Package ‘IOHexperimenter’

September 1, 2020

Type Package
Title Benchmarking Part of the 'IOHprofiler'
Version 0.1.4
Author Diederick Vermetten [cre, aut], Furong Ye [cre, aut], Hao Wang [aut], Carola Doerr [aut], Thomas Bäck [aut]
Maintainer Diederick Vermetten <d.l.vermetten@liacs.leidenuniv.nl>
Description The benchmarking module for the Iterative Optimization Heuristics Profiler ('IOHprofiler'). This module provides benchmarking in the 'IOHprofiler' format, which can be visualized using the 'IOHanalyzer' module.
License BSD_3_clause + file LICENSE
Encoding UTF-8
LazyData true
URL https://github.com/IOHprofiler/IOHexperimenter
BugReports https://github.com/IOHprofiler/IOHexperimenter/issues
Imports Rcpp (>= 1.0.1), assertthat, magrittr
LinkingTo Rcpp, BH
RoxygenNote 6.1.1
SystemRequirements GNU make
Date 2020-09-01
Suggests testthat
NeedsCompilation yes
Repository CRAN
Date/Publication 2020-09-01 16:00:12 UTC

R topics documented:

as.character.IOHexperimenter .................................................. 2
as.character.IOHproblem .......................................................... 3
benchmark_algorithm .............................................................. 3
as.character.IOHexperimenter

S3 generic as.character function for IOHexperimenter

Description
S3 generic as.character function for IOHexperimenter

Usage
## S3 method for class 'IOHexperimenter'
as.character(x, ...)

Arguments
x The IOHexperimenter to print

... Arguments for underlying function

Examples
as.character(IOHexperimenter())
as.character.IOHproblem

S3 generic as.character function for IOHproblem

Description

S3 generic as.character function for IOHproblem

Usage

## S3 method for class 'IOHproblem'
as.character(x, ...)

Arguments

x
The IOHproblem to print

... Arguments for underlying function

Examples

exp <- IOHexperimenter()
p <- next_problem(exp)
as.character(p)

benchmark_algorithm

Base procedure for benchmarking a custom algorithm

Description

Base procedure for benchmarking a custom algorithm

Usage

benchmark_algorithm(user_alg, suite = "PBO", functions = c(1, 2),
instances = c(1, 2), dimensions = 16, data.dir = NULL,
algorithm.info = "", algorithm.name = "", params.track = NULL,
repetitions = 5)

Arguments

user_alg Function defining the custom algorithm. Needs to accept one parameter: an IOH-
problem object, which has the following properties:
- dimension
- function_id
- instance
• suite (Currently 'BBOB' or 'PBO')

And the following functions:
• obj_func
• target_hit
• set_parameters

suite Which suite to test on
functions Which function to test on
instances Which instances to test on
dimensions Which dimensions to test on
data.dir Where the data should be stored (defaults to ".data" when not provided)
algorithm.info Additional information about the algorithm you plan on running
algorithm.name The name of the algorithm you plan on running
params.track Which parameters to track. Should be a vector of strings, containing no spaces or commas
repetitions How many independent runs of the algorithm to do for each problem instance

Examples

benchmark_algorithm(IOH_two_rate_GA, params.track = 'Mutation_rate', data.dir = './data')

---

IOHxperimenter S3 class 'IOHxperimenter'

Description

S3 class 'IOHxperimenter'

Usage

IOHxperimenter(suite = 'PBO', dims = NULL, functions = NULL,
 instances = NULL, algorithm.info = '', algorithm.name = '',
 data.dir = NULL, param.track = NULL)

Arguments

suite Which suite to use. Available: 'PBO', 'BBOB'
dims Numerical Which dimensions to use
functions Numerical Which functions from the selected suite to use
instances Numerical Which problem instances to use
algorithm.info Additional information about the algorithm you plan on running
**algorithm.name**  The name of the algorithm you plan on running

**data.dir**  Where the data should be stored. Defaults to NULL, meaning no data is stored.

**param.track**  Which parameters to track. Should be a vector of strings, containing no spaces or commas

**Value**

A S3 object `DataSet`

**Examples**

```r
exp <- IOHexperimenter()
```

---

**Description**

If an ‘experimenter’-argument is provided, this is the same function as ‘next_problem’ If not, this creates a suite consisting of a single function based on the other arguments

**Usage**

```r
IOHproblem(suite = "PBO", functionnr = 1, dim = 16, instance = 1,
            experimenter = NULL)
```

**Arguments**

- **suite**  The suite to use. Either 'PBO' or 'BBOB'
- **functionnr**  The number of the function to create
- **dim**  The dimension of the function to create
- **instance**  The instance of the function to create
- **experimenter**  (optional) an IOHexperimenter object

**Details**

An IOHproblem-object has the following attributes:

- "Dimension": The dimension of the problem
- "function_id": The number of the function
- "instance": The number of the function-instance
- **suite**: The suite of the function. Either 'PBO' or 'BBOB'
- **fopt**: If known, the optimal value of the function
- **lboud**: The lower bound of the searchspace
- **ubound**: The upper bound of the searchspace
• maximization: Boolean indicating whether the function should be maximized or minimized
• params.track: The parameters which are being tracked on this function. Only available if initialized in the underlying IOHexperimenter-object (or when using the ‘benchmark_algorithm’-function)

In addition to these attributes, there are three function-attributes available to use:

• obj_function: The interface to evaluate the function
• target_hit: Boolean indicating if the optimal has been hit (if known)
• set_parameters: Interface to storing the current parameter values (if param.track is initialized).
  This has two arguments: the list of names of parameters to update (must match those of param.track) and a list of equal length containing their respective values.

Value
An IOHproblem object

Examples

```r
p <- IOHproblem()
```

---

**Description**

For easier use with the IOHexperimenter

The simplest stochastic optimization algorithm for discrete problems. A randomly chosen position in the solution vector is perturbated in each iteration. Only improvements are accepted after perturbation.

**Usage**

```r
IOH_random_local_search(IOHproblem, budget = NULL)
random_local_search(dimension, obj_func, target_hit = function() {
  FALSE }, budget = NULL)
```

**Arguments**

- `IOHproblem`: An IOHproblem object
- `budget`: integer, maximal allowable number of function evaluations
- `dimension`: Dimension of search space
- `obj_func`: The evaluation function
- `target_hit`: Optional, function which enables early stopping if a target value is reached
**IOH_random_search**

**Examples**

```r
benchmark_algorithm(IOH_random_local_search, data.dir = NULL)
```

---

<table>
<thead>
<tr>
<th><strong>IOH_random_search</strong></th>
<th><strong>IOHexperimenter-based wrapper</strong></th>
</tr>
</thead>
</table>

**Description**

For easier use with the IOHexperimenter

Random walk in $0,1^d$ space; Maximization

Random walk in continuous space;

**Usage**

```r
IOH_random_search(IOHproblem, budget = NULL)
```

```r
random_search_PB(dim, obj_func, target_hit = function() { FALSE },
                 budget = NULL)
```

```r
random_search(dim, obj_func, target_hit = function() { FALSE },
              budget = NULL, lbound = -1, ubound = 1, maximize = T)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOHproblem</td>
<td>An IOHproblem object</td>
</tr>
<tr>
<td>budget</td>
<td>Integer, maximal allowable number of function evaluations</td>
</tr>
<tr>
<td>dim</td>
<td>Dimension of search space</td>
</tr>
<tr>
<td>obj_func</td>
<td>The evaluation function</td>
</tr>
<tr>
<td>target_hit</td>
<td>Optional, function which enables early stopping if a target value is reached</td>
</tr>
<tr>
<td>lbound</td>
<td>Lower bound of search space. Either single number or vector of size 'dim'</td>
</tr>
<tr>
<td>ubound</td>
<td>Upper bound of search space. Either single number or vector of size 'dim'</td>
</tr>
<tr>
<td>maximize</td>
<td>Whether to perform maximization or minimization. The function assumes minimization, achieved by inverting the obj_func when 'maximize' is FALSE</td>
</tr>
</tbody>
</table>

**Examples**

```r
benchmark_algorithm(IOH_random_search, data.dir = NULL)
```
IOH_self_adaptive_GA  

**IOHexperimenter-based wrapper**

**Description**

For easier use with the IOHexperimenter

A genetic algorithm that controls the mutation rate (strength) using the so-called self-adaptation mechanism: the mutation rate is firstly perturbated and then the resulting value is taken to mutate Lambda solution vector. The best solution is selected along with its mutation rate.

**Usage**

```r
IOH_self_adaptive_GA(IOHproblem, lambda_ = 1, budget = NULL)

self_adaptive_GA(dimension, obj_func, lambda_ = 10, budget = NULL,
set_parameters = NULL, target_hit = function() { FALSE })
```

**Arguments**

- `IOHproblem`: An IOHproblem object
- `lambda_`: The size of the offspring
- `budget`: How many times the objective function can be evaluated
- `dimension`: Dimension of search space
- `obj_func`: The evaluation function
- `set_parameters`: Function to call to store the value of the registered parameters
- `target_hit`: Optional, function which enables early stopping if a target value is reached

**Examples**

```r
one_comma_two_EA <- function(IOHproblem) { IOH_self_adaptive_GA(IOHproblem, lambda_=2) }

benchmark_algorithm(one_comma_two_EA, params.track = "Mutation_rate",
algorithm.name = "one_comma_two_EA", data.dir = NULL,
algorithm.info = "Using one_comma_two_EA with specific parameter")
```
Description

For easier use with the IOHexperimenter

A genetic algorithm that controls the mutation rate (strength) using the so-called 2-rate self-adaptation mechanism: the mutation rate is based on a parameter \( r \). For each generation, half offspring are generated by mutation rate \( 2r/dim \), and half by \( r/2dim \). \( r \) that the best offspring has been created with will be inherited by probability 3/4, the other by 1/4.

Usage

\[
\text{IOH\_two\_rate\_GA}(\text{IOHproblem}, \lambda = 1, \text{budget} = \text{NULL})
\]

\[
\text{two\_rate\_GA}(\text{dimension}, \text{obj\_func}, \text{target\_hit} = \text{function}() \{ \quad \text{FALSE} \},
\quad \lambda = 2, \text{budget} = \text{NULL}, \text{set\_parameters} = \text{NULL})
\]

Arguments

- \text{IOHproblem} An IOHproblem object
- \lambda \_ The size of the offspring
- \text{budget} How many times the objective function can be evaluated
- \text{dimension} Dimension of search space
- \text{obj\_func} The evaluation function
- \text{target\_hit} Optional, function which enables early stopping if a target value is reached
- \text{set\_parameters} Function to call to store the value of the registered parameters

Examples

\[
\text{benchmark\_algorithm}(\text{IOH\_two\_rate\_GA})
\]

next\_problem \hspace{1cm} Get the next function of the currently initialized IOHexperimenter object

Description

Get the next function of the currently initialized IOHexperimenter object
Usage

next_problem(experimenter)

Arguments

experimenter   The IOHexperimenter object

Value

An IOHproblem object if available, NULL otherwise

Examples

exp <- IOHexperimenter()
p <- next_problem(exp)

print.IOHexperimenter  S3 print function for IOHexperimenter

Description

S3 print function for IOHexperimenter

Usage

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

Arguments

x      The IOHexperimenter to print
...
     Arguments for underlying function

Examples

print(IOHexperimenter())
**print.IOHproblem**

_S3 print function for IOHproblem_

**Description**

S3 print function for IOHproblem

**Usage**

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

**Arguments**

- `x`: The IOHproblem to print
- `...`: Arguments for underlying function

**Examples**

```r
exp <- IOHexperimenter()
p <- next_problem(exp)
print(p)
```

---

**reset_problem**

_Reset the IOHproblem_

**Description**

Reset the IOHproblem

**Usage**

```r
reset_problem(problem)
```

**Arguments**

- `problem`: The IOHproblem object

**Value**

An IOHproblem object

**Examples**

```r
exp <- IOHexperimenter()
p <- next_problem(exp)
IOH_random_search(p)
p <- reset_problem(p)
```
Description

S3 generic summary operator for IOHexperimenter

Usage

```r
## S3 method for class 'IOHexperimenter'
summary(object, ...)
```

Arguments

- `object`: A IOHexperimenter object
- `...`: Arguments passed to other methods

Value

A summary of the IOHexperimenter object.

Examples

```r
summary(IOHexperimenter())
```
Index

as.character.IOHexperimenter, 2
as.character.IOHproblem, 3

benchmark_algorithm, 3

IOH_random_local_search, 6
IOH_random_search, 7
IOH_self_adaptive_GA, 8
IOH_two_rate_GA, 9
IOHexperimenter, 4
IOHproblem, 5

next_problem, 9

print.IOHexperimenter, 10
print.IOHproblem, 11

random_local_search
 (IOH_random_local_search), 6
random_search (IOH_random_search), 7
reset_problem, 11

self_adaptive_GA
 (IOH_self_adaptive_GA), 8
summary.IOHexperimenter, 12

two_rate_GA (IOH_two_rate_GA), 9