Package 'pomdpSolve'

August 31, 2023

Title Interface to 'pomdp-solve' for Partially Observable Markov Decision Processes

Version 1.0.4

Date 2023-08-31

Description Installs an updated version of 'pomdp-solve', a program to solve Partially Observable Markov Decision Processes (POMDPs) using a variety of exact and approximate value iteration algorithms. A convenient R infrastructure is provided in the separate package pomdp. Kaelbling, Littman and Cassandra (1998) <doi:10.1016/S0004-3702(98)00023-X>.

Classification/ACM G.4, G.1.6, I.2.6

URL https://github.com/mhahsler/pomdpSolve

BugReports https://github.com/mhahsler/pomdpSolve/issues

Depends R (>= 3.5.0)

Imports utils

Suggests pomdp

Encoding UTF-8

License GPL (>= 3)

Copyright pomdp-solve is Copyright (C) Anthony R. Cassandra; LASPack is Copyright (C) Tomas Skalicky; lp-solve is Copyright (C) Michel Berkelaar, Kjell Eikland, Peter Notebaert; all other code is Copyright (C) Michael Hahsler.

RoxygenNote 7.2.3

NeedsCompilation yes

Author Michael Hahsler [aut, cph, cre] (<https://orcid.org/0000-0003-2716-1405>), Anthony R. Cassandra [aut, cph]

Maintainer Michael Hahsler <mhahsler@lyle.smu.edu>

Repository CRAN

Date/Publication 2023-08-31 19:20:02 UTC

R topics documented:

`pomdpSolve-package` *pomdpSolve: Interface to 'pomdp-solve' for Partially Observable Markov Decision Processes*

Description

Installs an updated version of 'pomdp-solve', a program to solve Partially Observable Markov Decision Processes (POMDPs) using a variety of exact and approximate value iteration algorithms. This package only provides the executable and a few reading routines. A convenient R infrastructure to use the solver is provided in the separate package **pomdp** (pomdp::pomdp-package).

Key functions

- Solve a POMDP file with pomdp-solve using `pomdp_solve()`.
- Read and write files for pomdp-solve (see read_write).
- Find the pomdp-solve executable using `find_pomdp_solve()`.

Package pomdp provides more convenient support to

- Define a POMDP using pomdp::POMDP
- Solve a POMDP using `pomdp::solve_POMDP()`

Author(s)

Michael Hahsler

`find_pomdp_solve` *Find the executable for 'pomdp-solve'*

Description

Find the `pomdp-solve` executable to solve Partially Observable Decision Processes (POMDPs) (Kaelbling et al, 1998) installed by the **pomdpSolve** package.

Usage

```
find_pomdp_solve()
```

Details

This package only provides a direct interface to the executable. A more convenient and powerful interface is provided by the function `pomdp::solve_POMDP()` in package **pomdp**.

The executable of `pomdp-solve` in this direct interface needs to be called with `system2()` and runs in a separate process. This way, a failure in the solver will not compromise the R session. `pomdp-solve` creates files with the value function and the policy graph (see read_write).

Value

returns the path to the 'pomdp-solve' executable as a string or stops with an error.

References

Kaelbling, L.P., Littman, M.L., Cassandra, A.R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence.* **101** (1–2): 99-134. doi:10.1016/S00043702(98)00023X

Anthony R. Cassandra, pomdp-solve documentation, `https://www.pomdp.org/code/index.html`

See Also

read_write

Examples

```
# find the location of the pomdp-solve executable
find_pomdp_solve()

# get pomdp-solve options
system2(find_pomdp_solve(), args = "-h")

# an example of how to solve a simple POMDP can be found in the man page
# for read_write.
```

`pomdp_solve` *Solving a POMDP with 'pomdp-solve'*

Description

This function provides a bare bones interface to run pomdp-solve on a POMDP file. The results can be read with the function provides in read_write.

Usage

```
pomdp_solve(pomdp, options = list(), verbose = TRUE)

pomdp_solve_help()
```

Arguments

pomdp	the POMDP file to solve.
options	a list with options for pomdp-solve.
verbose	logical; show the program text output?

Details

Calling `solve_pomdp()` first cleans results from previous runs and then executes pomdp-solve with the specified options.

The options are specified in `options` as a list with entries of the form `<option> = <value>`. `pomdp_solve_help()` displays the available options. Note that the leading dash is not used on the option name. For example: `list(method = "grid", epsilon = 0.1)` sets the method option to grid and epsilon to 0.1. Here is a slightly more detailed description of pomdp-solve's options.

Value

nothing

References

Anthony R. Cassandra, pomdp-solve documentation, https://www.pomdp.org/code/index.html

See Also

find_pomdp_solve read_write

Examples

```
# display available options
pomdp_solve_help()

# solve a POMDP file that ships with this package in a temporary directory
old_wd <- setwd(tempdir())

file.copy(system.file("tiger.aaai.POMDP", package = "pomdpSolve"), "./tiger.aaai.POMDP")

# Example 1: run solver to completion
pomdp_solve("tiger.aaai.POMDP", options = list(method = "incprune"))
dir()
# you can inspect the files with file.show()

# read the raw policy graph (-0 means infinite horizon solution)
read_pg_file("tiger.aaai-0.pg")

# read the raw value function
read_alpha_file("tiger.aaai-0.alpha")

# Example 2: use method finite grid (point-based algorithm) and save the used belief points
pomdp_solve("tiger.aaai.POMDP", options = list(method = "grid", fg_save = TRUE))
dir()
```

```
read_belief_file("tiger.aaai-0.belief")

# Example 3: Stop value iteration after 50 epochs and then continue with a second call
pomdp_solve("tiger.aaai.POMDP", options = list(method = "incprune", horizon = 50))
alpha <- read_alpha_file("tiger.aaai-0.alpha")

write_terminal_values("terminal.alpha", alpha)
pomdp_solve("tiger.aaai.POMDP", options = list(method = "incprune",
  terminal_values = "terminal.alpha"))

# return to the old directory
setwd(old_wd)
```

`read_write` *Read and Write Files for 'pomdp-solve'*

Description

Read and write files for the pomdp-solve executable.

Usage

```
read_alpha_file(file)

read_pg_file(file)

read_belief_file(file)

write_grid_file(file, belief_points, digits = 7)

write_terminal_values(file, alpha, digits = 7)
```

Arguments

`file`	name of the file to read from or to write to.
`belief_points`	a numeric matrix with the number of states columns. Rows represent belief points.
`digits`	number of digits used to write files.
`alpha`	a numeric alpha vector with the length of the number of states.

Details

pomdp-solve uses text format for its input and output. The input is a POMDP file. The outputs are the following.

Value Function

The value function is returned as files with the extension `.alpha` in the format:

```
A
V1 V2 V3 ... VN

A
V1 V2 V3 ... VN

...
```

Where `A` is an action number and the `V1` through `VN` are real values representing the components of a particular vector that has the associated action. The action number is the 0-based index of the action as specified in the input POMDP file. The vector represents the coefficients of a hyperplane representing one facet of the piecewise linear and convex (PWLC) value function. Note that the length of the lists needs to be equal to the number of states in the POMDP.

`read_alpha_file()` reads the V components from the file and returns a matrix.

Policy Graph

The policy graph is returned as a file with the extension `.pg`. Each line of the file represents one node of the policy graph and its contents are:

```
N A  Z1 Z2 Z3 ...
...
```

Here `N` is a node ID giving the node a unique name, numbered sequentially and lining up with the value function vectors in the corresponding output `.alpha` file above.

The `A` is the action number defined for this node; it is an integer referring to the the POMDP file actions by its 0-based index number. These are followed by a list of node IDs, one for each observation. Thus the list will have a length equal to the number of observations in the POMDP. This list specifies the transitions in the policy graph. The nth number in the list will be the index of the node that follows this one when the observation received is `n`.

`read_pg_file()` returns a data.frame with the nodes in the policy graph as rows.

Terminal Values

Terminal values can be specified as a single alpha vector.

Grid-based Solver Specific Files

The grid-based method can write the used belief points do disk (command line option `-fg_save`). The file can be read using `read_belief_file()`.

A matrix with belief points can be written using `write_grid_file()`. This file can be used

Details about the file formats and pomdp-solve can be found in the References section.

See `pomdp_solve()` for examples.

Value

- `read_alpha_file()` returns the value function (alpha vectors) as a matrix.
- `read_pg_file()` returns the policy graph as a data.frame.
- `read_belief_file()` returns a matrix if the solver wrote a belief file.
- `write_grid_file()` returns nothing.
- `write_terminal_values()` returns nothing.

References

Anthony R. Cassandra, pomdp-solve documentation, https://www.pomdp.org/code/index.html

See Also

find_pomdp_solve

Index