simple benchmark function:
f_my_sphere <- function(x)
  sum((x-1)*(x-1))

## we can define a corresponding 2d soo_function:
f <- soo_function("My Sphere", "my-sphere-2d", f_my_sphere, 2,
                  c(-10, -10), c(10, 10),
                  0, c(1, 1))

## And then plot it:
plot(f)
```

---

sphere_function *Sphere test function generator.*

Description

The sphere function is arguably the simplest test function. It is defined as

Usage

```
sphere_function(dimensions)
```

Arguments

dimensions	Size of parameter space.

Details

$$f(x) = x'x.$$

Value

A soo_function.

---

weierstrass_function *Generator for the Weierstrass function.*

Description

Generator for the Weierstrass function.

Usage

```
weierstrass_function(dimensions)
```

Arguments

dimensions	Size of parameter space.

Value

A soo_function.

Examples

```
f <- weierstrass_function(2)
plot(f, rank=TRUE)
```

Index