Package 'eaf'

March 22, 2018

Type Package

Title Plots of the Empirical Attainment Function

Version 1.8

Date 2018-03-22

Description Plots of the empirical attainment function for two objectives.

Depends R (>= 2.10.0)

Imports modeltools, graphics, grDevices, stats

Suggests testthat

License GPL (>= 2)

BugReports https://github.com/MLopez-Ibanez/eaf/issues

URL http://lopez-ibanez.eu/eaftools,
https://github.com/MLopez-Ibanez/eaf

LazyLoad true

LazyData true

Encoding UTF-8

RoxygenNote 6.0.1

NeedsCompilation yes

Author Manuel López-Ibáñez [aut, cre]
(<https://orcid.org/0000-0001-9974-1295>),
Marco Chiarandini [aut],
Carlos Fonseca [aut],
Luis Paquete [aut],
Thomas Stützle [aut]

Maintainer Manuel López-Ibáñez <manuel.lopez-ibanez@manchester.ac.uk>

Repository CRAN

Date/Publication 2018-03-22 20:18:51 UTC

R topics documented:

`eaf-package` *Plots of the Empirical Attainment Function*

Description

The empirical attainment function (EAF) describes the probabilistic distribution of the outcomes obtained by a stochastic algorithm in the objective space. This package implements plots of summary attainment surfaces and differences between the first-order EAFs. These plots may be used for exploring the performance of stochastic local search algorithms for biobjective optimization problems and help in identifying certain algorithmic behaviors in a graphical way.

Details

Functions:

`eafdiffplot`	Empirical attainment function differences
`eafplot`	Plot the Empirical Attainment Function for two objectives
`read.data.sets`	Read several data.frame sets

Data:

`gcp2x2` Metaheuristics for solving the Graph Vertex Coloring Problem

`HybridGA` Results of Hybrid GA on vanzyl and Richmond water networks

`SPEA2minstoptimeRichmond` Results of SPEA2 when minimising electrical cost and maximising the minimum idle time of pumps on Richmond water network

Extras are available at `file.path(system.file(package="eaf"))`:

`extdata`	External data sets (see `read.data.sets`)
`scripts/eaf`	EAF command-line program
`scripts/eafplot`	Perl script to generate plots of attainment surfaces
`scripts/eafdiff`	Perl script to generate plots of EAF differences

Author(s)

Maintainer: Manuel López-Ibáñez <manuel.lopez-ibanez@ulb.ac.be>

Contributors: Carlos Fonseca, Luis Paquete, Thomas Stützle, Manuel López-Ibáñez and Marco Chiarandini.

References

V. Grunert da Fonseca, C. M. Fonseca, and A. O. Hall, *Inferential performance assessment of stochastic optimisers and the attainment function*, in Evolutionary Multi-Criterion Optimization. First International Conference, EMO 2001 (E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, eds.), vol. 1993 of Lecture Notes in Computer Science, pp. 213-225, Berlin: Springer, 2001.

V. Grunert da Fonseca and C. M. Fonseca, *The attainment-function approach to stochastic multiobjective optimizer assessment and comparison*. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, Experimental Methods for the Analysis of Optimization Algorithms, pages 103-130, Springer, Berlin, Germany, 2010.

M. López-Ibáñez, L. Paquete, and T. Stützle. *Exploratory Analysis of Stochastic Local Search Algorithms in Biobjective Optimization*. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, Experimental Methods for the Analysis of Optimization Algorithms, pages 209–222. Springer, Berlin, Germany, 2010. doi: 10.1007/978-3-642-02538-9_9

See Also

Useful links:

- http://lopez-ibanez.eu/eaftools
- https://github.com/MLopez-Ibanez/eaf
- Report bugs at https://github.com/MLopez-Ibanez/eaf/issues

Examples

```
data(gcp2x2)
tabucol<-subset(gcp2x2, alg!="TSinN1")
tabucol$alg<-tabucol$alg[drop=TRUE]
eafplot(time+best~run,data=tabucol,subset=tabucol$inst=="DSJC500.5")

eafplot(time+best~run|inst,groups=alg,data=gcp2x2)
eafplot(time+best~run|inst,groups=alg,data=gcp2x2,
percentiles=c(0,50,100),include.extremes=TRUE,
cex=1.4, lty=c(2,1,2),lwd=c(2,2,2),
       col=c("black","blue","grey50"))

A1<-read.data.sets(file.path(system.file(package="eaf"),"extdata","ALG_1_dat"))
A2<-read.data.sets(file.path(system.file(package="eaf"),"extdata","ALG_2_dat"))
eafplot(A1,A2, percentiles=c(50))
eafplot(list(A1=A1, A2=A2), percentiles=c(50))
eafdiffplot(A1, A2)
## Save to a PDF file
# dev.copy2pdf(file="eaf.pdf", onefile=TRUE, width=5, height=4)
```

`eafdiffplot` *Empirical attainment function differences*

Description

Plot the differences between the empirical attainment functions of two data sets as a two-panel plot, where the left side shows the values of the left EAF minus the right EAF and the right side shows the differences in the other direction.

Usage

```
eafdiffplot(data.left, data.right, col = c("#FFFFFF", "#808080", "#000000"),
  intervals = 5, percentiles = c(50), full.eaf = FALSE, type = "area",
  legend.pos = if (full.eaf) "bottomleft" else "topright",
  title.left = deparse(substitute(data.left)),
  title.right = deparse(substitute(data.right)), xlim = NULL, ylim = NULL,
  cex = par("cex"), cex.lab = par("cex.lab"), cex.axis = par("cex.axis"),
  maximise = c(FALSE, FALSE), grand.lines = TRUE, left.panel.last = NULL,
  right.panel.last = NULL, ...)
```

Arguments

`data.left, data.right`
: Data frames corresponding to the input data of left and right sides, respectively. Each data frame has at least three columns, the third one being the set of each point. See also `read.data.sets`.

`col`
: A character vector of three colors for the magnitude of the differences of 0, 0.5, and 1. Intermediate colors are computed automatically given the value of `intervals`.

`intervals`
: An integer or a character vector. The absolute range of the differences [0,1] is partitioned into the number of intervals provided. If an integer is provided, then labels for each interval are computed automatically. If a character vector is provided, its length is taken as the number of intervals.

`percentiles`
: The percentiles of the EAF of each side that will be plotted as attainment surfaces. `NA` does not plot any. See `eafplot.default`.

`full.eaf`
: Whether to plot the EAF of each side instead of the differences between the EAFs.

`type`
: Whether the EAF differences are plotted as points (‘`points`’) or whether to color the areas that have at least a certain value (‘`area`’).

`legend.pos`
: The position of the legend. See `legend`.

`title.left, title.right`
: Title for left and right panels, respectively.

`xlim, ylim, cex, cex.lab, cex.axis`
: Graphical parameters, see `plot.default`.

`maximise`
: Whether the first and/or second objective correspond to a maximisation problem.

`grand.lines`
: Whether to plot the grand-best and grand-worst attainment surfaces.

`left.panel.last, right.panel.last`
: An expression to be evaluated after plotting has taken place on each panel (left or right). This can be useful for adding points or text to either panel. Note that this works by lazy evaluation: passing this argument from other `plot` methods may well not work since it may be evaluated too early.

`...`
: Other graphical parameters are passed down to `plot.default`.

Details

This function calculates the differences between the EAFs of two data sets, and plots on the left the differences in favour of the left data set, and on the right the differences in favour of the right data set. By default, it also plots the grand best and worst attainment surfaces, that is, the 0% and 100%-attainment surfaces over all data. This two surfaces delimit the area where differences may exist. In addition, it also plots the 50%-attainment surface of each data set.

With `type = "point"`, only the points where there is a change in the value of the EAF difference are plotted. This means that for areas where the EAF differences stays constant, the region will appear in white even if the value of the differences in that region is large. This explain "white holes" surrounded by black points.

With `type = "area"`, the area where the EAF differences has a certain value is plotted. The idea for the algorithm to compute the areas was provided by Carlos M. Fonseca. The implementation uses R polygons, which some PDF viewers may have trouble rendering correctly (See https://cran.r-project.org/doc/FAQ/R-FAQ.html#Why-are-there-unwanted-borders). Plots (should) look correct when printed.

Large differences that appear when using `type = "points"` may seem to dissapear when using `type = "area"`. The explanation is the points size is independent of the axes range, therefore, the plotted points may seem to cover a much larger area than the actual number of points. On the other hand, the areas size is plotted with respect to the objective space, without any extra borders. If the range of an area becomes smaller than one-pixel, it won't be visible. As a consequence, zooming in or out certain regions of the plots does not change the apparent size of the points, whereas it affects considerably the apparent size of the areas.

Value

No return value.

See Also

`read.data.sets`, `eafplot`

Examples

```
A1 <- read.data.sets(file.path(system.file(package="eaf"), "extdata", "ALG_1_dat"))
A2 <- read.data.sets(file.path(system.file(package="eaf"), "extdata", "ALG_2_dat"))
# These take time
```

```
eafdiffplot(A1, A2, full.eaf = TRUE)
eafdiffplot(A1, A2, type = "area")
eafdiffplot(A1, A2, type = "point")

# A more complex example
a1 <- read.data.sets(file.path(system.file(package="eaf"), "extdata", "wrots_l100w10_dat"))
a2 <- read.data.sets(file.path(system.file(package="eaf"), "extdata", "wrots_l10w100_dat"))
DIFF <- eafdiffplot(a1, a2, col = c("white", "blue", "red"), intervals = 5,
type = "point",
         title.left = expression("W-RoTS, " * lambda * "=" * 100 * ", " * omega * "=" * 10),
           title.right= expression("W-RoTS, " * lambda * "=" * 10 *
", " * omega * "=" * 100),
           right.panel.last={ abline(a = 0, b = 1, col = "red", lty = "dashed")})
DIFF$right[,3] <- -DIFF$right[,3]

## Save the values to a file.
# write.table(rbind(DIFF$left,DIFF$right),
#             file = "wrots_l100w10_dat-wrots_l10w100_dat-diff.txt",
#             quote = FALSE, row.names = FALSE, col.names = FALSE)
```

---

eafplot *Plot the Empirical Attainment Function for two objectives generic*

Description

Computes and plots the Empirical Attainment Function, either as attainment surfaces for certain percentiles or as points.

Usage

```
eafplot(x, ...)

## S3 method for class 'formula'
eafplot(formula, data, groups = NULL, subset = NULL, ...)

## S3 method for class 'list'
eafplot(x, ...)

## S3 method for class 'data.frame'
eafplot(x, y = NULL, ...)

## Default S3 method:
eafplot(x, sets = NULL, groups = NULL,
  percentiles = c(0, 50, 100), attsurfs = NULL, xlab = "objective 1",
  ylab = "objective 2", xlim = NULL, ylim = NULL, log = "",
  type = "point", col = NULL, lty = c("dashed", "solid", "solid", "solid",
  "dashed"), lwd = c(1.75), pch = NA, cex.pch = par("cex"),
  las = par("las"), legend.pos = "topright", legend.txt = NULL,
```

```
extra.points = NULL, extra.legend = NULL, extra.pch = c(4:25),
extra.lwd = 0.5, extra.lty = "dashed", extra.col = "black",
maximise = c(FALSE, FALSE), xaxis.side = "below", yaxis.side = "left",
axes = TRUE, ...)
```

Arguments

x	Either a matrix of data values, or a data frame, or a list of data frames of exactly three columns.
...	Other graphical parameters to plot.default.
formula	A formula of the type: time + cost ~ run | instance will draw time on the x-axis and cost on the y-axis. If instance is present the plot is conditional to the instances.
data	Dataframe containing the fields mentioned in the formula and in groups.
groups	This may be used to plot profiles of different algorithms on the same plot.
subset	A vector indicating which rows of the data should be used. If left to default NULL all data in the data frame are used.
y	Either a matrix of data values, or a data frame.
sets	Vector indicating which set each point belongs to.
percentiles	Vector indicating which percentile should be plot. The default is to plot only the median attainment curve.
attsurfs	TODO
xlab, ylab, xlim, ylim, log, col, lty, lwd, pch, cex.pch, las	Graphical parameters, see plot.default.
type	string giving the type of plot desired. The following values are possible, 'points' and 'area'.
legend.pos	the position of the legend, see legend.
legend.txt	a character or expression vector to appear in the legend. If NULL, appropriate labels will be generated.
extra.points	A list of matrices or data.frames with two-columns. Each element of the list defines a set of points, or lines if one of the columns is NA.
extra.legend	A character vector providing labels for the groups of points.
extra.pch, extra.lwd, extra.lty, extra.col	Control the graphical aspect of the points. See points and lines.
maximise	Whether the first and/or second objective correspond to a maximisation problem.
xaxis.side	On which side that xaxis is drawn. Valid values are "below" and "above". See axis.
yaxis.side	On which side that yaxis is drawn. Valid values are "left" and "right". See axis.
axes	A logical value indicating whether both axes should be drawn on the plot.

Details

This function can be used to plot random sets of points like those obtained by different runs of biobjective stochastic optimization algorithms. An EAF curve represents the boundary separating points that are known to be attainable (that is, dominated in Pareto sense) in at least a fraction (quantile) of the runs from those that are not. The median EAF represents the curve where the fraction of attainable points is 50%. In single objective optimization the function can be used to plot the profile of solution quality over time of a collection of runs of a stochastic optimizer.

Value

No value is returned.

Methods (by class)

- `formula`: Formula interface
- `list`: List interface for lists of data.frames
- `data.frame`: Data.frame interface
- `default`: Main function

See Also

`read.data.sets` `eafdiffplot`

Examples

```
data(gcp2x2)
tabucol <- subset(gcp2x2, alg != "TSinN1")
tabucol$alg <- tabucol$alg[drop=TRUE]
eafplot(time+best~run,data=tabucol,subset=tabucol$inst=="DSJC500.5")

## Not run:  # These take time
eafplot(time+best~run|inst,groups=alg,data=gcp2x2)
eafplot(time+best~run|inst,groups=alg,data=gcp2x2,
percentiles=c(0,50,100),include.extremes=TRUE,
cex=1.4, lty=c(2,1,2),lwd=c(2,2,2),
        col=c("black","blue","grey50"))

A1 <- read.data.sets(file.path(system.file(package = "eaf"), "extdata", "ALG_1_dat"))
A2 <- read.data.sets(file.path(system.file(package = "eaf"), "extdata", "ALG_2_dat"))
eafplot(A1, A2, percentiles = c(50))
eafplot(list(A1 = A1, A2 = A2), percentiles = c(50))

## Save as a PDF file.
# dev.copy2pdf(file = "eaf.pdf", onefile = TRUE, width = 5, height = 4)

## End(Not run)

## Using extra.points
## Not run:
data(HybridGA)
```

```
data(SPEA2relativeVanzyl)
eafplot(SPEA2relativeVanzyl, percentiles = c(25, 50, 75),
        xlab = expression(C[E]), ylab = "Total switches", xlim = c(320, 400),
        extra.points = HybridGA$vanzyl, extra.legend = "Hybrid GA")

data(SPEA2relativeRichmond)
eafplot (SPEA2relativeRichmond, percentiles = c(25, 50, 75),
         xlab = expression(C[E]), ylab = "Total switches",
         xlim = c(90, 140), ylim = c(0, 25),
         extra.points = HybridGA$richmond, extra.lty = "dashed",
         extra.legend = "Hybrid GA")

data(SPEA2minstoptimeRichmond)
SPEA2minstoptimeRichmond[,2] <- SPEA2minstoptimeRichmond[,2] / 60
eafplot (SPEA2minstoptimeRichmond, xlab = expression(C[E]),
         ylab = "Minimum idle time (minutes)",
         las = 1, log = "y", maximise = c(FALSE, TRUE), main = "SPEA2 (Richmond)")

## End(Not run)
```

`eafs` *Exact computation of the EAF*

Description

This function computes the EAF given a set of points and a vector `set` that indicates to which set each point belongs.

Usage

```
eafs(points, sets, groups = NULL, percentiles = NULL)
```

Arguments

`points`	Either a matrix or a data frame of numerical values, where each row gives the coordinates of a point.
`sets`	A vector indicating which set each point belongs to.
`groups`	Indicates that the EAF must be computed separately for data belonging to different groups.
`percentiles`	The percentiles of the EAF of each side that will be plotted as attainment surfaces. `NA` does not plot any. See `eafplot.default`.

Value

A data frame (`data.frame`) containing the exact representation of EAF. The last column gives the percentile that corresponds to each point. If groups is not `NULL`, then an additional column indicates to which group the point belongs.

Note

There are several examples of data sets in `file.path(system.file(package="eaf"),"extdata")`.

Author(s)

Manuel López-Ibáñez

See Also

`read.data.sets`

Examples

```
eaf.path <- system.file(package="eaf")

x <- read.data.sets(file.path(eaf.path, "extdata","example1_dat"))
# Compute full EAF
eafs(x[,1:2], x[,3])

# Compute only best, median and worst
eafs(x[,1:2], x[,3], percentiles = c(0, 50, 100))

x <- read.data.sets(file.path(eaf.path,"extdata","spherical-250-10-3d.txt"))
y <- read.data.sets(file.path(eaf.path,"extdata","uniform-250-10-3d.txt"))
x <- data.frame(x, group = "spherical")
x <- rbind(x, data.frame(y, group = "uniform"))

# Compute only median separately for each group
eafs(x[,1:3], sets = x[,4], groups = x[,5], percentiles = 50)
```

---

gcp2x2 *Metaheuristics for solving the Graph Vertex Coloring Problem*

Description

Two metaheuristic algorithms, TabuCol (Hertz et al., 1987) and simulated annealing (Johnson et al., 1991), to find a good approximation of the chromatic number of two random graphs. The data here has the only goal of providing an example of use of eafplot for comparing algorithm performance with respect to both time and quality when modelled as two objectives in trade off.

Usage

```
gcp2x2
```

Format

A data frame with 3133 observations on the following 6 variables.

`alg` a factor with levels `SAKempeFI` and `TSinN1`

`inst` a factor with levels `DSJC500.5` and `DSJC500.9`. Instances are taken from the DIMACS repository.

`run` a numeric vector indicating the run to which the observation belong.

`best` a numeric vector indicating the best solution in number of colors found in the corresponding run up to that time.

`time` a numeric vector indicating the time since the beginning of the run for each observation. A rescaling is applied.

`titer` a numeric vector indicating iteration number corresponding to the observations.

Details

Each algorithm was run 10 times per graph registering the time and iteration number at which a new best solution was found. A time limit corresponding to 500*10^5 total iterations of TabuCol was imposed. The time was then normalized on a scale from 0 to 1 to make it instance independent.

Source

M. Chiarandini (2005). Stochastic local search methods for highly constrained combinatorial optimisation problems. Ph.D. thesis, Computer Science Department, Darmstadt University of Technology, Darmstadt, Germany. page 138.

References

A. Hertz and D. de Werra. Using Tabu Search Techniques for Graph Coloring. Computing, 1987, 39(4), 345-351.

D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning. Operations Research, 1991, 39(3), 378-406

Examples

```
data(gcp2x2)
```

HybridGA *Results of Hybrid GA on vanzyl and Richmond water networks*

Description

The data has the only goal of providing an example of use of eafplot.

Usage

```
HybridGA
```

Format

A list with two data frames, each of them with three columns, as produced by `read.data.sets`.

`$vanzyl` data frame of results on vanzyl network

`$richmond` data frame of results on Richmond network. The second column is filled with `NA`

Source

Manuel López-Ibáñez. Operational Optimisation of Water Distribution Networks. PhD thesis, School of Engineering and the Built Environment, Edinburgh Napier University, UK, 2009.

Examples

```
data(HybridGA)
print(HybridGA$vanzyl)
print(HybridGA$richmond)
```

read.data.sets *Read several data.frame sets*

Description

Reads a text file in table format and creates a data frame from it. The file may contain several sets, separated by empty lines. The function adds an additional column `set` to indicate to which set each row belongs.

Usage

```
read.data.sets(file, col.names)
```

Arguments

file	Filename that contains the data. Each row of the table appears as one line of the file. If it does not contain an *absolute* path, the file name is *relative* to the current working directory, getwd(). Tilde-expansion is performed where supported.
col.names	Vector of optional names for the variables. The default is to use '"V"' followed by the column number.

Value

A data frame (data.frame) containing a representation of the data in the file. An extra column set is added to indicate to which set each row belongs.

Warning

A known limitation is that the input file must use newline characters native to the host system, otherwise they will be, possibly silently, misinterpreted. In GNU/Linux the program dos2unix may be used to fix newline characters.

Note

There are several examples of data sets in file.path(system.file(package="eaf"),"extdata").

Author(s)

Manuel López-Ibáñez

See Also

read.table, eafplot, eafdiffplot

Examples

```
A1<-read.data.sets(file.path(system.file(package="eaf"),"extdata","ALG_1_dat"))
str(A1)
A2<-read.data.sets(file.path(system.file(package="eaf"),"extdata","ALG_2_dat"))
str(A2)
```

SPEA2minstoptimeRichmond

Results of SPEA2 when minimising electrical cost and maximising the minimum idle time of pumps on Richmond water network.

Description

The data has the only goal of providing an example of use of eafplot.

Usage

```
SPEA2minstoptimeRichmond
```

Format

A data frame as produced by read.data.sets. The second column measures time in seconds and corresponds to a maximisation problem.

Source

Manuel López-Ibáñez. Operational Optimisation of Water Distribution Networks. PhD thesis, School of Engineering and the Built Environment, Edinburgh Napier University, UK, 2009.

Examples

```
data(HybridGA)
data(SPEA2minstoptimeRichmond)
SPEA2minstoptimeRichmond[,2] <- SPEA2minstoptimeRichmond[,2] / 60
eafplot (SPEA2minstoptimeRichmond, xlab = expression(C[E]),
        ylab = "Minimum idle time (minutes)",
        las = 1, log = "y", maximise = c(FALSE, TRUE))
```

SPEA2relativeRichmond *Results of SPEA2 with relative time-controlled triggers on Richmond water network.*

Description

The data has the only goal of providing an example of use of eafplot.

Usage

```
SPEA2relativeRichmond
```

Format

A data frame as produced by read.data.sets.

Source

Manuel López-Ibáñez. Operational Optimisation of Water Distribution Networks. PhD thesis, School of Engineering and the Built Environment, Edinburgh Napier University, UK, 2009.

Examples

```
data(HybridGA)
data(SPEA2relativeRichmond)
eafplot (SPEA2relativeRichmond, percentiles = c(25, 50, 75),
        xlab = expression(C[E]), ylab = "Total switches",
        xlim = c(90, 140), ylim = c(0, 25),
        extra.points = HybridGA$richmond, extra.lty = "dashed",
        extra.legend = "Hybrid GA")
```

`SPEA2relativeVanzyl` *Results of SPEA2 with relative time-controlled triggers on Vanzyl's water network.*

Description

The data has the only goal of providing an example of use of eafplot.

Usage

```
SPEA2relativeVanzyl
```

Format

A data frame as produced by `read.data.sets`.

Source

Manuel López-Ibáñez. Operational Optimisation of Water Distribution Networks. PhD thesis, School of Engineering and the Built Environment, Edinburgh Napier University, UK, 2009.

Examples

```
data(HybridGA)
data(SPEA2relativeVanzyl)
eafplot(SPEA2relativeVanzyl, percentiles = c(25, 50, 75),
       xlab = expression(C[E]), ylab = "Total switches", xlim = c(320, 400),
       extra.points = HybridGA$vanzyl, extra.legend = "Hybrid GA")
```

Index