Package 'pomdp'

November 30, 2018

Title Solve & Visualize POMDP

Version 0.0.0.9000

Description The pomdp package enables the user to simply define all components of a POMDP model such as states, transitions, and rewards in various ways and then transforms all the inputs into a suitable and POMDP-friendly structure. The package then utilizes the 'pomdp-solver' to solve the user-defined problem using finite grid method and provides outputs such as optimal policy and solution. The package also visualizes the solutions.

Depends R (>= 3.1.0)

License GPL (>=3)

LazyData true

Imports igraph

R topics documented:

alpha *Return coefficients of a POMDP solution*

Description

The function returs all the coefficients of the optimal hyperplanes

Usage

```
alpha(x)
```

Arguments

x	object of class POMDP

Value

returns a dataframe that includes all the coefficients of the optimal hyperplanes

See Also

pomdp

Examples

```
# you need to use pomdp function to generate an object of class POMDP first.
# here for an example we use the Tiger example code to generate such an object.
discount <- 0.75
values <- "reward"
states <- c("tiger-left" , "tiger-right")
actions <- c("listen" , "open-left" , "open-right")
observations <- c("tiger-left" , "tiger-right")
start <- "uniform"
grid_size <- 10
transition_prob <- list("listen" = "identity" ,
                        "open-left" = "uniform" ,
                        "open-right" = "uniform")
observation_prob <- list("listen" = matrix(c(0.85 , 0.15 ,
                                             0.15 , 0.85) , nrow = 2 , byrow = TRUE) ,
                         "open-left" = "uniform" ,
                         "open-right" = "uniform")
reward <- data.frame("action" = c("listen" , "open-left" , "open-left" ,
                                  "open-right" , "open-right") ,
                     "start-state" = c("*" , "tiger-left" , "tiger-right" ,
                                       "tiger-left" , "tiger-right") ,
                     "end-state" = c("*" , "*" , "*" , "*" , "*") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(-1 , -100 , 10 , 10 , -100))
result <- pomdp(discount, states, actions, observations, start, transition_prob,
                observation_prob, reward, values = "reward", grid_size, verbose = FALSE)
```

```
# then using the alpha() function we print all the coefficients of the optimal solution of
# the given POMDP object

alpha(result)
```

---

belief *Return belief states of a POMDP*

Description

the function simply returns all the belief states (belief points) used in the algorithm to solve the POMDP

Usage

```
belief(x)
```

Arguments

x	object of class POMDP

Value

returns a dataframe of all the belief states (rows); there is a column at the end that indicates which line (vector) provides the best value for the given belief state

See Also

pomdp

Examples

```
# you need to use pomdp function to generate an object of class POMDP first.
# here for an example we use the Tiger example code to generate such an object.
discount <- 0.75
values <- "reward"
states <- c("tiger-left" , "tiger-right")
actions <- c("listen" , "open-left" , "open-right")
observations <- c("tiger-left" , "tiger-right")
start <- "uniform"
grid_size <- 10
transition_prob <- list("listen" = "identity" ,
                        "open-left" = "uniform" ,
                        "open-right" = "uniform")
observation_prob <- list("listen" = matrix(c(0.85 , 0.15 ,
                                             0.15 , 0.85) , nrow = 2 , byrow = TRUE) ,
                         "open-left" = "uniform" ,
                         "open-right" = "uniform")
reward <- data.frame("action" = c("listen" , "open-left" , "open-left" ,
                                  "open-right" , "open-right") ,
                     "start-state" = c("*" , "tiger-left" , "tiger-right" ,
                                       "tiger-left" , "tiger-right") ,
                     "end-state" = c("*" , "*" , "*" , "*" , "*") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(-1 , -100 , 10 , 10 , -100))
result <- pomdp(discount, states, actions, observations, start, transition_prob,
                observation_prob, reward, values = "reward", grid_size, verbose = FALSE)

# then using the belief() function we print all the belief states of the given POMDP object

belief(result)
```

belief.proportions *Rerurn belief proportions of a POMDP policy graph*

Description

The function simply returns the proportions (probabilities) of being in each actual state for each node of a policy graph

Usage

```
belief.proportions(x)
```

Arguments

x	object of class POMDP

Value

returns a dataframe that includes the probabilities of being in each actual state for each node of the policy graph (rows)

See Also

pomdp

Examples

```
# you need to use pomdp function to generate an object of class POMDP first.
# here for an example we use the Tiger example code to generate such an object.
discount <- 0.75
values <- "reward"
states <- c("tiger-left" , "tiger-right")
actions <- c("listen" , "open-left" , "open-right")
observations <- c("tiger-left" , "tiger-right")
start <- "uniform"
grid_size <- 10
transition_prob <- list("listen" = "identity" ,
                          "open-left" = "uniform" ,
                          "open-right" = "uniform")
observation_prob <- list("listen" = matrix(c(0.85 , 0.15 ,
                                             0.15 , 0.85) , nrow = 2 , byrow = TRUE) ,
                           "open-left" = "uniform" ,
                           "open-right" = "uniform")
reward <- data.frame("action" = c("listen" , "open-left" , "open-left" ,
                                    "open-right" , "open-right") ,
                     "start-state" = c("*" , "tiger-left" , "tiger-right" ,
                                         "tiger-left" , "tiger-right") ,
                     "end-state" = c("*" , "*" , "*" , "*" , "*") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(-1 , -100 , 10 , 10 , -100))
result <- pomdp(discount, states, actions, observations, start, transition_prob,
                observation_prob, reward, values = "reward", grid_size, verbose = FALSE)

# then using the belief.proportions() function we print all the belief proportions of the given
# POMDP object

belief.proportions(result)
```

`initial.node` *Return initial node of a POMDP policy graph*

Description

The function simply returns which node of a given POMDP policy graph is the initial node.

Usage

```
initial.node(x)
```

Arguments

`x`	object of class POMDP

Value

returns an integer number indicating the initial node of the policy graph

See Also

pomdp

Examples

```
# you need to use pomdp function to generate an object of class POMDP first.
# here for an example we use the Tiger example code to generate such an object.
discount <- 0.75
values <- "reward"
states <- c("tiger-left" , "tiger-right")
actions <- c("listen" , "open-left" , "open-right")
observations <- c("tiger-left" , "tiger-right")
start <- "uniform"
grid_size <- 10
transition_prob <- list("listen" = "identity" ,
                        "open-left" = "uniform" ,
                        "open-right" = "uniform")
observation_prob <- list("listen" = matrix(c(0.85 , 0.15 ,
                                             0.15 , 0.85) , nrow = 2 , byrow = TRUE) ,
                         "open-left" = "uniform" ,
                         "open-right" = "uniform")
reward <- data.frame("action" = c("listen" , "open-left" , "open-left" ,
                                  "open-right" , "open-right") ,
                     "start-state" = c("*" , "tiger-left" , "tiger-right" ,
                                       "tiger-left" , "tiger-right") ,
                     "end-state" = c("*" , "*" , "*" , "*" , "*") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(-1 , -100 , 10 , 10 , -100))
result <- pomdp(discount, states, actions, observations, start, transition_prob,
                observation_prob, reward, values = "reward", grid_size, verbose = FALSE)

# then using the initial.node() function we print the initial node number of the policy
# graph of the given POMDP object

initial.node(result)
```

`model` *Return user-defined model components of a POMDP*

Description

The function simply returns all the inputs provided by the user as the POMDP model components but in a POMDP-friendly structure

Usage

```
model(x)
```

Arguments

x	object of class POMDP

Value

returns a list of all the inputs defined by the user but transformed into POMDP-friendly structure

See Also

pomdp

Examples

```
# you need to use pomdp function to generate an object of class POMDP first.
# here for an example we use the Tiger example code to generate such an object.
discount <- 0.75
values <- "reward"
states <- c("tiger-left" , "tiger-right")
actions <- c("listen" , "open-left" , "open-right")
observations <- c("tiger-left" , "tiger-right")
start <- "uniform"
grid_size <- 10
transition_prob <- list("listen" = "identity" ,
                        "open-left" = "uniform" ,
                        "open-right" = "uniform")
observation_prob <- list("listen" = matrix(c(0.85 , 0.15 ,
                                             0.15 , 0.85) , nrow = 2 , byrow = TRUE) ,
                         "open-left" = "uniform" ,
                         "open-right" = "uniform")
reward <- data.frame("action" = c("listen" , "open-left" , "open-left" ,
                                  "open-right" , "open-right") ,
                     "start-state" = c("*" , "tiger-left" , "tiger-right" ,
                                       "tiger-left" , "tiger-right") ,
                     "end-state" = c("*" , "*" , "*" , "*" , "*") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(-1 , -100 , 10 , 10 , -100))
result <- pomdp(discount, states, actions, observations, start, transition_prob,
                observation_prob, reward, values = "reward", grid_size, verbose = FALSE)

# then using the model() function we print all the the above components we just defined
# in a single list

model(result)
```

---

pg   *Return policy graph of a POMDP solution*

Description

The function returns all the nodes and arcs of a policy graph of a POMDP solution in a table

Usage

```
pg(x)
```

Arguments

x	object of class POMDP

Value

returns a dataframe of all the nodes (rows) and arcs in the policy graph and how they are connected as well as the optimal action associated with each node in the graph

See Also

pomdp

Examples

```
# you need to use pomdp function to generate an object of class POMDP first.
# here for an example we use the Tiger example code to generate such an object.
discount <- 0.75
values <- "reward"
states <- c("tiger-left" , "tiger-right")
actions <- c("listen" , "open-left" , "open-right")
observations <- c("tiger-left" , "tiger-right")
start <- "uniform"
grid_size <- 10
transition_prob <- list("listen" = "identity" ,
                        "open-left" = "uniform" ,
                        "open-right" = "uniform")
observation_prob <- list("listen" = matrix(c(0.85 , 0.15 ,
                                             0.15 , 0.85) , nrow = 2 , byrow = TRUE) ,
                         "open-left" = "uniform" ,
                         "open-right" = "uniform")
reward <- data.frame("action" = c("listen" , "open-left" , "open-left" ,
                                  "open-right" , "open-right") ,
                     "start-state" = c("*" , "tiger-left" , "tiger-right" ,
                                       "tiger-left" , "tiger-right") ,
                     "end-state" = c("*" , "*" , "*" , "*" , "*") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(-1 , -100 , 10 , 10 , -100))
result <- pomdp(discount, states, actions, observations, start, transition_prob,
                observation_prob, reward, values = "reward", grid_size, verbose = FALSE)

# then using the pg() function we print the policy graph of the given POMDP object as a table

pg(result)
```

---

plot *Plot POMDP policy graph*

Description

The function plots the POMDP policy graph provided in an object of classs POMDP

Usage

```
## S3 method for class 'POMDP'
plot(x, y = NULL, states = NULL, plot = TRUE, ...)
```

Arguments

`x`	object of class POMDP
`y`	default is NULL
`states`	a vector of numbers indicating which states to be represented in the graph
`plot`	a logical that indicates the plot should be printed or not
`...`	other ploting options

Details

x should be an object of class POMDP. `pomdp` function returns such an object.

Value

the function returns a POMDP policy graph with legends for states and observation arcs

See Also

`pomdp`

Examples

```
# you need to use pomdp function to generate an object of class POMDP.
# here for an example we use the Tiger example code to generate the graph.
discount <- 0.75
values <- "reward"
states <- c("tiger-left" , "tiger-right")
actions <- c("listen" , "open-left" , "open-right")
observations <- c("tiger-left" , "tiger-right")
start <- "uniform"
grid_size <- 10
transition_prob <- list("listen" = "identity" ,
                        "open-left" = "uniform" ,
                        "open-right" = "uniform")
observation_prob <- list("listen" = matrix(c(0.85 , 0.15 ,
                                             0.15 , 0.85) , nrow = 2 , byrow = TRUE) ,
                         "open-left" = "uniform" ,
                         "open-right" = "uniform")
reward <- data.frame("action" = c("listen" , "open-left" , "open-left" ,
                                  "open-right" , "open-right") ,
                     "start-state" = c("*" , "tiger-left" , "tiger-right" ,
                                       "tiger-left" , "tiger-right") ,
                     "end-state" = c("*" , "*" , "*" , "*" , "*") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(-1 , -100 , 10 , 10 , -100))
result <- pomdp(discount, states, actions, observations, start, transition_prob,
                observation_prob, reward, values = "reward", grid_size, verbose = FALSE)
plot(result)
```

pomdp *Solve a POMDP*

Description

This function utilizes the 'pomdp-solve' program (written in C) to solve problems that are formulated as partially observable Markov decision processes, a.k.a. POMDPs [1]. The solver actually have the option to use various POMDP solution algorithms to solve problems but the pomdp function in this R package uses only the Finite Grid method [2] to do so. The function provides user with various ways of defining the pomdp components such as states, actions, transitions and rewards and then transforms all the inputs into pomdp-friendly structures and feeds them to 'pomdp-solver'. Given there is an optimal solution, the function provides the optimal solution including the optimal policy.

Usage

```
pomdp(discount, states, actions, observations, start, transition_prob,
      observation_prob, reward, values = "reward", grid_size, verbose = FALSE)
```

Arguments

`discount`	numeric, a real number in `[0,1]`
`states`	character, a vector of strings specifying the names of the states
`actions`	character, a vector of strings specifying the names of the actions
`observations`	character, a vector of strings specifying the names of the observations
`start`	indicates the initial probabilities of being in each state. options are: • a vector of *n* numbers each in `[0,1]`, that add up to `1`, where *n* is the number of states, or • character, a string with the word "uniform" (a uniform distribution over all states), or • numeric, an integer in 1 to *n* (to specify a single starting state), or • character, a string specifying the name of a single state (to specify a single starting state), or • character, a vector of strings, specifying a subset of states (a uniform distribution over a subset of states) (the first element of the vector should be `"-"` if the following subset of states is supposed to be excluded)
`transition_prob`	contains the transition probabilities between states. options are: • a dataframe with 4 columns, where the columns specify *action*, *start-state*, *end-state* and the *probability* respectively. The frist 3 columns could be either character (the name of the action or state) or numeric with an integer starting from 0 (the number associated with the action or state e.g., 0 as the first action or 2 as the third state)

- a list of *m* matrices where *m* is the number of actions, each matrix is square of size *n* where *n* is the number of states (each matrix should have a name in the list and the name should be one of the actions). Also each matrix can be defined using "identity" or "uniform"

`observation_prob`
: options are:
  - a dataframe with 4 columns, where the columns specify *action*, *end-state*, *observation* and the *probability* respectively. The frist 3 columns could be either character (the name of the action, state, or observation) or numeric with an integer starting from 0 (the number associated with the action, state or observation e.g., 0 as the first action or 2 as the third state) or they can be "*" to indicate the independency
  - a list of *m* matrices where *m* is the number of actions, each matrix is of size nxo where *n* is the number of states and *o* is the number of observations (each matrix should have a name in the list and the name should be one of the actions). Also each matrix can be defined using "uniform"

`reward`
: options are:
  - a dataframe with 5 columns, where the columns specify *action*, *start-state*, *end-state*, *observation* and the *reward* respectively. The frist 4 columns could be either character (the name of the action, state, or observation) or numeric with an integer starting from 0 (the number associated with the action, state, observation e.g., 0 as the first action or 2 as the third state) or they can be "*" to indicate the independency
  - numeric, a matrix of size nxo where *n* is the number of states and *o* is the number of observations.

`values`
: character, a string with either "reward" or "cost". The default is reward

`grid_size`
: numeric, an integer that specifies the size of the grid to solve the model with

`verbose`
: logical, if set to true, the function provides the output of the pomdp solver in R console

Value

The function returns a list containing the followings:

`belief`
: a dataframe of all the belief states (rows); there is a column at the end that indicates which line (vector) provides the best value for the given belief state. Use `belief` function to get this dataframe directly

`belief_proportions`
: a dataframe that includes the probabilities of being in each actual state for each node of the policy graph (rows). Use `belief.proportions` function to get this dataframe directly

`alpha`
: a dataframe that includes all the coefficients of the optimal hyperplanes. Use `alpha` function to get this dataframe directly

`pg`
: a dataframe of all the nodes (rows) and arcs in the policy graph and how they are connected as well as the optimal action associated with each node in the graph. Use `pg` function to get this dataframe directly

`total_expected_reward`	a numeric value indicating the total expected reward of the optimal solution. Use `total.expected.reward` function to get this value directly
`initial_node`	an integer number indicating the initial node of the policy graph. Use `initial.node` function to get this node number directly
`solver_output`	a character that contains the output of the 'pomdp-solve' including its iterations. Use `solver.output` function to get this output directly
`model`	a list of all the inputs defined by the user but transformed into POMDP-friendly structure. Use `model` function to get this list directly

Note

'pomdp-solve' program uses the basic dynamic programming approach, solving one stage at a time working backwards in time. It does finite horizon problems with or without discounting. It will stop solving if the answer is within a tolerable range of the infinite horizon answer, and there are a couple of different stopping conditions (requires a discount factor less than 1.0). Alternatively you can solve a finite horizon problem for some fixed horizon length.

Author(s)

Hossein Kamalzadeh, Michael Hahsler

References

[1] For further details on how the POMDP solver utilized in this R package works check the following website: www.pomdp.org

[2] Cassandra, A. Rocco, Exact and approximate algorithms for partially observable markov decision processes, (1998). https://dl.acm.org/citation.cfm?id=926710

Examples

```
# below are examples and different ways of how you can define the input arguments of the function.
# you can find complete and executable examples such as tiger problem at the end of this section.

### different ways of input definition #################################################
discount <- 0.9
states <- c("state1" , "state2" , "state3")
actions <- c("action1" , "action2")
observations <- c("obs1" , "obs2")

## for the start vector, based on the options you can do either of the followings:
start <- c(0.5 , 0.3 , 0.2) # probabilities of being in each state
start <- "uniform" # equal probabilities for all the states
start <- 1 # initial state is "state2"
start <- "state2" # initial state is "state2"
start <- c("state1" , "state3") # uniform distribution over "state1" and "state3"
start <- c("-" , "state2") # uniform distribution over "state1" and "state3" ("state2" excluded)

## possible ways to define transition probabilities:
transition_prob <- data.frame("action" = c("action1" , "action1" , "action1" ,
                                          0 , 0 , "action1" , 0 , 0 , 0 ,
                                         "action2" , "action2" , 1 ,
                                         1 , 1 , 1 , 1 , 1 , 1),
                                "start-state" = c("state1" , "state1" , 0 ,
                                                  "state2" , "state2" , "state2" ,
                                                  "state3" , 2 , 2 ,
                                                  "state1" , "state1" , 0 ,
                                                  "state2" , "state2" , "state2" ,
                                                  "state3" , 2 , 2),
                                "end-state" = c("state1" , "state2" , 2 ,
                                                "state1" , "state2" , "state3" ,
                                                "state1" , 1 , 2 ,
                                                "state1" , "state2" , 2 ,
                                                "state1" , "state2" , "state3" ,
                                                "state1" , 1 , 2),
                         "probability" = c(0.1 , 0.4 , 0.5 , 0 , 0.7 , 0.3 , 0.4 , 0.4 , 0.2,
                                           0 , 0.6 , 0.4 , 0.1 , 0.9 , 0 , 0.7 , 0.3 , 0))


transition_prob <- list("action1" = matrix(c( 0.1 , 0.4 , 0.5 ,
                                              0 , 0.7 , 0.3 ,
                                            0.4 , 0.4 , 0.2) , nrow = 3 , byrow = TRUE) ,
                        "action2" = matrix(c( 0 , 0.6 , 0.4 ,
                                              0.1 , 0.9 , 0 ,
                                             0.7 , 0.3 , 0 ) , nrow = 3 , byrow = TRUE))

transition_prob <- list("action1" = "identity" ,
                        "action2" = "uniform")

transition_prob <- list("action1" = matrix(c( 0.1 , 0.4 , 0.5 ,
                                              0 , 0.7 , 0.3 ,
                                            0.4 , 0.4 , 0.2) , nrow = 3 , byrow = TRUE) ,
                        "action2" = "uniform")


## possible ways to define observation probabilities:
observation_prob <- data.frame("action" = c( "*" , "*" , "*" , "*" , "*" , "*") ,
                               "end-state" = c("state1" , "state1" , "state2" ,
                                               "state2" , "state3" , "state3") ,
                               "observation" = c("obs1" , "obs2" , "obs1" ,
                                                 "obs2" , "obs1" , "obs2"),
                               "probability" = c(0.1 , 0.9 , 0.3 , 0.7 , 0.4 , 0.6))


observation_prob <- list("action1" = matrix(c(0.1 , 0.9 ,
                                              0.3 , 0.7 ,
                                              0.4 , 0.6) , nrow = 3 , byrow = TRUE) ,
                         "action2" = matrix(c(0.1 , 0.9 ,
                                              0.3 , 0.7 ,
                                              0.4 , 0.6 ) , nrow = 3 , byrow = TRUE))

observation_prob <- list("action1" = "uniform" ,
                        "action2" = "uniform")
```

```
observation_prob <- list("action1" = "uniform" ,
                         "action2" = matrix(c(0.1 , 0.9 ,
                                              0.3 , 0.7 ,
                                              0.4 , 0.6 ) , nrow = 3 , byrow = TRUE))

# as we see above, both observation probability matrices are the same and
# it indicates that the observation probabilities are independent of the action taken.
# Thus we can use the dataframe format with the help of "*" to make it more concise.
# "*" here indicates that the probabilities are independent of the actions.
# "*" could be used anytime where there is independency.

## possible ways to define the rewards:
reward <- data.frame("action" = c("action1" , "action1" , "action1" ,
                                  "action2" , "action2" , "action2") ,
                     "start-state" = c("*" , "*" , "*" , "*" , "*" , "*") ,
                     "end-state" = c("state1" , "state2" , "state3" ,
                                     "state1" , "state2" , "state3") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(10000 , 2000 , 50 , 150 , 2500 , 100))

## values:
values <- "cost"
values <- "reward"

## grid size:
grid_size <- 10 # the larger, the longer it takes to solve


### The Tiger example ##################################################################
discount <- 0.75
values <- "reward"
states <- c("tiger-left" , "tiger-right")
actions <- c("listen" , "open-left" , "open-right")
observations <- c("tiger-left" , "tiger-right")
start <- "uniform"
grid_size <- 10
transition_prob <- list("listen" = "identity" ,
                        "open-left" = "uniform" ,
                        "open-right" = "uniform")
observation_prob <- list("listen" = matrix(c(0.85 , 0.15 ,
                                             0.15 , 0.85) , nrow = 2 , byrow = TRUE) ,
                         "open-left" = "uniform" ,
                         "open-right" = "uniform")
reward <- data.frame("action" = c("listen" , "open-left" , "open-left" ,
                                  "open-right" , "open-right") ,
                     "start-state" = c("*" , "tiger-left" , "tiger-right" ,
                                       "tiger-left" , "tiger-right") ,
                     "end-state" = c("*" , "*" , "*" , "*" , "*") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(-1 , -100 , 10 , 10 , -100))
result <- pomdp(discount, states, actions, observations, start, transition_prob,
                observation_prob, reward, values = "reward", grid_size, verbose = FALSE)
```

```
result
```

`solver.output` *Return output of the POMDP solver*

Description

The function simply returns all the output generated by the 'pomdp-solve'

Usage

```
solver.output(x)
```

Arguments

`x`	object of class POMDP

Value

returns a character that contains the output of the 'pomdp-solve' including its iterations

See Also

pomdp

Examples

```
# you need to use pomdp function to generate an object of class POMDP first.
# here for an example we use the Tiger example code to generate such an object.
discount <- 0.75
values <- "reward"
states <- c("tiger-left" , "tiger-right")
actions <- c("listen" , "open-left" , "open-right")
observations <- c("tiger-left" , "tiger-right")
start <- "uniform"
grid_size <- 10
transition_prob <- list("listen" = "identity" ,
                        "open-left" = "uniform" ,
                        "open-right" = "uniform")
observation_prob <- list("listen" = matrix(c(0.85 , 0.15 ,
                                             0.15 , 0.85) , nrow = 2 , byrow = TRUE) ,
                         "open-left" = "uniform" ,
                         "open-right" = "uniform")
reward <- data.frame("action" = c("listen" , "open-left" , "open-left" ,
                                  "open-right" , "open-right") ,
                     "start-state" = c("*" , "tiger-left" , "tiger-right" ,
                                       "tiger-left" , "tiger-right") ,
                     "end-state" = c("*" , "*" , "*" , "*" , "*") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(-1 , -100 , 10 , 10 , -100))
result <- pomdp(discount, states, actions, observations, start, transition_prob,
                observation_prob, reward, values = "reward", grid_size, verbose = FALSE)

# then using the solver.output() function we print all the outputs provided by the solver
# for the given POMDP object

solver.output(result)
```

`total.expected.reward` *Return total expected reward of a POMDP*

Description

The function simply returns the total expected reward of a POMDP

Usage

```
total.expected.reward(x)
```

Arguments

x	object of class POMDP

Value

returns a numeric value indicating the total expected reward of the optimal solution

See Also

pomdp

Examples

```
# you need to use pomdp function to generate an object of class POMDP first.
# here for an example we use the Tiger example code to generate such an object.
discount <- 0.75
values <- "reward"
states <- c("tiger-left" , "tiger-right")
actions <- c("listen" , "open-left" , "open-right")
observations <- c("tiger-left" , "tiger-right")
start <- "uniform"
grid_size <- 10
transition_prob <- list("listen" = "identity" ,
                        "open-left" = "uniform" ,
                        "open-right" = "uniform")
observation_prob <- list("listen" = matrix(c(0.85 , 0.15 ,
                                             0.15 , 0.85) , nrow = 2 , byrow = TRUE) ,
                         "open-left" = "uniform" ,
                         "open-right" = "uniform")
```

```
reward <- data.frame("action" = c("listen" , "open-left" , "open-left" ,
                                  "open-right" , "open-right") ,
                     "start-state" = c("*" , "tiger-left" , "tiger-right" ,
                                       "tiger-left" , "tiger-right") ,
                     "end-state" = c("*" , "*" , "*" , "*" , "*") ,
                     "observation" = c("*" , "*" , "*" , "*" , "*") ,
                     "reward" = c(-1 , -100 , 10 , 10 , -100))
result <- pomdp(discount, states, actions, observations, start, transition_prob,
                observation_prob, reward, values = "reward", grid_size, verbose = FALSE)

# then using the total.expected.reward() function we print total expected reward of
# the optimal solution of the given POMDP object

total.expected.reward(result)
```

Index