Package 'pomdp'

December 16, 2019

Title Solver for Partially Observable Markov Decision Processes (POMDP)

Version 0.9.2

Date 2019-12-06

Description Provides an interface to pomdp-solve, a solver for Partially Observable Markov Decision Processes (POMDP). The package enables the user to simply define all components of a POMDP model and solve the problem using several methods. The package also contains functions to analyze and visualize the POMDP solutions (e.g., the optimal policy).

Depends R (>= 3.5.0)

License GPL (>= 3)

Suggests knitr, rmarkdown

VignetteBuilder knitr

LazyData true

Imports igraph

NeedsCompilation yes

Author Hossein Kamalzadeh [aut, cph, cre], Michael Hahsler [aut, cph], Anthony R. Cassandra [ctb, cph]

Maintainer Hossein Kamalzadeh <hkamalzadeh@smu.edu>

Repository CRAN

Date/Publication 2019-12-16 09:30:08 UTC

R topics documented:

`model` *Extract the User-defined Model Components from a Solved POMDP*

Description

The function returns the POMDP model components of a solved POMDP.

Usage

```
model(x)
```

Arguments

`x`	object of class POMDP returned by `solve_POMDP`.

Value

An object of class "POMDP_model", i.e., a list of all model components.

See Also

`solve_POMDP`

Examples

```
data("TigerProblem")
tiger_solved <- solve_POMDP(model = TigerProblem)
tiger_solved

model(tiger_solved)
```

plot *Visualize a POMDP Policy Graph*

Description

The function plots the POMDP policy graph in an object of class POMDP. It uses `plot` in **igraph** with appropriate plotting options.

Usage

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

Arguments

`x`	object of class POMDP.
`y`	ignored.
`belief`	logical; display belief proportions as a pie chart in each node.
`legend`	logical; display a legend for colors used belief proportions?
`cols`	colors used for the states.
`...`	plotting options passed on to `plot.igraph` in **igraph** (see `plot.common` for available options).

Details

The policy graph nodes represent segments in the value function. Each segment represents one or more believe states. The pie chart in each node (if available) represent the average belief proportions of the belief states belonging to the node/segment.

See Also

`solve_POMDP`, `plot.igraph`, `igraph_options`, `plot.common`

Examples

```
data("TigerProblem")
tiger_solved <- solve_POMDP(model = TigerProblem)
tiger_solved

## policy graph
policy_graph(tiger_solved)

## visualization
plot(tiger_solved)

library(igraph)
```

```
## use a different graph layout (circle and manual)
plot(tiger_solved, layout = layout.circle)
plot(tiger_solved, layout = rbind(c(1,1), c(1,-1), c(0,0), c(-1,-1), c(-1,1)))

## hide edge labels
plot(tiger_solved, edge.label = NA)

## custom larger vertex labels (A, B, ...)
plot(tiger_solved,
  vertex.label = LETTERS[1:nrow(solution(tiger_solved)$pg)],
  vertex.label.cex = 2,
  vertex.label.color = "white")

## add a plot title
plot(tiger_solved, main = model(tiger_solved)$name)

## plotting using the graph object
## (e.g., using the graph in the layout and to change the edge curvature)
pg <- policy_graph(tiger_solved)
plot(pg,
  layout = layout_as_tree(pg, root = 3, mode = "out"),
  edge.curved = curve_multiple(pg, .2))
```

`policy_graph` *Extract the Policy Graph (as an igraph Object)*

Description

Convert the policy graph in a POMDP solution object into an igraph object.

Usage

```
policy_graph(x, belief = TRUE, cols = NULL)
```

Arguments

`x`	A POMDP object.
`belief`	logical; add belief proportions as a pie chart in each node of the graph.
`cols`	colors used for the states in the belief proportions.

Value

An object of class igraph containing a directed graph.

Author(s)

Hossein Kamalzadeh, Michael Hahsler

See Also

solve_POMDP

Examples

```
data("TigerProblem")
tiger_solved <- solve_POMDP(model = TigerProblem)
tiger_solved

pg <- policy_graph(tiger_solved)

plot(pg)
```

POMDP *Define a POMDP Problem*

Description

Defines all the elements of a POMDP problem including the discount rate, the set of states, the set of actions, the set of observations, the transition probabilities, the observation probabilities, and rewards.

Usage

```
POMDP(discount, states, actions, observations, transition_prob,
      observation_prob, reward, start = "uniform", max = TRUE, name = NA)

R_(action, start.state, end.state, observation, value)
O_(action, end.state, observation, probability)
T_(action, start.state, end.state, probability)
```

Arguments

discount	numeric; discount rate between 0 and 1.
states	a character vector specifying the names of the states.
actions	a character vector specifying the names of the available actions.
observations	a character vector specifying the names of the observations.
transition_prob	Specifies the transition probabilities between states. Options are: • a data frame with 4 columns, where the columns specify *action*, *start-state*, *end-state* and the *probability* respectively. The first 3 columns could be either character (the name of the action or state) or integer indices. • a named list of m (number of actions) matrices. Each matrix is square of size $n \times n$, where n is the number of states. The name of each matrix the action it applies to. Instead of a matrix, also the strings "identity" or "uniform" can be specified.

`observation_prob`

Specifies the probability that a state produces an observation. Options are:

- a data frame with 4 columns, where the columns specify *action*, *end-state*, *observation* and the *probability*, respectively. The first 3 columns could be either character (the name of the action, state, or observation), integer indices, or they can be `"*"` to indicate that the observation probability applies to all actions or states. Use `rbind()` with helper function `O_()` to create this data frame.
- a named list of m matrices, where m is the number of actions. Each matrix is of size $n \times o$, where n is the number of states and o is the number of observations. The name of each matrix is the action it applies to. Instead of a matrix, also the strings `"identity"` or `"uniform"` can be specified.

`reward` Specifies the rewards dependent on action, states and observations. Options are:

- a data frame with 5 columns, where the columns specify *action*, *start.state*, *end.state*, *observation* and the *reward*, respectively. The first 4 columns could be either character (names of the action, states, or observation), integer indices, or they can be `"*"` to indicate that the reward applies to all transitions. Use `rbind()` with helper function `R_()` to create this data frame.
- a named list of m lists, where m is the number of actions (names should be the actions). Each list contains n named matrices where each matrix is of size $n \times o$, in which n is the number of states and o is the number of observations. Names of these matrices should be the name of states.

`start` Specifies the initial probabilities for each state (i.e., the initial belief state) used to find the initial node in the policy graph and to calculate the total expected reward. The default initial belief state is a uniform distribution over all states. No initial belief state can be used by setting `start = NULL`. Options to specift start are:

- a probability distribution over the n states. That is, a vector of n probabilities, that add up to 1.
- the string `"uniform"` for a uniform distribution over all states.
- an integer in the range 1 to n to specify a single starting state. or
- a string specifying the name of a single starting state.
- a vector of strings, specifying a subset of states with a uniform start distribution. If the first element of the vector is `"-"`, then the following subset of states is excluded from the set of start states.

`max` logical; is this a maximization problem (maximize reward) or a minimization (minimize cost specified in `reward`)?

`name` a string to identify the POMDP problem.

`action, start.state, end.state, observation, probability, value`

Values used in the helper functions `O_()`, `R_()`, and `T_()` to create an entry for `observation_prob`, `reward`, or `transistion_prob` above, respectively.

Details

POMDP problems can be solved using `solve_POMDP`. Details about the available specifications can be found in [1].

Value

The function returns an object of class POMDP which is list with an element called `model` containing a list with the model specification. `solve_POMDP` reads the object and adds a list element called `solution`.

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: http://www.pomdp.org

See Also

solve_POMDP

Examples

```
## The Tiger Problem

TigerProblem <- POMDP(
  name = "Tiger Problem",

  discount = 0.75,

  states = c("tiger-left" , "tiger-right"),
  actions = c("listen", "open-left", "open-right"),
  observations = c("tiger-left", "tiger-right"),

  start = "uniform",

  transition_prob = list(
    "listen" =     "identity",
    "open-left" =  "uniform",
    "open-right" = "uniform"),

  observation_prob = list(
    "listen" = rbind(c(0.85, 0.15),
                     c(0.15, 0.85)),
    "open-left" =  "uniform",
    "open-right" = "uniform"),

  # the rew helper expects: action, start.state, end.state, observation, value
  reward = rbind(
    R_("listen",     "*",           "*", "*", -1  ),
    R_("open-left",  "tiger-left",  "*", "*", -100),
    R_("open-left",  "tiger-right", "*", "*", 10  ),
    R_("open-right", "tiger-left",  "*", "*", 10  ),
    R_("open-right", "tiger-right", "*", "*", -100)
```

```
  )
)

TigerProblem

model(TigerProblem)
```

`reward` *Calculate the Reward for a POMDP Solution*

Description

This function calculates the expexted total reward for a POMDP solution given a starting belief state.

Usage

```
reward(x, start = "uniform")
```

Arguments

`x`	a POMDP solution (object of class POMDP).
`start`	specification of the starting belief state (see argument start in `POMDP` for details).

Details

The value is calculated using the value function stored in the POMDP solution.

Value

A list with the components

`total_expected_reward`
: the total expected reward starting with the initial policy graph node representing the starting belief state.

`initial_pg_node`
: the policy graph node that represents the starting belief state.'

`start_belief_state`
: the starting belief state specified in `start`.

Author(s)

Michael Hahsler

See Also

`POMDP`, `solve_POMDP`

Examples

```
data("TigerProblem")
tiger_solved <- solve_POMDP(model = TigerProblem)

# if no start is specified, a uniform belief is used.
reward(tiger_solved)

# we have additional information that makes us belief that the tiger
# is more likely to the left.
reward(tiger_solved, start = c(0.85, 0.15))

# we start with strong evidence that the tiger is to the left.
reward(tiger_solved, start = "tiger-left")

# Note that in this case, the total discounted expected reward is greater
# than 10 since the tiger problem resets and another game staring with
# a uniform belief is played which produces addional reward.
```

---

solution *Extract the Solution of a POMDP*

Description

The function extracts the solution of a POMDP as an object of class POMDP_solution which is a list containing, e.g., the policy graph (pg) and the hyper-plane coefficients (alpha).

Usage

```
solution(x)
```

Arguments

x	object of class POMDP returned by solve_POMDP.

Value

returns an object is of class POMDP_solution, i.e., a list of all solution elements.

See Also

solve_POMDP

Examples

```
data("TigerProblem")
tiger_solved <- solve_POMDP(model = TigerProblem)
tiger_solved

solution(tiger_solved)
```

solver_output *Display the Output of the POMDP Solver*

Description

Displays the output generated by the solver 'pomdp-solve'. This includes used parameters, and iterations (i.e., epochs). This produces the same output as running `solve_POMDP` with the argument `verbose = TRUE`.

Usage

```
solver_output(x)
```

Arguments

x	object of class POMDP returned by `solve_POMDP`.

Value

returns invisibly a character string vector with the output of 'pomdp-solve'.

See Also

`solve_POMDP`

Examples

```
data("TigerProblem")
sol <- solve_POMDP(model = TigerProblem)

## solver output
solver_output(sol)
```

solve_POMDP *Solve a POMDP Problem*

Description

This function utilizes the 'pomdp-solve' program (written in C) to use different solution methods [2] to solve problems that are formulated as partially observable Markov decision processes (POMDPs) [1]. The result is a (close to) optimal policy.

Usage

```
solve_POMDP(model, horizon = NULL, method = "grid", parameter= NULL, verbose = FALSE)
solve_POMDP_parameter()
```

Arguments

model	a POMDP problem specification created with POMDP. Alternatively, a POMDP file or the URL for a POMDP file can be specified.
method	string; one of the following solution methods: "grid", "enum", "twopass", "witness", or "incprune". Details can be found in [1].
horizon	an integer with the number of iterations for finite horizon problems. If set to NULL, the algorithm continues running iterations till it converges to the infinite horizon solution.
parameter	a list with parameters passed on to the pomdp-solve program.
verbose	logical, if set to TRUE, the function provides the output of the pomdp solver in the R console.

Details

solve_POMDP_parameter() displays available solver parameter options.

Note: The parser for POMDP files is experimental. Please report problems here: https://github.com/farzad/pomdp/issues.

Value

The solver returns an object of class POMDP which is a list with the model specifications (model), the solution (solution), and the solver output (solver_output). The elements can be extracted with the functions model, solution, and solver_output.

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: http://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

```
data("TigerProblem")
TigerProblem

tiger_solved <- solve_POMDP(model = TigerProblem, parameter = list(fg_points = 10))
tiger_solved

## look at the model
model(tiger_solved)

## look at the solution
solution(tiger_solved)
```

```
## look at solver output
solver_output(tiger_solved)

## plot the policy graph
plot(tiger_solved)

## display available solver options which can be passed on to the solver as parameter.
solve_POMDP_parameter()

## solve a POMDP from http://www.pomdp.org/examples
sol <- solve_POMDP("http://www.pomdp.org/examples/cheese.95.POMDP")
sol
plot(sol)
```

`TigerProblem` *Tiger Problem POMDP Specification*

Description

The model for the Tiger Problem [1].

Usage

```
data("TigerProblem")
```

Format

A list with the elements: discount, states, actions, observations, start, transition_prob, observation_prob, reward, name.

Details

The Tiger Problem is defined as follows [1]. A tiger is put with equal probability behind one of two doors, while treasure is put behind the other one. You are standing in front of the two closed doors and need to decide which one to open. If you open the door with the tiger, you will get hurt by the tiger (negative reward), but if you open the door with the treasure, you receive a positive reward. Instead of opening a door right away, you also have the option to wait and listen for tiger noises. But listening is neither free nor entirely accurate. You might hear the tiger behind the left door while it is actually behind the right door and vice versa.

The states of the system are tiger behind the left door (tiger-left) and tiger behind the right door (tiger-right).

Available actions are: open the left door (open-left), open the right door (open-right) or to listen (listen).

Rewards associated with these actions depend on the resulting state: +10 for opening the correct door (the door with treasure), -100 for opening the door with the tiger. A reward of -1 is the cost of listening.

As a result of listening, there are two observations: either you hear the tiger on the right (tiger-right), or you hear it on the left (tiger-left).

The transition probability matrix for the action listening is identity, i.e., the position of the tiger does not change. Opening either door means that the game restarts by placing the tiger uniformly behind one of the doors.

References

[1] Anthony R. Cassandra, Leslie P Kaelbling, and Michael L. Littman (1994). Acting Optimally in Partially Observable Stochastic Domains. In Proceedings of the Twelfth National Conference on Artificial Intelligence, pp. 1023-1028.

Examples

```
data(TigerProblem)
TigerProblem

# solve the problem and look at the optimal policy graph (as a table and as a plot)
sol <- solve_POMDP(TigerProblem)
sol

solution(sol)$pg
plot(sol)
```

`write_POMDP` *Write a POMDP Model to a File in POMDP Format*

Description

Writes a POMDP file suitable for the pomdp-solve program. This function is used internally.

Usage

```
write_POMDP(model, file)
```

Arguments

`model`	an object of class POMDP_model.
`file`	a file name.

Author(s)

Hossein Kamalzadeh, Michael Hahsler

References

POMDP solver website: http://www.pomdp.org

See Also

POMDP

Index