

Package ‘tabuSearch’

January 15, 2026

Title Tabu Search Algorithm for Binary Configurations

Version 1.2.0

Description Tabu search algorithm for binary configurations. A basic version of the algorithm as described by Fouskakis and Draper (2007) <[doi:10.1111/j.1751-5823.2002.tb00174.x](https://doi.org/10.1111/j.1751-5823.2002.tb00174.x)>.

License GPL (>= 2)

LazyLoad yes

URL <https://github.com/domijan/tabuSearch>

BugReports <https://github.com/domijan/tabuSearch/issues>

Encoding UTF-8

RoxygenNote 7.3.3

Imports graphics, grDevices

NeedsCompilation no

Author Katarina Domijan [aut, cre] (ORCID:
<<https://orcid.org/0000-0002-4268-2236>>)

Maintainer Katarina Domijan <domijank@tcd.ie>

Repository CRAN

Date/Publication 2026-01-15 14:30:02 UTC

Contents

plot.tabu	2
summary.tabu	2
tabuSearch	3

Index	6
--------------	----------

`plot.tabu`*R Based Tabu Search Plot Function*

Description

Plots features of an optimization run of the tabu search algorithm for binary strings. The default plots show: (a) the number of times each element of the string was set to one over the search, (b) frequency of moves for each element of the string over the search, (c) the number of ones in the chosen configuration at each iteration, (d) the objective function value of the current configuration at each iteration of the algorithm. The "tracePlot" shows the current configurations for all iterations.

Usage

```
## S3 method for class 'tabu'  
plot(x, type = "default", ...)
```

Arguments

<code>x</code>	a tabu object.
<code>type</code>	"tracePlot" or default.
<code>...</code>	options directly passed to the plot function.

Examples

```
# A simple example  
  
evaluateSimple <- function(th) return(1)  
result <- tabuSearch(size = 20, iters = 100, objFunc = evaluateSimple)  
  
plot(result)  
plot(result, "tracePlot")
```

`summary.tabu`*R Based Tabu Search Summary Function*

Description

Summarizes the results of a tabu search optimization run.

Usage

```
## S3 method for class 'tabu'  
summary(object, verbose = FALSE, ...)
```

Arguments

object a tabu object.
verbose if true, the optimal configuration(s) will be printed.
... other options (ignored).

Value

a matrix of configurations from iterations with maximum value of objective function (i.e. Optimum configuration).

Examples

```
# A simple example

evaluateSimple <- function(th) return(1)
result <- tabuSearch(size = 20, iters = 100, objFunc = evaluateSimple)

summary(result)
summary(result, verbose = TRUE)
```

tabuSearch

R based tabu search algorithm for binary configurations

Description

A tabu search algorithm for optimizing binary strings. It takes a user defined objective function and reports the best binary configuration found throughout the search i.e. the one with the highest objective function value. The algorithm can be used for variable selection. The results can be plotted and summarized using `plot.tabu` and `summary.tabu`.

Usage

```
tabuSearch(  
  size = 10,  
  iters = 100,  
  objFunc = NULL,  
  config = NULL,  
  neigh = size,  
  listSize = 9,  
  nRestarts = 10,  
  repeatAll = 1,  
  verbose = FALSE  
)
```

Arguments

size	the length of the binary configuration.
iters	the number of iterations in the preliminary search of the algorithm.
objFunc	a user supplied method that evaluates the objective function for a given binary string. The objective function is required to take as an argument a vector of zeros and ones.
config	a starting configuration.
neigh	a number of neighbour configurations to check at each iteration. The default is all, which is the length of the string. If <code>neigh < size</code> , the neighbours are chosen at random.
listSize	tabu list size.
nRestarts	the maximum number of restarts in the intensification stage of the algorithm.
repeatAll	the number of times to repeat the search.
verbose	if true, the name of the current stage of the algorithm is printed e.g. preliminary stage, intensification stage, diversification stage.

Value

A tabu object which is a list giving:

- `configKeep` a matrix of configurations chosen at each iteration of the algorithm.
- `eUtilityKeep` value of the objective function for the above configurations.
- `iters` the number of iterations in the preliminary search of the algorithm.
- `neigh` a number of neighbour configurations checked at each iteration.
- `listSize` tabu list size.
- `repeatAll` the number of times the search was repeated.

Author(s)

K. Domijan

References

- Glover, F., 1977. Heuristics for integer programming using surrogate constraints. *Decision Sciences* 8, 156-166.
- Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research* 13, 533-549.
- Fouskakis, D., Draper, D., 2002. Stochastic optimization: a review. *International Statistical Review* 70, 315-349.

See Also

[plot.tabu\(\)](#) [summary.tabu\(\)](#)

Examples

```

# A simple example
evaluateSimple <- function(th)return(1)
result <- tabuSearch(size = 20, iters = 100, objFunc = evaluateSimple)
## Not run:
# simulate 10-d data: 150 samples from 3 bivariate normals and 8 noise variables.
# Variable selection should recover the first two variables
library(MASS)
NF <- 10
G <- 3
NTR <- 50
NTE <- 50
muA <- c(1,3)
SigmaA <- matrix(c(0.2, 0.04, 0.04, 0.2), 2, 2)
muB <- c(1.2,1)
SigmaB <- matrix(c(0.1, -0.06, 0.004, 0.2), 2, 2)
muC <- c(3,2)
SigmaC <- matrix(c(0.2, 0.004, 0.004, 0.2), 2, 2)

train <- rbind(mvrnorm(NTR, muA, SigmaA), mvrnorm(NTR, muB, SigmaB), mvrnorm(NTR, muC, SigmaC))
test <- rbind(mvrnorm(NTE, muA, SigmaA), mvrnorm(NTE, muB, SigmaB), mvrnorm(NTE, muC, SigmaC))

train <- cbind(train, matrix(runif(G * NTR * (NF - 2), 0, 4), nrow = G * NTR, ncol = (NF-2)))
test <- cbind(test, matrix(runif(G * NTE * (NF - 2), 0, 4), nrow = G * NTE, ncol = (NF-2)))

wtr <- as.factor(rep(1:G, each = NTR))
wte <- as.factor(rep(1:G, each = NTE))
pairs(train, col = wtr)

library(e1071)

evaluate <- function(th){
  if (sum(th) == 0)return(0)
  model <- svm(train[,th==1], wtr , gamma = 0.1)
  pred <- predict(model, test[,th==1])
  csRate <- sum(pred == wte)/NTE
  penalty <- (NF - sum(th))/NF
  return(csRate + penalty)
}
res <- tabuSearch(size = NF, iters = 50, objFunc = evaluate, config = matrix(1,1,NF),
  listSize = 5, nRestarts = 4)

plot(res)
plot(res, "tracePlot")
summary(res, verbose = TRUE)

## End(Not run)

```

Index

`plot.tabu`, 2
`plot.tabu()`, 4

`summary.tabu`, 2
`summary.tabu()`, 4

`tabuSearch`, 3