# Package ‘cooptrees’

February 19, 2015

**Type** Package

**Version** 1.0

**Date** 2014-09-01

**Title** Cooperative aspects of optimal trees in weighted graphs

**Author** Manuel Fontenla

**Maintainer** Manuel Fontenla <manu.fontenla@gmail.com>

**Depends** R (>= 2.7.0), igraph (>= 0.7.1), optrees (>= 1.0)

**Imports** gtools

**Description** Computes several cooperative games and allocation rules associated with minimum cost spanning tree problems and minimum cost arborescence problems.

**License** GPL-3

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2014-09-04 19:13:22

## R topics documented:

---

`cooptrees-package` *Cooperative aspects of optimal trees in weighted graphs*

---

### Description

Computes several cooperative games and allocation rules associated with minimum cost spanning tree problems and minimum cost arborescence problems.

### Details

| | |
|---|---|
| Package: | cooptrees |
| Type: | Package |
| Version: | 1.0 |
| Date: | 2014-09-01 |
| License: | GPL-3 |

The most important functions are mstCooperative and maCooperative. The other functions included in the package are auxiliary ones that can be used independently.

### Author(s)

Manuel Fontenla

Maintainer: Manuel Fontenla <manu.fontenla@gmail.com>

### References

B. Dutta and D. Mishra, "Minimum cost arborescences", Games and Economic Behavior, vol. 74, pp. 120-143, Jan. 2012.

C. G. Bird, "On Cost Allocation for a Spanning Tree: A Game Theoretic Approach", Networks, no. 6, pp. 335-350, 1976.

B. Dutta and A. Kar, "Cost monotonicity, consistency and minimum cost spanning tree games", Games and Economic Behavior, vol. 48, pp. 223-248, Aug. 2004.

A. Kar, "Axiomatization of the Shapley Value on Minimum Cost Spanning Tree Games", Games and Economic Behavior, vol. 38, pp. 265-277, Feb. 2002.

G. Bergantiños and J. J. Vidal-Puga, "A fair rule in minimum cost spanning tree problems", Journal of Economic Theory, vol. 137, pp. 326-352, Nov. 2007.

G. Bergantiños and J. J. Vidal-Puga, "The optimistic TU game in minimum cost spanning tree problems", International Journal of Game Theory, vol. 36, pp. 223-239, Feb. 2007.

V. Feltkamp, S. H. Tijs, S. Muto, "On the irreducible core and the equal remaining obligation rule of minimum cost extension problems", Mimeo, Tilburg University, 1994.

---

`maBird` *Bird rule for minimum cost arborescence problems*

---

## Description

Given a graph with a minimum cost arborescence, the `maBird` function divides the cost of this arborescence among the agents. For that purpose it, uses the Bird rule, where each agent pays the cost of the last arc that connects him to the source.

## Usage

```
maBird(nodes, arcs)
```

## Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

## Value

`maBird` returns a matrix with the agents and their costs.

## See Also

The more general function maRules.

## Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,7, 1,3,6, 1,4,4, 2,3,8, 2,4,6, 3,2,6,
                 3,4,5, 4,2,5, 4,3,7), ncol = 3, byrow = TRUE)
# Bird
maBird(nodes, arcs)
```

maCooperative *Cooperation in minimum cost arborescence problems*

## Description

Given a graph with at least one minimum cost arborescence, the `maCooperative` function computes the cooperative and "Bird" and "ERO" rules.

## Usage

```
maCooperative(nodes, arcs, show.data = TRUE)
```

## Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |
| `show.data` | logical value indicating if the function displays the console output (`TRUE`) or not (`FALSE`). By default its value is `TRUE`. |

## Value

`maCooperative` returns and prints a list with the cooperative games and allocation rules of a minimum cost arborescence problem.

## Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,2, 1,3,3, 1,4,4, 2,3,3, 2,4,4, 3,2,3,
                 3,4,1, 4,2,1, 4,3,2), ncol = 3, byrow = TRUE)
# Cooperation in minimum cost arborescence problems
maCooperative(nodes, arcs)
```

maERO *ERO rule for minimum cost arborescence problems*

## Description

Given a graph with a minimum cost arborescence, the `maERO` function divides the cost of the arborescence among the agents according to the ERO rule. For that purpose, the irreducible form of the problem is obtained. The ERO rule is just the Shapley value of the cooperative game associated with the irreducible form.

### Usage

```
maERO(nodes, arcs)
```

### Arguments

| | |
|---|---|
| nodes | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| arcs | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

### Value

maERO returns a matrix with the agents and their costs.

### See Also

The more general function maRules.

### Examples

```
# Graphs
nodes <- 1:4
arcs <- matrix(c(1,2,7, 1,3,6, 1,4,4, 2,3,8, 2,4,6, 3,2,6,
                 3,4,5, 4,2,5, 4,3,7), ncol = 3, byrow = TRUE)
# ERO
maERO(nodes, arcs)
```

---

## maGames — *Cooperative game associated with minimum cost arborescences*

---

### Description

Given a graph with at least one minimum cost arborescence the maGames function builds the cooperative game associated with it.

### Usage

```
maGames(nodes, arcs, game = "pessimistic", show.data = TRUE)
```

### Arguments

| | |
|---|---|
| nodes | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| arcs | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

| | |
|---|---|
| `game` | denotes the game to be obtained, known as the "pessimistic" game. |
| `show.data` | logical value indicating if the function displays the console output (`TRUE`) or not (`FALSE`). By default its value is `TRUE`. |

### Value

`maGames` returns a vector with the characteristic fuction of the cooperative game associated with the graph.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,7, 1,3,6, 1,4,4, 2,3,8, 2,4,6, 3,2,6,
                 3,4,5, 4,2,5, 4,3,7), ncol = 3, byrow = TRUE)
# Cooperative games
maGames(nodes, arcs, game = "pessimistic")
```

---

## `maIrreducible` *Irreducible form for a minimum cost arborescence problem*

---

### Description

Given a graph with at least one minimum cost arborescence the `maIrreducible` function obtains the irreducible form.

### Usage

```
maIrreducible(nodes, arcs)
```

### Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

### Value

`maIrreducible` returns a matrix with the list of arcs of the irreducible form.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,7, 1,3,6, 1,4,4, 2,3,8, 2,4,6, 3,2,6,
                 3,4,5, 4,2,5, 4,3,7), ncol = 3, byrow = TRUE)
# Irreducible form
maIrreducible(nodes, arcs)
```

## maPessimistic — *Pessimistic game associated with minimum cost arborescences*

### Description

Given a graph with at least one minimum cost arborescence, the `maPessimistic` function builds the pessimistic game.

### Usage

```
maPessimistic(nodes, arcs)
```

### Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

### Value

`maPessimistic` returns a vector with the characteristic function of the pessimitic game.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,7, 1,3,6, 1,4,4, 2,3,8, 2,4,6, 3,2,6,
                 3,4,5, 4,2,5, 4,3,7), ncol = 3, byrow = TRUE)
# Pessimistic game
maPessimistic(nodes, arcs)
```

## maRules — *Allocation rules for minimum cost arborescence problems*

### Description

Given a graph with at least one minimum cost arborescence, the `maRules` function divides the cost of the arborescence among the agents according to "Bird" and "ERO" rules.

### Usage

```
maRules(nodes, arcs, rule, show.data = TRUE)
```

### Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |
| `rule` | denotes the chosen allocation rule: "Bird" or "ERO". |
| `show.data` | logical value indicating if the function displays the console output (`TRUE`) or not (`FALSE`). By default its value is `TRUE`. |

### Value

`maRules` returns a matrix with the agents and their costs. It also prints the result in console.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,7, 1,3,6, 1,4,4, 2,3,8, 2,4,6, 3,2,6,
                 3,4,5, 4,2,5, 4,3,7), ncol = 3, byrow = TRUE)
# Allocation rules
maRules(nodes, arcs, rule = "Bird")
maRules(nodes, arcs, rule = "ERO")
```

---

## `mstBird` *Bird rule for minimum cost spanning tree problems*

---

### Description

Given a graph with at least one minimum cost spanning tree, the `mstBird` function divides the cost of the tree obtained with Prim's algorithm among the agents. For that purpose it uses the Bird rule, where each agent pays the cost of the arc that connects him to the tree source.

### Usage

```
mstBird(nodes, arcs)
```

### Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

## Value

`mstBird` returns a matrix with the agents and their costs.

## See Also

The more general function mstRules.

## Examples

```
# Graphs
nodes <- 1:4
arcs <- matrix(c(1,2,6, 1,3,10, 1,4,6, 2,3,4, 2,4,6, 3,4,4),
               byrow = TRUE, ncol = 3)
# Bird
mstBird(nodes, arcs)
```

---

`mstCooperative` *Cooperation in minimum cost spanning tree problems*

---

## Description

Given a graph with at least one minimum cost spanning tree, the `mstCooperative` function computes the pessimistic and optimistic games; and the most known allocation rules: "Bird", "Dutta-Kar","Kar" and "ERO".

## Usage

```
mstCooperative(nodes, arcs, show.data = TRUE)
```

## Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |
| `show.data` | logical value indicating if the function displays the console output (`TRUE`) or not (`FALSE`). By default its value is `TRUE`. |

## Value

`mstCooperative` returns and print a list with the cooperative games and the allocation rules of a minimum cost spanning tree problem.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,6, 1,3,10, 1,4,6, 2,3,4, 2,4,6, 3,4,4),
               byrow = TRUE, ncol = 3)
# Cooperation in minimum cost spanning tree problems
mstCooperative(nodes, arcs)
```

---

## mstDuttaKar    *Dutta-Kar rule for minimum cost spanning tree problems*

---

### Description

Given a graph with at least one minimum cost spanning tree, the `mstDuttaKar` function divides the cost of the tree obtained with Prim's algorithm among the agents according to the Dutta-Kar rule. This rule specifies that each agent chooses to pay the minimum cost between the last arc that connects him to the source and the cost that rejects his successor. The order is set by Prim's algorithm.

### Usage

```
mstDuttaKar(nodes, arcs)
```

### Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

### Value

`mstDuttaKar` returns a matrix with the agents and their costs.

### See Also

The more general function mstRules.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,6, 1,3,10, 1,4,6, 2,3,4, 2,4,6, 3,4,4),
               byrow = TRUE, ncol = 3)
# Dutta-Kar
mstDuttaKar(nodes, arcs)
```

`mstEROKruskal` *ERO rule for minimum cost spanning tree problems with Kruskal's algorithm*

### Description

Given a graph with at least one minimum cost spanning tree, the `mstEROKruskal` function divides the cost of the tree among the agents according to the ERO rule.

### Usage

```
mstEROKruskal(nodes, arcs)
```

### Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number from 1 until the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

### Value

`mstEROKruskal` returns a matrix with the agents and their costs.

### See Also

The more general function mstRules.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,6, 1,3,10, 1,4,6, 2,3,4, 2,4,6, 3,4,4),
               byrow = TRUE, ncol = 3)
# ERO with Kruskal
mstEROKruskal(nodes, arcs)
```

mstGames *Cooperative games from minimum cost spanning tree problems*

## Description

Given a graph with at least one minimum cost spanning tree, mstGames builds both cooperative games: the pessimistic and the optimistic game.

The pessimistic game associated with a minimum cost spanning tree problem is a cooperative game in which every coalition of agents obtains the minimum cost assuming that the agents outside the coalition are not connected.

The optimistic game associated with with a minimum cost spanning tree problem is a cooperative game in which every coalition of agents obtains the minimum cost assuming that that the agents outside the coalition are connected. Thus, the agents in the coalition can benefit from their connections to the source

## Usage

```
mstGames(nodes, arcs, game, show.data = TRUE)
```

## Arguments

| | |
|---|---|
| nodes | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| arcs | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |
| game | denotes the game that we want to obtain: "pessimistic" or "optimistic". |
| show.data | logical value indicating if the function displays the console output (TRUE) or not (FALSE). By default its value is TRUE. |

## Value

mstGames returns a vector with the characteristic fuction of the selected game associated with the graph and prints the result in console.

## Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,6, 1,3,10, 1,4,6, 2,3,4, 2,4,6, 3,4,4),
               byrow = TRUE, ncol = 3)
# Cooperative games
mstGames(nodes, arcs, game = "pessimistic")
mstGames(nodes, arcs, game = "optimistic")
```

`mstIrreducible` *Irreducible form for a minimum cost spanning tree problem*

### Description

Given a graph with at least one minimum cost spanning tree, the `mstIrreducible` function obtains the irreducible form.

### Usage

```
mstIrreducible(nodes, arcs)
```

### Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

### Value

`mstIrreducible` returns a matrix with the list of arcs of the irreducible form.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,6, 1,3,10, 1,4,6, 2,3,4, 2,4,6, 3,4,4),
               byrow = TRUE, ncol = 3)
# Irreducible form
mstIrreducible(nodes, arcs)
```

`mstKar` *Kar rule for minimum cost spanning tree problems*

### Description

Given a graph with at least one minimum cost spanning tree, the `mstKar` function divides the cost of the tree among the agents according to the Kar rule. That rule is obtained with the Shapley value of the pessimistic game.

### Usage

```
mstKar(nodes, arcs)
```

### Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

### Value

`mstKar` returns a matrix with the agents and their costs.

### See Also

The more general function mstRules.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,6, 1,3,10, 1,4,6, 2,3,4, 2,4,6, 3,4,4),
               byrow = TRUE, ncol = 3)
# Kar
mstKar(nodes, arcs)
```

---

## `mstOptimistic` *Optimistic game of a minimum cost spanning tree problem*

---

### Description

Given a graph with at least one minimum cost spanning tree, the `mstOptimistic` function builds the optimistic game associated with it.

### Usage

```
mstOptimistic(nodes, arcs)
```

### Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

### Value

`mstOptimistic` returns a vector with the characteristic function of the optimistic game.

### See Also

The more general function mstGames.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,6, 1,3,10, 1,4,6, 2,3,4, 2,4,6, 3,4,4),
               byrow = TRUE, ncol = 3)
# Optimistic game
mstOptimistic(nodes, arcs)
```

---

## mstPessimistic — *Pessimistic game from a minimum cost spanning tree problem*

---

### Description

Given a graph with at least one minimum cost spanning tree, the `mstPessimistic` function builds the pessimistic game.

### Usage

```
mstPessimistic(nodes, arcs)
```

### Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |

### Value

`mstPessimistic` returns a vector with the characteristic function of the pessimistic game.

### See Also

The more general function mstGames.

### Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,6, 1,3,10, 1,4,6, 2,3,4, 2,4,6, 3,4,4),
               byrow = TRUE, ncol = 3)
# Pessimistic game
mstPessimistic(nodes, arcs)
```

`mstRules` *Allocation rules for minimum cost spanning tree problem*

## Description

Given a graph with at least one minimum cost spanning tree, the `mstRules` function divides the cost of the tree among the agents according to the most known rules: "Bird", "Dutta-Kar", "Kar", "ERO".

## Usage

```
mstRules(nodes, arcs, rule, algorithm = "Kruskal", show.data = TRUE)
```

## Arguments

| | |
|---|---|
| `nodes` | vector containing the nodes of the graph, identified by a number that goes from 1 to the order of the graph. |
| `arcs` | matrix with the list of arcs of the graph. Each row represents one arc. The first two columns contain the two endpoints of each arc and the third column contains their weights. |
| `rule` | denotes the chosen allocation rule: "Bird", "Dutta-Kar", "Kar" or "ERO". |
| `algorithm` | denotes the algorithm used with the ERO rule: "Kruskal". |
| `show.data` | logical value indicating if the function displays the console output`TRUE` or no `FALSE`. The deafult is `TRUE`. |

## Value

`mstRules` returns a matrix with the agents and the cost that each one of them has to pay. It also prints the result in console.

## Examples

```
# Graph
nodes <- 1:4
arcs <- matrix(c(1,2,6, 1,3,10, 1,4,6, 2,3,4, 2,4,6, 3,4,4),
               byrow = TRUE, ncol = 3)
# Allocation Rules
mstRules(nodes, arcs, rule = "Bird")
mstRules(nodes, arcs, rule = "Dutta-Kar")
mstRules(nodes, arcs, rule = "Kar")
mstRules(nodes, arcs, rule = "ERO", algorithm = "Kruskal")
```

## shapleyValue — *Shapley value of a cooperative game*

### Description

Given a cooperative game, the `shapleyValue` function computes its Shapley value.

### Usage

```
shapleyValue(n, S = NULL, v)
```

### Arguments

| | |
|---|---|
| `n` | number of players in the cooperative game. |
| `S` | vector with all the possible coalitions. If none has been specified the function generates it automatically. |
| `v` | vector with the characteristic function of the cooperative game. |

### Details

The Shapley value is a solution concept in cooperative game theory proposed by Lloyd Shapley in 1953. It is obtained as the average of the marginal contributions of the players associated with all the posible orders of the players.

### Value

The `shapleyValue` functions returns a matrix with all the marginal contributions of the players (`contributions`) and a vector with the Shapley value (`value`).

### References

Lloyd S. Shapley. "A Value for n-person Games". In Contributions to the Theory of Games, volume II, by H.W. Kuhn and A.W. Tucker, editors. Annals of Mathematical Studies v. 28, pp. 307-317. Princeton University Press, 1953.

### Examples

```
# Cooperative game
n <- 3  # players
v <- c(4, 4, 4, 8, 8, 8, 14)  # characteristic function
# Shapley value
shapleyValue(n, v = v)
```

# Index