POMDPFiles.jl

March 31, 2024

Contents

Part I

Home

Chapter 1

Home

This is the first page of my documentation.

1.1 Types

POMDPFiles.ContainerNames – Type.

```
ContainerNames

    names: Vector{String} - a vector with the names of the actions, states, or observations.
    number: Int - the number of actions, states, or observations.

This type was created to store the names and number of actions, states, and observations when
↪  parsed from the preamble of a .pomdp file format.
```

source

POMDPFiles.InitialStateParam – Type.

```
InitialStateParam was created to store the initial state distribution of a POMDP. The
↪  distribution can be of any type, but it is stored as a vector of Float64 values.
```

source

POMDPFiles.SFilePOMDP – Type.

```
SFilePOMDP is used whenever the names of the states, actions, and observations are known. It is
↪  used to represent a POMDP problem from a file.

    dic_states: dictionary with the names of the states
    dic_actions: dictionary with the names of the actions
    dic_obs: dictionary with the names of the observations
    pomdp: FilePOMDP structure
```

source

POMDPFiles.FilePOMDP – Type.

```
FilePOMDP is the main data structure of the package. It is used to represent a POMDP problem
↪  from a file.

    ns: number of states
    na: number of actions
    no: number of observations

    support_initialstate: support of the initial state distribution
    initialstate_distribution: initial state distribution

    discount: discount factor

    T: transition matrix
    O: observation matrix
    R: reward matrix
```

source

1.2 Functions

POMDPFiles.test_if_probability – Function.

```
test_if_probability(prob::Union{Vector{Float64}, Vector{Nothing}, Nothing};rtol=1e-3)

Built-in function that tests whether a vector is a probability distribution. It checks if the
↪  elements are between 0 and 1 and if the sum of the elements is approximately 1. The function
↪  returns true if the vector is a probability distribution and false otherwise.
```

source

POMDPFiles.read_pomdp – Function.

```
Read a .pomdp file following the specfication at http://www.pomdp.org/code/pomdp-file-spec.html
↪  and returns a FilePOMDP or SFilePOMDP object that can be used within the POMDPs.jl
↪  interface.
```

source

Base.write – Function.

Writes out the alpha vectors in the .alpha file format

source

Write out a .pomdp file using the POMDPs.jl interface Specification: http://cs.brown.edu/research/ai/pomdp/examples/pomdp-file-spec.html A more recent version of the spec: https://pomdp.org/code/pomdp-file-spec.html

source

POMDPFiles.prob – Function.

```
 value returns the a vector with the initial state distribution.
```

source

POMDPFiles.number – Function.

```
number returns the number of actions, states, or observations in the ContainerNames object.
```

source

```
number returns the number of states in the initial state distribution.
```

source

POMDPFiles.remove_comments_and_white_space – Function.

```
remove_comments_and_white_space(file::Vector{String}) is used by read_pomdp to remove comments
↪  and white spaces from the file. This function allows for some standardization of process of
↪  parsing files.
```

source

POMDPFiles.read_alpha – Function.

```
Read a `.alpha` file as generated by pomdp-solve.
Works the same was as `read_pomdp` in `POMDPXFile.jl`.

The `.alpha` file format is recapped here as follows,
see: http://www.pomdp.org/code/alpha-file-spec.html

A set of vectors is the representation use for the value function and each
vector has an action associated with it. The vectors represent the coefficients
of a hyperplane passing through the origin. The format specified here is what is
output from the 'pomdp-solve' program and what will be necessary for input to
the 'pomdp-solve' program with the '-terminal_values' command line option.

The format is simply:

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 specificed in
the input POMDP file. The vector represents the coefficients of a hyperplane
representing one facet of the piecewise linear and convex value function.
Note that the length of the lists needs to be equal to the number of states in
the POMDP.

To find which action is the "best" for a given set of alpha vectors, the belief
state probabilities would be used in a dot product against each alpha vectors'
coefficients. The action associated with the vector with the highest value is
the best action to take for that belief state given the value function.
```

source

POMDPs.states – Function.

```
    Implementing the functions required by the POMDP interface. See
    ↪  [POMDPs.jl](https://juliapomdp.github.io/POMDPs.jl/latest/) for more details on the
    ↪  interface.
```

source

```
Implementing the functions required by the POMDP interface. See
↪  [POMDPs.jl](https://juliapomdp.github.io/POMDPs.jl/latest/) for more details on the
↪  interface.
```

source

POMDPFiles.convert_to_data_structure – Function.

```
convert_to_data_structure(field::String, preamble::Dict{String,String}) is used by read_pomdp to
↪  convert the information in the preamble into an intermidiate format before passing it into a
↪  ContainerNames object.
```

source

POMDPFiles.check_preamble_fields – Function.

```
check_preamble_fields(preamble::String) is used by read_pomdp to check if the preamble of the
↪  file has all the necessary fields. An error is issued if one of the fields "discount",
↪  "values", "states", "actions", or "observations" is missing.
```

source

POMDPFiles.support – Function.

support returns the support of the initial state distribution.

source

POMDPFiles.process_preamble – Function.

```
process_preamble(preamble::Dict{String, String}) is used by read_pomdp to process the preamble
↪  of the file and check if the fields "discount", "values", "states", "actions", and
↪  "observations" have the correct syntax. The output are the discount, values, actions,
↪  states, and observations parameters, where actions, states, and observations are converted
↪  into ContainerNames objects.
```

source

POMDPFiles.order_of_transition_reward_observation – Function.

```
order_of_transition_reward_observation(file_lines::Vector{String}, start_line::Int64) is used by
↪  read_pomdp to find the order of the transition, reward, and observation matrices in the
↪  file.
```

source