Package 'ProjectManagement'

September 14, 2023

Type Package

Title Management of Deterministic and Stochastic Projects

Date 2023-09-14

Version 1.5.2

Maintainer Juan Carlos Gonçalves Dosantos <juan.carlos.goncalves@udc.es>

Description Management problems of deterministic and stochastic projects. It obtains the duration of a project and the appropriate slack for each activity in a deterministic context. In addition it obtains a schedule of activities' time (Castro, Gómez & Tejada (2007) <doi:10.1016/j.orl.2007.01.003>). It also allows the management of resources. When the project is done, and the actual duration for each activity is known, then it can know how long the project is delayed and make a fair delivery of the delay between each activity (Bergantiños, Valencia-Toledo & Vidal-Puga (2018) <doi:10.1016/j.dam.2017.08.012>). In a stochastic context it can estimate the average duration of the project and plot the density of this duration, as well as, the density of the early and last times of the chosen activities. As in the deterministic case, it can make a distribution of the delay generated by observing the project already carried out.

Depends R (>= 4.3.0), plotly

Imports lpSolveAPI, triangle, kappalab, igraph

License GPL (>= 2)

Encoding UTF-8

Author Juan Carlos Gonçalves Dosantos [aut, cre],
Ignacio García Jurado [aut],
Julián Costa Bouzas [aut]

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-09-14 12:10:08 UTC

R topics documented:

ProjectManagement-package . . . 2
dag.plot . . . 3
delay.pert . . . 4
delay.stochastic.pert . . . 6
early.time . . . 8
last.time . . . 9
levelling.resources . . . 10
mce . . . 11
organize . . . 12
rebuild . . . 13
resource.allocation . . . 14
schedule.pert . . . 15
stochastic.pert . . . 16

Index **19**

`ProjectManagement-package`
Management of Deterministic and Stochastic Projects

Description

Management of Deterministic and Stochastic Projects

Details

Management problems of deterministic and stochastic projects. It obtains the duration of a project and the appropriate slack for each activity in a deterministic context. In addition it obtains a schedule of activities' time (Castro, Gómez & Tejada (2007) <doi:10.1016/j.orl.2007.01.003>). It also allows the management of resources. When the project is done, and the actual duration for each activity is known, then it can know how long the project is delayed and make a fair delivery of the delay between each activity (Bergantiños, Valencia-Toledo & Vidal-Puga (2018) <doi:10.1016/j.dam.2017.08.012>). In a stochastic context it can estimate the average duration of the project and plot the density of this duration, as well as, the density of the early and last times of the chosen activities. As in the deterministic case, it can make a distribution of the delay generated by observing the project already carried out.

dag.plot *DAG plot*

Description

This function plots a directed acyclic graph (DAG).

Usage

```
dag.plot(
  prec1and2 = matrix(0),
  prec3and4 = matrix(0),
  critical.activities = NULL
)
```

Arguments

prec1and2	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
prec3and4	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
critical.activities	A vector indicating the critical activities to represent them in a different color (Default=NULL) .

Value

A plot.

Examples

```
prec1and2<-matrix(c(0,1,0,2,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,2,0),nrow=5,ncol=5,byrow=TRUE)
prec3and4<-matrix(0,nrow=5,ncol=5)
prec3and4[3,1]<-3
dag.plot(prec1and2,prec3and4)
```

`delay.pert` *Problems of distribution of delay in deterministic projects*

Description

This function calculates the delay of a project once it has been completed. In addition, it also calculates the distribution of the delay between the different activities with the proportional, truncated proportional and Shapley rule.

Usage

```
delay.pert(
  duration,
  prec1and2 = matrix(0),
  prec3and4 = matrix(0),
  observed.duration,
  delta = NULL,
  cost.function = NULL
)
```

Arguments

`duration`	Vector with the expected duration for each activity.
`prec1and2`	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`prec3and4`	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`observed.duration`	Vector with the observed duration for each activity.
`delta`	Value to indicate the maximun time that the project can take without delay. If this is not added, the function will use as delta the expected project time. This value is only used if the function uses the default cost function.
`cost.function`	Delay costs function. If this value is not added, a default cost function will be used.

Details

Given a problem of sharing delays in a project $(N, \prec, \{\bar{X}_i\}_{i\in N}, \{x_i\}_{i\in N})$, such that $\{\bar{X}_i\}_{i\in N}$ is the expected value of activities' duration and $\{x_i\}_{i\in N}$ the observed value. If $D(N, \prec, \{\bar{X}_i\}_{i\in N})$ is the expected project time and $D(N, \prec, \{x_i\}_{i\in N})$ is the observed project time, it has to $d = D(N, \prec , \{\bar{X}_i\}_{i\in N}) - \delta$ is the delay, where δ can be any arbitrary value greater than zero. The following rules distribute the delay costs among the different activities.

The proportional rule, from *Brânzei et al. (2002)*, distributes the delay, d, proportionally. So that each activity receives a payment of:

$$\phi_i = \frac{x_i - \bar{X}_i}{\sum_{j \in N} \max\{x_j - \bar{X}_j, 0\}} \cdot C(D(N, \prec, \{\bar{X}_i\}_{i \in N})).$$

The truncated proportional rule, from *Brânzei et al. (2002)*, distributes the delay, d, proportionally, where the individual delay of each player is reduced to d if if is larger. So that each activity receives a payment of:

$$\bar{\phi}_i = \frac{\min\{x_i - \bar{X}_i, C(D(N, \prec, \{\bar{X}_i\}_{i \in N}))\}}{\sum_{j \in N} \max\{\min\{x_j - \bar{X}_j, C(D(N, \prec, \{\bar{X}_i\}_{i \in N}))\}, 0\}} \cdot C(D(N, \prec, \{\bar{X}_i\}_{i \in N})).$$

These values are only well defined when the sum of the individual delays is different from zero.

Shapley rule distributes the delay, d, based on the Shapley value for TU games, see *Bergantiños et al. (2018)*. Given a project problem with delays $(N, \prec, \{\bar{X}_i\}_{i \in N}, \{x_i\}_{i \in N})$, its associated TU game, (N, v), is $v(S) = C(D(N, \prec, (\{\bar{X}_i\}_{i \in N \setminus S}, \{x_i\}_{i \in S})))$ for all $S \subseteq N$, where C is the costs function (by default $C(D(N, \prec, y)) = D(N, \prec, y) - \delta$. If the number of activities is greater than ten, the Shapley value, of the game (N, v), is estimated using a unique sampling process for all players, see *Castro et al. (2009)*.

Value

The delay value and a solution matrix.

References

berg Bergantiños, G., Valencia-Toledo, A., & Vidal-Puga, J. (2018). Hart and Mas-Colell consistency in PERT problems. Discrete Applied Mathematics, 243, 11-20.

bran Brânzei, R., Ferrari, G., Fragnelli, V., & Tijs, S. (2002). Two approaches to the problem of sharing delay costs in joint projects. Annals of Operations Research, 109(1-4), 359-374.

castro Castro, J., Gómez, D., & Tejada, J. (2009). Polynomial calculation of the Shapley value based on sampling. Computers & Operations Research, 36(5), 1726-1730.

Examples

```
prec1and2<-matrix(c(0,1,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0),nrow=5,ncol=5,byrow=TRUE)
duration<-c(2,1,1,4,2)
observed.duration<-c(2.5,1.25,2,4.5,3)
delta<-6
delay.pert(duration,prec1and2=prec1and2,observed.duration=observed.duration,
delta=delta,cost.function=NULL)
```

`delay.stochastic.pert` *Problems of distribution of delay in stochastic projects*

Description

This function calculates the delay of a stochastic project, once it has been carried out. In addition, it also calculates the distribution of the delay on the different activities with the Stochastic Shapley rule.

Usage

```
delay.stochastic.pert(
  prec1and2 = matrix(0),
  prec3and4 = matrix(0),
  distribution,
  values,
  observed.duration,
  percentile = NULL,
  delta = NULL,
  cost.function = NULL,
  compilations = 1000
)
```

Arguments

`prec1and2`	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`prec3and4`	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`distribution`	Type of distribution that each activities' duration has. It can be NORMAL, TRIANGLE, EXPONENTIAL, UNIFORM, T-STUDENT, FDISTRIBUTION, CHI-SQUARED, GAMMA, WEIBULL, BINOMIAL, POISSON, GEOMETRIC, HYPERGEOMETRIC and EMPIRICAL.
`values`	Matrix with the parameters corresponding to the distribution associated with the duration for each activity. Considering i as an activity we have the following cases. If the distribution is TRIANGLE, then (i, 1) it is the minimum value, (i, 2) the maximum value and (i, 3) the mode. If the distribution is NORMAL, (i, 1) is the mean and (i, 2) the variance. If the distribution is EXPONENTIAL, then (i, 1) is the λ parameter. If the distribution is UNIFORM, (i, 1) it is the minimum value and (i, 2) the maximum value. If the distribution is T-STUDENT, (i, 1) degrees of freedom and (i, 2) non-centrality parameter delta. In FDISTRIBUTION, (i, 1) and (i, 2) degrees of freedom and (i, 3) non-centrality parameter. In

CHI-SQUARED, (i, 1) degrees of freedom and (i, 2) non-centrality parameter (non-negative). In GAMMA, (i, 1) and (i, 3) shape and scale parameters and (i, 2) an alternative way to specify the scale. In WEIBULL, (i, 1) and (i, 2) shape and scale parameters. In BINOMIAL, (i, 1) number of trials (zero or more) and (i, 2) probability of success on each trial. In POISSON, (i, 1) non-negative mean. In GEOMETRIC, (i, 1) probability of success in each trial, between 0 and 1. In HYPERGEOMETRIC, (i, 1) number of white balls in the urn, (i, 2) number of black balls in the urn and (i, 3) numer of balls drawn from the urn. Finally, if the distribution is EMPIRICAL, then (i,j), for all $j \in \{1, ..., M\}$ such that $M > 0$, is the sample.

`observed.duration`
: Vector with the observed duration for each activity.

`percentile`
: Percentile used to calculate the maximum time allowed for the duration of the project (Default=NULL). Only percentile or delta is necessary. This value is only used if the function uses the default cost function.

`delta`
: Maximum time allowed for the duration of the project (Default=NULL). Only delta or pencetile is necessary. This value is only used if the function uses the default cost function.

`cost.function`
: Delay costs function. If this value is not added, a default cost function will be used.

`compilations`
: Number of compilations that the function will use for average calculations (Default=1000).

Details

Given a problem of sharing delays in a stochastic project $(N, \prec, \{X_i\}_{i\in N}, \{x_i\}_{i\in N})$, such that $\{X_i\}_{i\in N}$ is the random variable of activities' durations and $\{x_i\}_{i\in N}$ the observed value. It is defined as $E(D(N, \prec, \{X_i\}_{i\in N}))$ the expected project time, where E is the mathematical expectation, and $D(N, \prec, \{x_i\}_{i\in N})$ the observed project time, then $d = D(N, \prec, \{X_i\}_{i\in N}) - \delta$, with $\delta > 0$, normally $\delta > E(D(N, \prec, \{X_i\}_{i\in N}))$, is the delay. The proportional and truncated proportional rule, see delay.pert function, can be adapted to this context by using the mean of the random variables.

The Stochastic Shapley, *Gonçalves-Dosantos et al. (2020)*, rule is based on the Shapley value for the TU game (N, v) where $v(S) = E(C(D(N, \prec, (\{X_i\}_{i\in N\setminus S}, \{x_i\}_{i\in S})))$, for all $S \subseteq N$, where C is the costs function (by default $C(y) = D(N, \prec, y) - \delta$). If the number of activities is greater than ten, the Shapley value, of the game (N, v), is estimated using a unique sampling process for all players, see *Castro et al. (2009)*.

The Stochastic Shapley rule 2 is based on the sum of the Shapley values for the TU games (N, v) and (N, w) where $v(S) = E(C(D(N, \prec, (\{X_i\}_{i\in N\setminus S}, \{x_i\}_{i\in S})))) - E(C(D(N, \prec, (\{X_i\}_{i\in N}))))$ and $w(S) = E(C(D(N, \prec, (\{0_i\}_{i\in N\setminus S}, \{X_i\}_{i\in S}))))$, for all $S \subseteq N$, 0_N denotes the vector in R^N whose components are equal to zero and where C is the costs function (by default $C(y) = D(N, \prec , y) - \delta$).

Value

A delay value and solution vector.

References

castro Castro, J., Gómez, D., & Tejada, J. (2009). Polynomial calculation of the Shapley value based on sampling. Computers & Operations Research, 36(5), 1726-1730.

gon Gonçalves-Dosantos, J.C., García-Jurado, I., Costa, J. (2020) Sharing delay costs in Stochastic scheduling problems with delays. 4OR, 18(4), 457-476

Examples

```
prec1and2<-matrix(c(0,1,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0),nrow=5,ncol=5,byrow=TRUE)
distribution<-c("TRIANGLE","TRIANGLE","TRIANGLE","TRIANGLE","EXPONENTIAL")
values<-matrix(c(1,3,2,1/2,3/2,1,1/4,9/4,1/2,3,5,4,1/2,0,0),nrow=5,byrow=TRUE)
observed.duration<-c(2.5,1.25,2,4.5,3)
percentile<-NULL
delta<-6.5
delay.stochastic.pert(prec1and2=prec1and2,distribution=distribution,values=values,
observed.duration=observed.duration,percentile=percentile,delta=delta,
cost.function=NULL,compilations=1000)
```

---

early.time *Early time for a deterministic projects*

Description

This function calculates the early time for one project.

Usage

```
early.time(prec1and2 = matrix(0), prec3and4 = matrix(0), duration)
```

Arguments

prec1and2	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
prec3and4	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
duration	vector with the duración for each activities.

Value

Early time vector.

References

burke Burke, R. (2013). Project management: planning and control techniques. New Jersey, USA.

Examples

```
prec1and2<-matrix(c(0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0),nrow=5,ncol=5,byrow=TRUE)
duration<-c(3,2,1,1.5,4.2)
early.time(prec1and2,duration=duration)
```

`last.time` *Last time for a deterministic projects*

Description

This function calculates the last time for one project.

Usage

```
last.time(prec1and2 = matrix(0), prec3and4 = matrix(0), duration, early.times)
```

Arguments

`prec1and2`	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`prec3and4`	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`duration`	Vector with the duración for each activity.
`early.times`	Vector with the early times for each activities.

Value

Last time vector.

References

bur Burke, R. (2013). Project management: planning and control techniques. New Jersey, USA.

Examples

```
prec1and2<-matrix(c(0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0),nrow=5,ncol=5,byrow=TRUE)
duration<-c(3,2,1,1.5,4.2)
early.times<-c(0,0,3.5,2,0)
last.time(prec1and2,duration=duration,early.times=early.times)
```

`levelling.resources` *Project resource levelling*

Description

This function calculates the schedule of the project so that the consumption of resources is as uniform as possible.

Usage

```
levelling.resources(
  duration,
  prec1and2 = matrix(0),
  prec3and4 = matrix(0),
  resources,
  int = 1
)
```

Arguments

`duration`	Vector with the duration for each activity.
`prec1and2`	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`prec3and4`	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`resources`	Vector indicating the necessary resources for each activity per period of time.
`int`	Numerical value indicating the duration of each period of time (Default=1).

Details

The problem of leveling resources takes into account that in order for activities to be carried out in the estimated time, a certain level of resources must be used. The problem is to find a schedule that allows to execute the project in the estimated time so that the temporary consumption of resources is as level as possible.

Value

A solution matrices.

References

heg Hegazy, T. (1999). Optimization of resource allocation and leveling using genetic algorithms. Journal of construction engineering and management, 125(3), 167-175.

Examples

```
duration<-c(3,4,2,1)
resources<-c(4,1,3,3)
prec1and2<-matrix(c(0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0),nrow=4,ncol=4,byrow=TRUE)

levelling.resources(duration,prec1and2,prec3and4=matrix(0),resources,int=1)
```

---

mce *Build a precedence matrix*

Description

This function calculates the costs per activity to accelerate the project.

Usage

```
mce(
  duration,
  minimum.durations,
  prec1and2 = matrix(0),
  prec3and4 = matrix(0),
  activities.costs,
  duration.project = NULL
)
```

Arguments

duration	Vector with the duration for each activity.
minimum.durations	Vector with the Minimum duration allowed for each activity.
prec1and2	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
prec3and4	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
activities.costs	Vector indicating the cost of accelerating a unit of time the duration of each activity.
duration.project	numerical value indicating the minimum time sought in the project (Default=NULL).

Details

The MCE method (Minimal Cost Expediting) tries to speed up the project at minimum cost. It considers that the duration of some project activities could be reduced by increasing the resources allocated to them (and thus increasing their implementation costs).

Value

A solution matrices.

References

kelley Kelley Jr, J. E. (1961). Critical-path planning and scheduling: Mathematical basis. Operations research, 9(3), 296-320.

Examples

```
duration<-c(5,4,5,2,2)
minimum.durations<-c(3,2,3,1,1)
activities.costs<-c(1,1,1,1,1)
prec1and2<-matrix(c(0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0),nrow=5,ncol=5,byrow=TRUE)
duration.project<-6

mce(duration,minimum.durations,prec1and2,prec3and4=matrix(0),activities.costs,duration.project)
```

`organize` *Organize project activities*

Description

This function organizes the activities of a project, in such a way that if i precedes j then i is less strict than j.

Usage

```
organize(prec1and2 = matrix(0), prec3and4 = matrix(0))
```

Arguments

`prec1and2`	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`prec3and4`	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.

Value

A list containing:

- Precedence: ordered precedence matrix.
- Order: new activities values.

Examples

```
prec1and2<-matrix(c(0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0),nrow=5,ncol=5,byrow=TRUE)
organize(prec1and2)
```

`rebuild` *Build a precedence matrix*

Description

This function builds a unique type 1 precedence matrix given any kind of precedence.

Usage

```
rebuild(prec1and2 = matrix(0), prec3and4 = matrix(0))
```

Arguments

`prec1and2`	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`prec3and4`	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.

Details

There are four types of precedence between two activities i, j: Type 1: the activity j cannot start until activity i has finished. Type 2: the activity j cannot start until activity i has started. Type 3: the activity j cannot end until activity i has ended. Type 4: the activity j cannot end until activity i has started.

All these precedences can be written only as type 1. It should be noted that precedence type 1 implies type 2, and type 2 implies type 4. On the other hand, precedence type 1 implies type 3, and type 3 implies type 4.

Value

A list containing:

- Precedence: precedence matrix.
- Type 2: activities related to type 2 precedence.
- Type 3: activities related to type 3 precedence.
- Type 4: activities related to type 4 precedence.

Examples

```
prec1and2<-matrix(c(0,1,0,2,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,2,0),nrow=5,ncol=5,byrow=TRUE)
prec3and4<-matrix(0,nrow=5,ncol=5)
prec3and4[3,1]<-3
rebuild(prec1and2,prec3and4)
```

---

resource.allocation *Project resource allocation*

Description

This function calculates the project schedule so that resource consumption does not exceed the maximum available per time period..

Usage

```
resource.allocation(
  duration,
  prec1and2,
  prec3and4 = matrix(0),
  resources,
  max.resources,
  int = 1
)
```

Arguments

duration	Vector with the duration for each activity.
prec1and2	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
prec3and4	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.

resources	Vector indicating the necessary resources for each activity per period of time.
max.resources	Numerical value indicating the maximum number of resources that can be used in each period.
int	Numerical value indicating the duration of each period of time (Default=1).

Details

The problem of resource allocation takes into account that in order for activities to be carried out in the estimated time, a certain level of resources must be used. The problem is that the level of resources available in each period is limited. The aim is to obtain the minimum time and a schedule for the execution of the project taking into account this new restriction.

Value

A solution matrices.

References

hega Hegazy, T. (1999). Optimization of resource allocation and leveling using genetic algorithms. Journal of construction engineering and management, 125(3), 167-175.

Examples

```
duration<-c(3,4,2,1)
prec1and2<-matrix(c(0,1,0,0,0,0,0,0,0,0,1,0,0,0,0),nrow=4,ncol=4,byrow=TRUE)
resources<-c(4,1,3,3)
max.resources<-4

resource.allocation(duration,prec1and2,prec3and4=matrix(0),resources,max.resources,int=1)
```

---

schedule.pert *Schedule for deterministic projects*

Description

This function calculates the duration of the project, the slacks for each activity, as well as the schedule of each activity.

Usage

```
schedule.pert(
  duration,
  prec1and2 = matrix(0),
  prec3and4 = matrix(0),
  PRINT = TRUE
)
```

Arguments

duration	Vector with the duration for each activity.
prec1and2	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
prec3and4	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
PRINT	Logical indicator to show the schedule represented in a graph (Default=TRUE)

Value

A list of a project schedule and if PRINT=TRUE a plot of schedule.

References

burk Burke, R. (2013). Project management: planning and control techniques. New Jersey, USA.

Examples

```
prec1and2<-matrix(c(0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0),nrow=5,ncol=5,byrow=TRUE)
duration<-c(3,2,1,1.5,4.2)
schedule.pert(duration,prec1and2)
```

---

stochastic.pert *Stochastic projects*

Description

This function calculates the average duration time for a stochastic project and the activities criticality index. It also plots the estimate density of the project duration, as well as the estimate density of the early and last times.

Usage

```
stochastic.pert(
  prec1and2 = matrix(0),
  prec3and4 = matrix(0),
  distribution,
  values,
  percentile = 0.95,
  plot.activities.times = NULL,
  compilations = 1000
)
```

Arguments

`prec1and2`	A matrix indicating the order of precedence type 1 and 2 between the activities (Default=matrix(0)). If value $(i, j) = 1$ then activity i precedes type 1 to j, and if $(i, j) = 2$ then activity i precedes type 2 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`prec3and4`	A matrix indicating the order of precedence type 3 and 4 between the activities (Default=matrix(0)). If value $(i, j) = 3$ then activity i precedes type 3 to j, and if $(i, j) = 4$ then activity i precedes type 4 to j. Cycles cannot exist in a project, i.e. if an activity i precedes j then j cannot precede i.
`distribution`	Type of distribution that each activities' duration has. It can be NORMAL, TRIANGLE, EXPONENTIAL, UNIFORM, T-STUDENT, FDISTRIBUTION, CHI-SQUARED, GAMMA, WEIBULL, BINOMIAL, POISSON, GEOMETRIC, HYPERGEOMETRIC and EMPIRICAL.
`values`	Matrix with the parameters corresponding to the distribution associated with the duration for each activity. Considering i as an activity we have the following cases. If the distribution is TRIANGLE, then (i, 1) it is the minimum value, (i, 2) the maximum value and (i, 3) the mode. If the distribution is NORMAL, (i, 1) is the mean and (i, 2) the variance. If the distribution is EXPONENTIAL, then (i, 1) is the λ parameter. If the distribution is UNIFORM, (i, 1) it is the minimum value and (i, 2) the maximum value. If the distribution is T-STUDENT, (i, 1) degrees of freedom and (i, 2) non-centrality parameter delta. In FDISTRIBUTION, (i, 1) and (i, 2) degrees of freedom and (i, 3) non-centrality parameter. In CHI-SQUARED, (i, 1) degrees of freedom and (i, 2) non-centrality parameter (non-negative). In GAMMA, (i, 1) and (i, 3) shape and scale parameters and (i, 2) an alternative way to specify the scale. In WEIBULL, (i, 1) and (i, 2) shape and scale parameters. In BINOMIAL, (i, 1) number of trials (zero or more) and (i, 2) probability of success on each trial. In POISSON, (i, 1) non-negative mean. In GEOMETRIC, (i, 1) probability of success in each trial, between 0 and 1. In HYPERGEOMETRIC, (i, 1) number of white balls in the urn, (i, 2) number of black balls in the urn and (i, 3) numer of balls drawn from the urn. Finally, if the distribution is EMPIRICAL, then (i,j), for all $j \in \{1, ..., M\}$ such that $M > 0$, is the sample.
`percentile`	Percentile used to calculate the maximum time allowed for the duration of the project (Default=0.95).
`plot.activities.times`	Vector of selected activities to show the distribution of their early and last times (Default=NULL).
`compilations`	Number of compilations that the function will use for average calculations (Default=1000).

Value

Two values, average duration time and the maximum time allowed, a critically index vector and a durations histogram.

Examples

```
prec1and2<-matrix(c(0,1,0,1,0,0,0,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0),nrow=5,ncol=5,byrow=TRUE)
distribution<-c("TRIANGLE","TRIANGLE","TRIANGLE","TRIANGLE","EXPONENTIAL")
values<-matrix(c(1,3,2,1/2,3/2,1,1/4,9/4,1/2,3,5,4,1/2,0,0),nrow=5,byrow=TRUE)
percentile<-0.95
plot.activities.times<-c(1,4)
stochastic.pert(prec1and2=prec1and2,distribution=distribution,values=values,
percentile=percentile,plot.activities.times=plot.activities.times)
```

Index

dag.plot, 3
delay.pert, 4
delay.stochastic.pert, 6

early.time, 8

last.time, 9
levelling.resources, 10

mce, 11

organize, 12

ProjectManagement
        *(*ProjectManagement-package*)*, 2
ProjectManagement-package, 2

rebuild, 13
resource.allocation, 14

schedule.pert, 15
stochastic.pert, 16