Package ‘FuzzyLP’

August 17, 2017

Title Fuzzy Linear Programming

Type Package

Description Provides methods to solve Fuzzy Linear Programming Problems with fuzzy constraints (following different approaches proposed by Verdegay, Zimmermann, Werners and Tanaka), fuzzy costs, and fuzzy technological matrix.

Version 0.1-5

Date 2017-08-17

License GPL (>= 3)

ByteCompile TRUE

Imports ROI, FuzzyNumbers, ROI.plugin.glpk

URL http://decsai.ugr.es/~pjvi

Suggests R.rsp

VignetteBuilder R.rsp

NeedsCompilation no

RoxygenNote 6.0.1

Author Carlos A. Rabelo [aut, cre],
Pablo J. Villacorta [ctb]

Maintainer Carlos A. Rabelo <carabelo@gmail.com>

Repository CRAN

Date/Publication 2017-08-17 17:09:33 UTC

R topics documented:

`crispLP` *Solves a crisp linear programming problem.*

Description

`crispLP` use the classic solver (simplex) to solve a crisp linear programming problem:

$$Max\ f(x)\ or\ Min\ f(x)$$

$$s.t.: \quad Ax <= b$$

Usage

```
crispLP(objective, A, dir, b, maximum = TRUE, verbose = TRUE)
```

Arguments

`objective`	A vector $(c_1, c_2, \ldots, c_n)$ with the objective function coefficients $f(x) = c_1 x_1 + \ldots + c_n x_n$.
`A`	Technological matrix of Real Numbers.
`dir`	Vector of strings with the direction of the inequalities, of the same length as `b`. Each element of the vector must be one of "=", ">=", "<=", "<" or ">".
`b`	Vector with the right hand side of the constraints.
`maximum`	`TRUE` to maximize the objective function, `FALSE` to minimize the objective function.
`verbose`	`TRUE` to show aditional screen info, `FALSE` to hide aditional screen info.

Value

`crispLP` returns the solution if the solver has found it or NULL if not.

Examples

```
## maximize:   3*x1 + x2
## s.t.:       1.875*x1   - 1.5*x2 <= 4
##              4.75*x1 + 2.125*x2 <= 14.5
##               x1, x2 are non-negative real numbers

obj <- c(3, 1)
A <- matrix(c(1.875, 4.75, -1.5, 2.125), nrow = 2)
dir <- c("<=", "<=")
b <- c(4, 14.5)
max <- TRUE

crispLP(obj, A, dir, b, maximum = max, verbose = TRUE)
```

`FCLP.classicObjective` *Solves a Fuzzy Linear Programming problem with fuzzy constraints trying to assure a minimum (maximum) value of the objective function.*

Description

The goal is to solve a linear programming problem having fuzzy constraints trying to assure a minimum (or maximum) value of the objective function.

$$Max\, f(x) \text{ or } Min\, f(x)$$

$$s.t.: \quad Ax <= b + (1 - \beta) * t$$

Where t means we allow not to satisfy the constraint, exceeding the bound b at most in t.

`FCLP.classicObjective` solves the problem trying to assure a minimum (maximum) value z_0 of the objective function ($f(x) >= z_0$ in maximization problems, $f(x) <= z_0$ in minimization problems).

`FCLP.fuzzyObjective` solves the problem trying to assure a minimum (maximum) value z_0 of the objective function with tolerance t_0 ($f(x) >= z_0 - (1 - \beta)t_0$ in maximization problems, $f(x) <= z_0 + (1 - \beta)t_0$ in minimization problems).

`FCLP.fuzzyUndefinedObjective` solves the problem trying to assure a minimum (maximum) value of the objective function with tolerance but the user doesn't fix the bound nor the tolerance. The function estimate a bound and a tolerance and call `FCLP.fuzzyObjective` using them.

`FCLP.fuzzyUndefinedNormObjective` solves the problem trying to assure a minimum (maximum) value of the objective function with tolerance but the user doesn't fix the bound nor the tolerance. The function normalize the objective, estimate a bound and a tolerance and call `FCLP.fuzzyObjective` using them.

Usage

```
FCLP.classicObjective(objective, A, dir, b, t, z0 = 0, maximum = TRUE,
  verbose = TRUE)

FCLP.fuzzyObjective(objective, A, dir, b, t, z0 = 0, t0 = 0,
  maximum = TRUE, verbose = TRUE)

FCLP.fuzzyUndefinedObjective(objective, A, dir, b, t, maximum = TRUE,
  verbose = TRUE)

FCLP.fuzzyUndefinedNormObjective(objective, A, dir, b, t, maximum = TRUE,
  verbose = TRUE)
```

Arguments

`objective`	A vector $(c_1, c_2, \ldots, c_n)$ with the objective function coefficients $f(x) = c_1x_1 + \ldots + c_nx_n$.

`A`	Technological matrix of Real Numbers.
`dir`	Vector of strings with the direction of the inequalities, of the same length as `b` and `t`. Each element of the vector must be one of "=", ">=", "<=", "<" or ">".
`b`	Vector with the right hand side of the constraints.
`t`	Vector with the tolerance of each constraint.
`z0`	The minimum (maximum in a minimization problem) value of the objective function to reach. Only used in `FCLP.classicObjective` and `FCLP.fuzzyObjective`.
`maximum`	`TRUE` to maximize the objective function, `FALSE` to minimize the objective function.
`verbose`	`TRUE` to show aditional screen info, `FALSE` to hide aditional screen info.
`t0`	The tolerance value to the minimum (or maximum) bound for the objective function. Only used in `FCLP.fuzzyObjective`.

Value

`FCLP.classicObjective` returns a solution reaching the given minimum (maximum) value of the objective function if the solver has found it (trying to maximize β) or NULL if not. Note that the found solution may not be the optimum for the β returned, trying β in `FCLP.fixedBeta` may obtain better results.

`FCLP.fuzzyObjective` returns a solution reaching the given minimum (maximum) value of the objective function if the solver has found it (trying to maximize β) or NULL if not. Note that the found solution may not be the optimum for the β returned, trying β in `FCLP.fixedBeta` may obtain better results.

`FCLP.fuzzyUndefinedObjective` returns a solution reaching the estimated minimum (maximum) value of the objective function if the solver has found it (trying to maximize β) or NULL if not.

`FCLP.fuzzyUndefinedNormObjective` returns a solution reaching the estimated minimum (maximum) value of the objective function if the solver has found it (trying to maximize β) or NULL if not.

References

Zimmermann, H. Description and optimization of fuzzy systems. International Journal of General Systems, 2:209-215, 1976.

Werners, B. An interactive fuzzy programming system. Fuzzy Sets and Systems, 23:131-147, 1987.

Tanaka, H. and Okuda, T. and Asai, K. On fuzzy mathematical programming. Journal of Cybernetics, 3,4:37-46, 1974.

See Also

`FCLP.fixedBeta`, `FCLP.sampledBeta`

Examples

```
## maximize:   3*x1 + x2 >= z0
## s.t.:       1.875*x1   - 1.5*x2 <= 4 + (1-beta)*5
##              4.75*x1 + 2.125*x2 <= 14.5 + (1-beta)*6
##               x1, x2 are non-negative real numbers

obj <- c(3, 1)
A <- matrix(c(1.875, 4.75, -1.5, 2.125), nrow = 2)
dir <- c("<=", "<=")
b <- c(4, 14.5)
t <- c(5, 6)
max <- TRUE

# Problem with solution
FCLP.classicObjective(obj, A, dir, b, t, z0=11, maximum=max, verbose = TRUE)

# This problem has a bound impossible to reach
FCLP.classicObjective(obj, A, dir, b, t, z0=14, maximum=max, verbose = TRUE)


# This problem has a fuzzy bound impossible to reach
FCLP.fuzzyObjective(obj, A, dir, b, t, z0=14, t0=1, maximum=max, verbose = TRUE)

# This problem has a fuzzy bound reachable
FCLP.fuzzyObjective(obj, A, dir, b, t, z0=14, t0=2, maximum=max, verbose = TRUE)


# We want the function estimates a bound and a tolerance
FCLP.fuzzyUndefinedObjective(obj, A, dir, b, t, maximum=max, verbose = TRUE)


# We want the function estimates a bound and a tolerance
FCLP.fuzzyUndefinedNormObjective(obj, A, dir, b, t, maximum=max, verbose = TRUE)
```

FCLP.fixedBeta — *Solves a Fuzzy Linear Programming problem with fuzzy constraints.*

Description

The goal is to solve a linear programming problem having fuzzy constraints.

$$Max\ f(x)\ or\ Min\ f(x)$$

$$s.t.:\quad Ax <= b + (1-\beta) * t$$

Where t means we allow not to satisfy the constraint, exceeding the bound b at most in t.

`FCLP.fixedBeta` uses the classic solver (simplex) to solve the problem with a fixed value of β.

`FCLP.sampledBeta` solves the problem in the same way than `FCLP.fixedBeta` but using several $\beta's$ taking values in a sample of the $[0, 1]$ inteval.

Usage

```
FCLP.fixedBeta(objective, A, dir, b, t, beta = 0.5, maximum = TRUE,
  verbose = TRUE)

FCLP.sampledBeta(objective, A, dir, b, t, min = 0, max = 1, step = 0.25,
  maximum = TRUE, verbose = TRUE)
```

Arguments

`objective`	A vector $(c_1, c_2, \ldots, c_n)$ with the objective function coefficients $f(x) = c_1x_1 + \ldots + c_nx_n$.
`A`	Technological matrix of Real Numbers.
`dir`	Vector of strings with the direction of the inequalities, of the same length as `b` and `t`. Each element of the vector must be one of "=", ">=", "<=", "<" or ">".
`b`	Vector with the right hand side of the constraints.
`t`	Vector with the tolerance of each constraint.
`beta`	The value of β to be used.
`maximum`	`TRUE` to maximize the objective function, `FALSE` to minimize the objective function.
`verbose`	`TRUE` to show aditional screen info, `FALSE` to hide aditional screen info.
`min`	The lower bound of the interval to take the sample.
`max`	The upper bound of the interval to take the sample.
`step`	The sampling step.

Value

`FCLP.fixedBeta` returns the solution for the given beta if the solver has found it or NULL if not.

`FCLP.sampledBeta` returns the solutions for the sampled $\beta's$ if the solver has found them. If the solver hasn't found solutions for any of the $\beta's$ sampled, return NULL.

References

Verdegay, J.L. Fuzzy mathematical programming. In: Fuzzy Information and Decision Processes, pages 231-237, 1982. M.M. Gupta and E.Sanchez (eds).

Delgado, M. and Herrera, F. and Verdegay, J.L. and Vila, M.A. Post-optimality analisys on the membership function of a fuzzy linear programming problem. Fuzzy Sets and Systems, 53:289-297, 1993.

See Also

`FCLP.classicObjective`, `FCLP.fuzzyObjective`

`FCLP.fuzzyUndefinedObjective`, `FCLP.fuzzyUndefinedNormObjective`

Examples

```
## maximize:   3*x1 + x2
## s.t.:       1.875*x1   - 1.5*x2 <= 4 + (1-beta)*5
##              4.75*x1 + 2.125*x2 <= 14.5 + (1-beta)*6
##               x1, x2 are non-negative real numbers

obj <- c(3, 1)
A <- matrix(c(1.875, 4.75, -1.5, 2.125), nrow = 2)
dir <- c("<=", "<=")
b <- c(4, 14.5)
t <- c(5, 6)
valbeta <- 0.5
max <- TRUE

FCLP.fixedBeta(obj, A, dir, b, t, beta=valbeta, maximum = max, verbose = TRUE)

FCLP.sampledBeta(obj, A, dir, b, t, min=0, max=1, step=0.25, maximum = max, verbose = TRUE)
```

`FOLP.multiObj` *Solves a fuzzy objective linear programming problem using Representation Theorem.*

Description

The goal is to solve a linear programming problem having Trapezoidal Fuzzy Numbers as coefficients in the objective function ($f(x) = c_1^f x_1 + \ldots + c_n^f x_n$).

$$Max\, f(x)\ or\ Min\, f(x)$$

$$s.t.: \quad Ax <= b$$

`FOLP.multiObj` uses a multiobjective approach. This approach is based on each β-cut of a Trapezoidal Fuzzy Number is an interval (different for each β). So the problem may be considered as a Parametric Linear Problem. For a value of β fixed, the problem becomes a Multiobjective Linear Problem, this problem may be solved from different approachs, `FOLP.multiObj` solves it using weights, the same weight for each objective.

`FOLP.interv` uses an intervalar approach. This approach is based on each β-cut of a Trapezoidal Fuzzy Number is an interval (different for each β). Fixing an β, using interval arithmetic and defining an order relation for intervals is posible to compare intervals, this transforms the problem in a biobjective problem (involving the minimum and the center of intervals). Finally `FOLP.interv` use weights to solve the biobjective problem.

`FOLP.strat` uses a stratified approach. This approach is based on that β-cuts are a sequence of nested intervals. Fixing an β two auxiliary problems are solved, the first replacing the fuzzy coefficients by the lower limits of the β-cuts, the second doing the same with the upper limits. The results of the two auxiliary problems allows to formulate a new auxiliary problem, this problem tries to maximize a parameter λ.

Usage

```
FOLP.multiObj(objective, A, dir, b, maximum = TRUE, min = 0, max = 1,
  step = 0.25)

FOLP.interv(objective, A, dir, b, maximum = TRUE, w1 = 0.5, min = 0,
  max = 1, step = 0.25)

FOLP.strat(objective, A, dir, b, maximum = TRUE, min = 0, max = 1,
  step = 0.25)
```

Arguments

objective	A vector $(c_1^f, c_2^f, ..., c_n^f)$ of Trapezoidal Fuzzy Numbers with the objective function coefficients $f(x) = c_1^f x_1 + \ldots + c_n^f x_n$. Note that any of the coefficients may also be Real Numbers.
A	Technological matrix of Real Numbers.
dir	Vector of strings with the direction of the inequalities, of the same length as b. Each element of the vector must be one of "=", ">=", "<=", "<" or ">".
b	Vector with the right hand side of the constraints.
maximum	TRUE to maximize the objective function, FALSE to minimize the objective function.
min	The lower bound of the interval to take the sample.
max	The upper bound of the interval to take the sample.
step	The sampling step.
w1	Weight to be used, w2 is calculated as w2=1-w1. w1 must be in the interval [0,1]. Only used in FOLP.interv.

Value

FOLP.multiObj returns the solutions for the sampled $\beta's$ if the solver has found them. If the solver hasn't found solutions for any of the $\beta's$ sampled, return NULL.

FOLP.interv returns the solutions for the sampled $\beta's$ if the solver has found them. If the solver hasn't found solutions for any of the $\beta's$ sampled, return NULL.

FOLP.strat returns the solutions and the value of λ for the sampled $\beta's$ if the solver has found them. If the solver hasn't found solutions for any of the $\beta's$ sampled, return NULL. A greater value of λ may be interpreted as the obtained solution is better.

References

Verdegay, J.L. Fuzzy mathematical programming. In: Fuzzy Information and Decision Processes, pages 231-237, 1982. M.M. Gupta and E.Sanchez (eds).

Delgado, M. and Verdegay, J.L. and Vila, M.A. Imprecise costs in mathematical programming problems. Control and Cybernetics, 16 (2):113-121, 1987.

Bitran, G.. Linear multiple objective problems with interval coefficients. Management Science, 26(7):694-706, 1985.

Alefeld, G. and Herzberger, J. Introduction to interval computation. 1984.

Moore, R. Method and applications of interval analysis, volume 2. SIAM, 1979.

Rommelfanger, H. and Hanuscheck, R. and Wolf, J. Linear programming with fuzzy objectives. Fuzzy Sets and Systems, 29:31-48, 1989.

See Also

FOLP.ordFun, FOLP.posib

Examples

```
## maximize:   [0,2,3]*x1 + [1,3,4,5]*x2
## s.t.:         x1 + 3*x2 <= 6
##               x1 +   x2 <= 4
##               x1, x2 are non-negative real numbers

obj <- c(FuzzyNumbers::TrapezoidalFuzzyNumber(0,2,2,3),
         FuzzyNumbers::TrapezoidalFuzzyNumber(1,3,4,5))
A<-matrix(c(1, 1, 3, 1), nrow = 2)
dir <- c("<=", "<=")
b <- c(6, 4)
max <- TRUE

# Using a Multiobjective approach.
FOLP.multiObj(obj, A, dir, b, maximum = max, min=0, max=1, step=0.2)


# Using a Intervalar approach.
FOLP.interv(obj, A, dir, b, maximum = max, w1=0.3, min=0, max=1, step=0.2)


# Using a Stratified approach.
FOLP.strat(obj, A, dir, b, maximum = max, min=0, max=1, step=0.2)
```

FOLP.ordFun — *Solves a fuzzy objective linear programming problem using ordering functions.*

Description

The goal is to solve a linear programming problem having Trapezoidal Fuzzy Numbers as coefficients in the objective function ($f(x) = c_1^f x_1 + \ldots + c_n^f x_n$).

$$Max\, f(x)\ or\ Min\, f(x)$$

$$s.t.: \quad Ax <= b$$

`FOLP.ordFun` uses ordering functions to compare Fuzzy Numbers.

Usage

```
FOLP.ordFun(objective, A, dir, b, maximum = TRUE, ordf = c("Yager1",
  "Yager3", "Adamo", "Average", "Custom"), ..., FUN = NULL)
```

Arguments

- `objective` A vector $(c_1^f, c_2^f, ..., c_n^f)$ of Trapezoidal Fuzzy Numbers with the objective function coefficients $f(x) = c_1^f x_1 + \ldots + c_n^f x_n$. Note that any of the coefficients may also be Real Numbers.
- `A` Technological matrix of Real Numbers.
- `dir` Vector of strings with the direction of the inequalities, of the same length as `b`. Each element of the vector must be one of "=", ">=", "<=", "<" or ">".
- `b` Vector with the right hand side of the constraints.
- `maximum` `TRUE` to maximize the objective function, `FALSE` to minimize the objective function.
- `ordf` Ordering function to be used, ordf must be one of "Yager1", "Yager3", "Adamo", "Average" or "Custom". The "Custom" option allows to use a custom linear ordering function that must be placed as FUN argument. If a non linear function is used the solution may not be optimal.
- `...` Additional parameters to the ordering function if needed.
  - Yager1 doesn't need any parameters.
  - Yager3 doesn't need any parameters.
  - Adamo needs a `alpha` parameter which must be in the interval `[0,1]`.
  - Average needs two parameters, `lambda` must be in the interval `[0,1]` and `t` that must be greater than `0`.
  - If Custom function needs parameters, put them here. Although not required, it is recommended to name the parameters.
- `FUN` Custom linear ordering function to be used if the value of ordf is "Custom". If any of the coefficients of the objective function are Real Numbers, the user must assure that the function `FUN` works well not only with Trapezoidal Fuzzy Numbers but also with Real Numbers.

Value

`FOLP.ordFun` returns the solution if the solver has found it or NULL if not.

References

Gonzalez, A. A studing of the ranking function approach through mean values. Fuzzy Sets and Systems, 35:29-41, 1990.

Cadenas, J.M. and Verdegay, J.L. Using Fuzzy Numbers in Linear Programming. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 27, No. 6, December 1997.

Tanaka, H., Ichihashi, H. and Asai, F. A formulation of fuzzy linear programming problems based a comparison of fuzzy numbers. Control and Cybernetics, 13:185-194, 1984.

See Also

FOLP.multiObj, FOLP.interv, FOLP.strat, FOLP.posib

Examples

```
## maximize:   [0,2,3]*x1 + [1,3,4,5]*x2
## s.t.:         x1 + 3*x2 <= 6
##               x1 +   x2 <= 4
##               x1, x2 are non-negative real numbers

obj <- c(FuzzyNumbers::TrapezoidalFuzzyNumber(0,2,2,3),
        FuzzyNumbers::TrapezoidalFuzzyNumber(1,3,4,5))
A<-matrix(c(1, 1, 3, 1), nrow = 2)
dir <- c("<=", "<=")
b <- c(6, 4)
max <- TRUE

FOLP.ordFun(obj, A, dir, b, maximum = max, ordf="Yager1")
FOLP.ordFun(obj, A, dir, b, maximum = max, ordf="Yager3")
FOLP.ordFun(obj, A, dir, b, maximum = max, ordf="Adamo", 0.5)
FOLP.ordFun(obj, A, dir, b, maximum = max, ordf="Average", lambda=0.8, t=3)

# Define a custom linear function
av <- function(tfn) {mean(FuzzyNumbers::core(tfn))}
FOLP.ordFun(obj, A, dir, b, maximum = max, ordf="Custom", FUN=av)

# Define a custom linear function
avp <- function(tfn, a) {a*mean(FuzzyNumbers::core(tfn))}
FOLP.ordFun(obj, A, dir, b, maximum = max, ordf="Custom", FUN=avp, a=2)
```

FOLP.posib — *Solves a fuzzy objective linear programming problem using Representation Theorem.*

Description

The goal is to solve a linear programming problem having Trapezoidal Fuzzy Numbers as coefficients in the objective function ($f(x) = c_1^f x_1 + \ldots + c_n^f x_n$).

$$Max\, f(x)\ or\ Min\, f(x)$$

$$s.t.: \quad Ax <= b$$

FOLP.posib uses a possibilistic approach. This approach is based on Trapezoidal Fuzzy Numbers arithmetic, so the whole objective may be considered as a Fuzzy Number itself. Defining a notion of maximum for this kind of numbers (a weighted average of the minimum and maximum of the support of the Trapezoidal number).

Usage

```
FOLP.posib(objective, A, dir, b, maximum = TRUE, w1 = 0.5)
```

Arguments

`objective`	A vector $(c_1^f, c_2^f, ..., c_n^f)$ of Trapezoidal Fuzzy Numbers with the objective function coefficients $f(x) = c_1^f x_1 + \ldots + c_n^f x_n$. Note that any of the coefficients may also be Real Numbers.
`A`	Technological matrix of Real Numbers.
`dir`	Vector of strings with the direction of the inequalities, of the same length as `b`. Each element of the vector must be one of "=", ">=", "<=", "<" or ">".
`b`	Vector with the right hand side of the constraints.
`maximum`	`TRUE` to maximize the objective function, `FALSE` to minimize the objective function.
`w1`	Weight to be used, `w2` is calculated as `w2=1-w1`. `w1` must be in the interval `[0,1]`.

Value

`FOLP.posib` returns the solution for the given weights if the solver has found it or NULL if not.

References

Dubois, D. and Prade, H. Operations in fuzzy numbers. International Journal of Systems Science, 9:613-626, 1978.

See Also

`FOLP.ordFun`, `FOLP.multiObj`, `FOLP.interv`, `FOLP.strat`

Examples

```
## maximize:   [0,2,3]*x1 + [1,3,4,5]*x2
## s.t.:         x1 + 3*x2 <= 6
##               x1 +   x2 <= 4
##               x1, x2 are non-negative real numbers

obj <- c(FuzzyNumbers::TrapezoidalFuzzyNumber(0,2,2,3),
         FuzzyNumbers::TrapezoidalFuzzyNumber(1,3,4,5))
A<-matrix(c(1, 1, 3, 1), nrow = 2)
dir <- c("<=", "<=")
b <- c(6, 4)
max <- TRUE

FOLP.posib(obj, A, dir, b, maximum = max, w1=0.2)
```

FuzzyLP *Fuzzy Linear Programming*

Description

FuzzyLP is a package to solve fuzzy linear programming problems.

Details

FuzzyLP implements several algorithm for solving fuzzy linear programming, the algorithms are fundamentally based in the book of...

References

Cadenas, J.M. and Verdegay, J.L. PROBO: an interactive system in fuzzy linear programming. Fuzzy Sets and Systems 76(3):319-332, 1995.

Cadenas, J.M. and Verdegay, J.L. Modelos de Optimizacion con Datos Imprecisos. Servicio de Publicaciones, Universidad de Murcia, 1999.

Bellman, R. and Zadeh, L. Decision making in a fuzzy environment. Management Science 17 (B) 4, pages 141-164, 1970.

GFLP *Solves a maximization (minimization) problem having fuzzy coefficients in the constraints, the objective function and/or the technological matrix.*

Description

The goal is to solve a linear programming problem having Trapezoidal Fuzzy Numbers as coefficients in the constraints, the objective function and/or the technological matrix.

$$Max\, f(x) \text{ or } Min\, f(x)$$

$$s.t.: \quad \widetilde{A}x <= \widetilde{b} + (1-\beta) * \widetilde{t}$$

This function uses different ordering functions for the objective function and for the constraints.

Usage

```
GFLP(objective, A, dir, b, t, maximum = TRUE, ordf_obj = c("Yager1",
  "Yager3", "Adamo", "Average"), ordf_obj_param = NULL,
  ordf_res = c("Yager1", "Yager3", "Adamo", "Average"),
  ordf_res_param = NULL, min = 0, max = 1, step = 0.25)
```

Arguments

- `objective` A vector $(c_1^f, c_2^f, ..., c_n^f)$ of Trapezoidal Fuzzy Numbers with the objective function coefficients $f(x) = c_1^f x_1 + \ldots + c_n^f x_n$. Note that any of the coefficients may also be Real Numbers.
- `A` Technological matrix containing Trapezoidal Fuzzy Numbers and/or Real Numbers.
- `dir` Vector of strings with the direction of the inequalities, of the same length as `b` and `t`. Each element of the vector must be one of "=", ">=", "<=", "<" or ">".
- `b` Vector with the right hand side of the constraints. `b` may contain Trapezoidal Fuzzy Numbers and/or Real Numbers.
- `t` Vector with the tolerance of each constraint. `t` may contain Trapezoidal Fuzzy Numbers and/or Real Numbers.
- `maximum` `TRUE` to maximize the objective function, `FALSE` to minimize the objective function.
- `ordf_obj` Ordering function to be used in the objective function, `ordf_obj` must be one of `"Yager1"`, `"Yager3"`, `"Adamo"` or `"Average"`.
- `ordf_obj_param` Parameters need by ordf_obj function, if it needs more than one parameter, use a named vector. See `FOLP.ordFun` for more information about the ordering functions parameters.
- `ordf_res` Ordering function to be used in the constraints, `ordf_res` must be one of `"Yager1"`, `"Yager3"`, `"Adamo"` or `"Average"`.
- `ordf_res_param` Parameters need by ordf_res function, if it needs more than one parameter, use a named vector. See `FOLP.ordFun` for more information about the ordering functions parameters.
- `min` The lower bound of the interval to take the sample.
- `max` The upper bound of the interval to take the sample.
- `step` The sampling step.

Value

`GFLP` returns the solutions for the sampled $\beta's$ if the solver has found them. If the solver hasn't found solutions for any of the $\beta's$ sampled, return NULL.

References

Gonzalez, A. A studing of the ranking function approach through mean values. Fuzzy Sets and Systems, 35:29-41, 1990.

Cadenas, J.M. and Verdegay, J.L. Using Fuzzy Numbers in Linear Programming. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 27, No. 6, December 1997.

Tanaka, H., Ichihashi, H. and Asai, F. A formulation of fuzzy linear programming problems based a comparison of fuzzy numbers. Control and Cybernetics, 13:185-194, 1984.

Examples

```
## maximize:   [1,3,4,5]*x1 + x2
## s.t.:         [0,2,3,3.5]*x1 +   [0,1,1,4]*x2 <= [2,2,2,3] + (1-beta)*[1,2,2,3]
##               [3,5,5,6]*x1   + [1.5,2,2,3]*x2 <= 12
##               x1, x2 are non-negative real numbers

obj <- c(FuzzyNumbers::TrapezoidalFuzzyNumber(1,3,4,5), 1)

a11 <- FuzzyNumbers::TrapezoidalFuzzyNumber(0,2,2,3.5)
a21 <- FuzzyNumbers::TrapezoidalFuzzyNumber(3,5,5,6)
a12 <- -FuzzyNumbers::TrapezoidalFuzzyNumber(0,1,1,4)
a22 <- FuzzyNumbers::TrapezoidalFuzzyNumber(1.5,2,2,3)
A <- matrix(c(a11, a21, a12, a22), nrow = 2)

dir <- c("<=", "<=")
b<-c(FuzzyNumbers::TrapezoidalFuzzyNumber(2,2,2,3), 12)
t<-c(FuzzyNumbers::TrapezoidalFuzzyNumber(1,2,2,3),0);
max <- TRUE

GFLP(obj, A, dir, b, t, maximum = max, ordf_obj="Yager1", ordf_res="Yager3")
GFLP(obj, A, dir, b, t, maximum = max, ordf_obj="Adamo", ordf_obj_param=0.5, ordf_res="Yager3")
GFLP(obj, A, dir, b, t, maximum = max, "Average", ordf_obj_param=c(t=3, lambda=0.5),
ordf_res="Adamo", ordf_res_param = 0.5)
GFLP(obj, A, dir, b, t, maximum = max, ordf_obj="Average", ordf_obj_param=c(t=3, lambda=0.8),
ordf_res="Yager3", min = 0, max = 1, step = 0.2)
```

Index