Package: simmer.optim (via r-universe)

August 28, 2024

Type Package

Title Parameter Optimization Functions for 'simmer'

Version 0.1.1

Description A set of optimization functions for variable optimization in simmer simulations.

License MIT + file LICENSE

Encoding UTF-8

URL http://r-simmer.org, https://github.com/r-simmer/simmer.optim

BugReports https://github.com/r-simmer/simmer.optim/issues

Depends R (>= 3.1.2), simmer (>= 3.6.0)

Imports dplyr, tidyr, methods, stats, utils

Suggests simmer.plot (>= 0.1.12), knitr, rmarkdown, testthat, RcppDE, GenSA

VignetteBuilder knitr

RoxygenNote 7.1.2

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

RemoteUrl https://github.com/r-simmer/simmer.optim

RemoteRef HEAD

RemoteSha 2a1334af365bfa4850461bc55ca8e1c27d640ded

Contents

`assert_avg_waiting_time_max` *Assert that average waiting time < max_val*

Description

Assert that average waiting time < max_val

Usage

```
assert_avg_waiting_time_max(envs, max_val)
```

Arguments

`envs`	a list of `envs` as produced by `run_instance`
`max_val`	the max waiting time duration

`assert_waiting_time_max` *Assert that waiting time < max_val*

Description

Assert that waiting time < max_val

Usage

```
assert_waiting_time_max(envs, max_val)
```

Arguments

envs	a list of envs as produced by run_instance
max_val	the max waiting time duration

---

constraints_evaluator *Evaluator for the constraint functions*

Description

Evaluator for the constraint functions

Usage

```
constraints_evaluator(envs, constraints)
```

Arguments

envs	a list of envs as produced by run_instance
constraints	a list of constraint functions

---

de_optim *A simmer differential evolution optimizer*

Description

Implements the functionality of the DEoptim package.

Usage

```
de_optim(
  model,
  direction = c("min", "max"),
  objective,
  constraints,
  params,
  control,
  big_m = 1e+06
)
```

Arguments

model	the simmer model encapsulated in a function
direction	optimization direction (max or min)
objective	the objective function
constraints	a list of constraint functions
params	a list of parameters to optimize over
control	a control list created by a call to optim_control()
big_m	a penalty value for solutions with non-satisfied constraints

---

grid_optim *A simmer grid optimizer*

Description

Executes an exhaustive search over the solution space

Usage

```
grid_optim(
  model,
  direction = c("min", "max"),
  objective,
  constraints,
  params,
  control
)
```

Arguments

model	the simmer model encapsulated in a function
direction	optimization direction (max or min)
objective	the objective function
constraints	a list of constraint functions
params	a list of parameters to optimize over
control	a control list created by a call to optim_control()

method_results *A helper function to return the results of an optimization method to the optimization framework*

Description

A helper function to return the results of an optimization method to the optimization framework

Usage

```
method_results(
  method,
  objective_value,
  constraints_satisfied,
  params,
  envs = NULL,
  extra_info = list()
)
```

Arguments

method	the name of the optimization function (string)
objective_value	the value of the objective
constraints_satisfied	boolean indicating whether or not all constraints were satisfied
params	the found parameters
envs	a copy of the generated envs (optional)
extra_info	a list of extra information (optional)

msr_arrivals_finished *Measure the number of finished arrivals*

Description

Measure the number of finished arrivals

Usage

```
msr_arrivals_finished(envs, agg = mean)
```

Arguments

envs	a list of envs as produced by run_instance
agg	the method of aggregation of per replication results

msr_arrivals_rejected *Measure the number of rejected arrivals*

Description

Measure the number of rejected arrivals

Usage

```
msr_arrivals_rejected(envs, agg = mean)
```

Arguments

envs	a list of envs as produced by run_instance
agg	the method of aggregation of per replication results

msr_resource_capacity *Measure the capacity of a resource type*

Description

Measure the capacity of a resource type

Usage

```
msr_resource_capacity(envs, name, agg = mean)
```

Arguments

envs	a list of envs as produced by run_instance
name	the name of the resource
agg	the method of aggregation of per replication results

msr_resource_utilization *Measure the utilization of a resource type*

Description

Measure the utilization of a resource type

Usage

```
msr_resource_utilization(envs, name, agg = mean)
```

Arguments

envs	a list of envs as produced by run_instance
name	the name of the resource
agg	the method of aggregation of per replication results

msr_runtime *Measure the runtime of the model*

Description

Measure the runtime of the model

Usage

```
msr_runtime(envs, agg = mean)
```

Arguments

envs	a list of envs as produced by run_instance
agg	the method of aggregation of per replication results

objective_evaluator *Evaluator for the objective function*

Description

Evaluator for the objective function

Usage

```
objective_evaluator(envs, objective)
```

Arguments

envs	a list of envs as produced by run_instance
objective	an objective function

optim_control *Control object to configure the optimization procedure*

Description

Control object to configure the optimization procedure

Usage

```
optim_control(
  run_args = list(until = 1000),
  replications = 1,
  parallel = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

run_args	the run arguments, a list with the following arguments: until
replications	the number of replications
parallel	whether or not replications should be run in parallel (leveraging mclapply)
verbose	boolean determining verbosity
...	extra named arguments added to the control object

optim_results — *Function to pass results of simmer evaluation back to the optimization framework*

Description

Function to pass results of simmer evaluation back to the optimization framework

Usage

```
optim_results(objective, constraints = list(), envs = NULL)
```

Arguments

objective	the value of the objective
constraints	a list with named objectives containing only TRUE and FALSE values
envs	the simmer env (OPTIONAL)

opt_func — *Value function generator*

Description

Internal usage

Usage

```
opt_func(params)
```

Arguments

params	a named list of params

par_continuous — *A parameter vector of type continuous*

Description

A parameter vector of type continuous

Usage

```
par_continuous(vec)
```

Arguments

vec	the original vector

par_discrete *A parameter vector of type discrete*

Description

A parameter vector of type discrete

Usage

```
par_discrete(vec)
```

Arguments

vec	the original vector

results *Show the results of an optimization procedure*

Description

Show the results of an optimization procedure

Usage

```
results(optim_obj)
```

Arguments

optim_obj	the optimization object

run_instance *Run n simmer models*

Description

Run n simmer models

Usage

```
run_instance(model, control, params)
```

Arguments

model	the simmer model
control	the optim_control object
params	a list of named parameters to be passed to the simmer expression and accessible for the model through the .opt variable

run_optimized *(re)run a simmer expression using the optimized parameter list*

Description

For this to work an envs object has to be returned with the optim_results

Usage

```
run_optimized(optim_obj)
```

Arguments

optim_obj	the optimization object

See Also

results

sa_optim *A simmer simulated annealing optimizer*

Description

Implements the functionality of the GenSA package.

Usage

```
sa_optim(
  model,
  direction = c("min", "max"),
  objective,
  constraints,
  params,
  control,
  big_m = 1e+06
)
```

Arguments

model	the simmer model encapsulated in a function
direction	optimization direction (max or min)
objective	the objective function
constraints	a list of constraint functions
params	a list of parameters to optimize over

control	a control list created by a call to optim_control()
big_m	a penalty value for solutions with non-satisfied constraints

---

simmer_optim *Main entry point for the simmer optimization framework*

Description

Main entry point for the simmer optimization framework

Usage

```
simmer_optim(
  model,
  method,
  direction = "max",
  objective = msr_arrivals_finished,
  constraints,
  params = list(),
  control = optim_control(),
  ...
)
```

Arguments

model	the simmer model encapsulated in a function
method	the method to be used (e.g. grid_optim)
direction	optimization direction (max or min)
objective	the objective function
constraints	a list of constraint functions
params	a list of parameters to optimize over
control	a control list created by a call to optim_control()
...	extra parameters that are passed on to the optimization procedure

`with_args` *Helper function to run objective / constraint functions with specified arguments*

Description

Helper function to run objective / constraint functions with specified arguments

Usage

```
with_args(f, ...)
```

Arguments

`f`	the objective / constraint function
`...`	a list of named arguments which will be used in the call to `f`

Index