A Shiny implementation of the Threat-Species MOMDP

This document serves as a user manual for the Shiny implementation of the threat-species MOMDP described in "A general optimal adaptive management approach for demonstrating effectiveness in threatened species management" (Nicol et al, 2021). The app is an interactive user interface for eliciting the necessary parameters, solving the MOMDP and exploring the solution via simulation.

Licence:

The code is licensed using the MIT open source licence. See licence information on github. Basically this is free to use and modify with attribution. No warranty or liability is given for the code.

Installation:

The app is written in the R programming language (version 4.0.2; note that testing on earlier R versions created warnings and issues rendering the plots in the app). It has been tested using RStudio (version 1.3.1093) on Windows 10 version 20H2. We recommend using RStudio to run the app, since it has features such as progress messages that will be printed to the console.

The app is available from github at: https://github.com/nicols02/SpeciesThreatAM.git. Source code can be obtained from this URL by cloning the repository to a directory on your local system.

You will need to install a Linux distribution to 'make' the sarsop file. We recommend using WSL2, which is a Windows supported tool for running Linux distributions through your Windows 10 machine. To install WSL2, follow the instructions at:
https://docs.microsoft.com/en-us/windows/wsl/install-win10.
Note that if you choose a software other than WSL (e.g. Cygwin) then you will need to manually update the code in the app that calls sarsop (line 405-6). The syntax for use with Cygwin is included at these lines but is commented out in the code.

After installing WSL, open Linux (i.e. Ubuntu) and run "`sudo apt update && sudo apt upgrade`" to update your Ubuntu packages to the latest version.
Once you have installed WSL and a Linux distribution (we used Ubuntu), you need to navigate to the folder where you stored sarsop. To access your C:, you need to open the Ubuntu drive and type "cd /mnt/c/". From here it should be straightforward to cd (i.e. change directory) to the folder where you stored sarsop and locate the sarsop/src folder. For example, my default location was:

"`cd /mnt/c/Users/<your_username>/Documents/SpeciesThreatAM/sarsop/src`".

Next you'll need to 'make' the sarsop files so that you can run sarsop. You'll need to add the 'make' command to Ubuntu, since you only installed a minimal Linux distribution. To do this, type the following into the Ubuntu console:

"`sudo apt-get install build-essential`"

You'll need to enter your password that you created when you installed and first launched Ubuntu. Build-essential gives you a suite of important functions needed by sarsop including make and gcc.

Once you've added the make command and are in the sarsop/src directory, you can simply type "`make`".

A good troubleshooting step here is to check that sarsop is installed. From Ubuntu, try to solve an example problem by entering the following into the console:
`"./pomdpsol ../examples/POMDPX/Tiger.pomdpx"`
You should see sarsop solve a toy problem on the screen and a policy file will be generated to `"out.policy"` by default (see https://github.com/AdaCompNUS/sarsop for more information). If this works, then sarsop is installed correctly.

You can also check that WSL2 is working in the Windows command window. To do this, open a windows cmd prompt (Click the windows icon and type "cmd"). Navigate to the sarsop/src directory and type:
`"wsl ./pomdpsol ../examples/POMDPX/Tiger.pomdpx"`
This should solve the same problem as above, but via the Windows command prompt. If this works, then sarsop should be ready to run in the Shiny app.

Note that the Shiny app will not run from a URL (e.g. `shiny::runGitHub("SpeciesThreatAM", "nicols02", ref= "main")`) because it has a predefined folder structure that reads and writes files as you step through the process of solving the MOMDP.

Running from RStudio:

The Shiny app is stored in a file called "`app.R`". To run the app, place the files into an app directory in your working directory. First, include the shiny library by typing "`library("shiny")` into the console. You can then launch the app in R using either `runApp("app.R")`, or by opening the app file in RStudio and clicking "Run App" in the top right hand corner of the scripting window.

"app.R" sources three other files that contain functions used by the app. These are called "generate_transition_matrices.R", "sarsop_parse_Shiny.R", and "alpha_min_fast.R". These should be sourced automatically by the app, but you'll need to be sure that were download from the git repository and that they're stored in the same directory as the app. The packages used by app.R should be automatically downloaded and sourced if you don't already have them installed —this may take a few moments the first time that the app is run and progress will be documented in the RStudio console window.

Notes on additional software:

The app calls other software, in particular the SARSOP MOMDP solver (original version available from https://github.com/AdaCompNUS/sarsop). If installed directly from source, SARSOP requires a linux environment (WSL2 is recommended if you are on Windows—see text above). However, an unpacked version of SARSOP is packaged with the app, so there should be no need to install it from its original location. We made minor modifications to the SARSOP code (specifically, our implementation stores the sampled belief vectors in a file for use in alpha-min-fast), so we recommend using the version that comes with our app.

As well as a modified version of SARSOP the alpha-min-fast algorithm uses the lp_solve program (http://lpsolve.sourceforge.net/5.5/). GraphViz (https://graphviz.org/) may be useful for viewing policy graphs, but we used the R package DiagrammeR to visualise .dot files produced by SARSOP and alpha-min-fast. DiagrammeR will automatically install and load when the app is run.

Using the app

1. Simulation and setup page

After launching the app, you should see a two-column layout (Figure 1). The left sidebar contains information that is required to generate the POMDP solution. There are default values in each box but these can be edited as required to solve different problems. The right sidebar may take a moment to load. It shows simulations of the system given the parameters on the left. As you update the values in the sidebar, the simulations will update automatically. Users can use the simulations to see the trajectory of the species and the threat. The visualisations provide a sanity check on the values entered into the elicitation boxes by allowing the users to see the impacts of the values they enter on the long-term behaviour of the system. If an optimal solution has previously been generated then this will be plotted on the graph too.

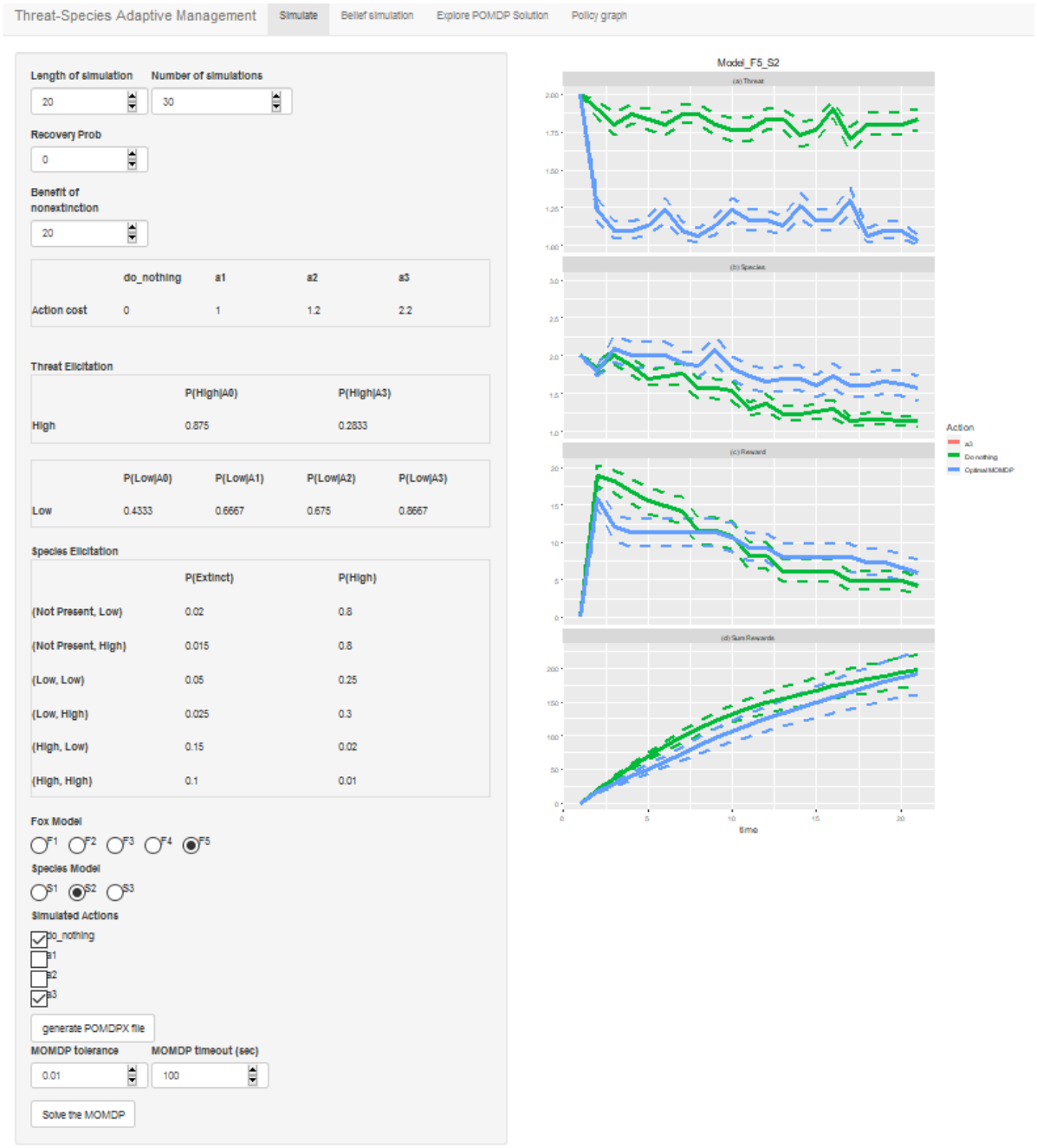


Figure 1: The Simulate tab. This is the view that you should see when you first open the app.

Below we provide information on each of the fields in the left sidebar. Acceptable values for each field are included in the brackets following the description.

Left sidebar fields description:

- **Length of simulation:** The number of years to run the simulations—used only for exploring the simulated performance in the plots on the right (integer).
- **Number of simulations:** The number of simulation runs to plot in the simulated performance plots on the right (integer).
- **Recovery prob:** The annual probability that a locally extinct species is reintroduced (e.g. by immigration or translocation). Default is zero (decimal value between 0 and 1).
- **Benefit of non-extinction:** The relative benefit of the species being in either the low or high states. The extinct state has a benefit score of 0. The units are the same as the units of action cost. The default values are benchmarked against the cost of action a1. See the manuscript for further details of the reward function (positive number).
- **Action cost matrix:** This matrix contains the relative costs of each action. As in the manuscript, there are 3 actions plus a do nothing action (a0). Actions a1 and a2 are general actions that can have any characteristics, however a3 is a combination of actions a1 and a2[1]; see the main text for further information about the assumed relationship between actions (positive number for each matrix entry).
- **Threat elicitation:** These two matrices elicit the response of the threatening process to the management actions. The column on the left lists the current threat state. The columns to the right request the user to enter the probability of remaining in the threat state given that the actions are implemented. For example, the first entry in this matrix looks like this:

Threat Elicitation	
	P(High|A0)
High	0.875

  This entry can be read as "Given that the threat state is currently high (left column), the probability that the threat state remains high, given action A0 is implemented, is equal to 0.875". Note that there are fewer elicitations required for the high state because we implement the interpolation method of (Cain 2001) to reduce the number of questions required (positive number between 0-1 for each matrix entry).
- **Species elicitation:** This matrix elicits the response of the species to the threatening process. The column on the left lists the current threat state and species state respectively. The columns to the right request the user to enter the probability of transitioning to the locally extinct and high species states given the current threat and species states. For example, the first row in this matrix looks like this:

[1] The relationships between actions are hard coded into the app for the purposes of this paper, however if they need to be changed for other uses, they are found in the "sarsop_parse" function in the source file "sarsop_parse_Shiny.R". The relevant variable is "model.effectiveness.Mat" (line 71).

Species Elicitation

	P(Extinct)	P(High)
(Not Present, Low)	0.02	0.8

This entry can be read as: "Given that the threat is not present and the species state is currently low, the probability that the species will be locally extinct next year is 0.02. The probability that the species state will be high next year is 0.8" (positive number between 0-1 for each matrix entry; must sum to less than or equal to 1).

- **Threat model and species model:** These radio buttons tell the plot window on the right which models to simulate as the 'true' model dynamics. They can be used for scenario exploration and to understand how changing the assumptions about species and threat interactions will affect expected performance. The plots on the right display the performance of each action given the model dynamics selected in these radio buttons. Clicking on a different combination of species and threat models will cause the plots to automatically update to reflect the changes in the true dynamics. These selected "true" dynamics are also carried across to the "belief simulation" page of the app (see below). The threat models F1-5 represent different assumptions about the effectiveness of management actions, i.e.:
    - F1 All actions ineffective
    - F2 A1, A3 effective, A2 ineffective
    - F3 A1 ineffective, A2, A3 effective
    - F4 A3 effective (A1, A2 ineffective)
    - F5 Any action is effective (A1-A3 effective)

  Note that A3 (combined action) is effective in all models except F1 and that A0 (do nothing) is always ineffective. The species models S1-3 represent different assumptions about the impacts of threat state on species:

    - S1 species respond negatively to any level of threat presence (high or low threat)
    - S2 species respond negatively to high threat presence (no or limited impact of low threat density)
    - S3 species don't respond to threat presence (no impact of either high or low threat density)
- **Simulated actions:** These checkboxes allow the user to select which actions are plotted in the figures on the right. Multiple selections can be made.
- **Generate POMDPX file:** This button will run a script to generate the MOMDP input file in the SARSOP POMDPX file (a factored format that utilises the conditional structure of the problem to produce a more compact representation). On clicking the button, a message (i.e. "starting writing POMDPX file") will appear in the RStudio window. It should run almost instantaneously. When the script is finished, a message ("finished writing to \"./pomdpx_files/filename\"") will appear in the RStudio window. The pomdpx file will be saved in "./pomdpx_files/sarsop_input_ShinyGrab.pomdpx". It can be viewed with any text editor, e.g. Notepad++.
- **MOMDP tolerance and timeout:** These numbers specify the stopping conditions for SARSOP when solving the MOMDP. The tolerance is a precision threshold for successive iterations of

the value function; timeout is a time (in seconds) before SARSOP returns a solution. SARSOP will stop when either of the two stopping conditions are satisfied. For our default problem, we recommend a tolerance of 0.5. If the tolerance is too low then SARSOP will still solve (it stops after the maximum time specified by the timeout), but it may cause issues with the alpha-min-fast algorithm on the last page if the solution generates too many alpha vectors.

- **Solve the MOMDP:** This button calls SARSOP to solve the MOMDP using the stopping conditions in the previous step. Progress output from SARSOP is written to the console; it is finished when the following message is displayed in the console: "finished solving. Writing to: ./pomdp_solved/ShinySolution_0_xx_xx.policy", where xx depends on the value of "benefit of nonextinction" entered above. The policy file can be viewed with any text editor, e.g. Notepad++. Once the MOMDP policy file exists, the plots on the right will update automatically to display the simulated performance of the MOMDP solution for the given assumed dynamics. Note that the optimal MOMDP will not necessarily out-perform all the other actions, since the MOMDP doesn't "know" the true dynamics and is learning the best actions to take based on past feedback. It is the optimal policy when we start from ignorance and hedge our bets across all of the models.
- **Note that the subsequent pages of the app will not work until the MOMDP has been solved.** The app looks for the existence of the ShinySolution_0_xx_xx.policy; if it is not found then the other pages will not work. However if the policy file exists from previous runs, it will plot even though you may have made changes while on the Simulate Tab. To avoid this, *it is good practice to solve the MOMDP each time you launch the app before moving on from the Simulate tab* (to clear out any previous runs), otherwise the subsequent pages may be displaying incorrect results.

Right sidebar plots

The four plots on the right all show the simulated performance of a quantity over time and are controlled by the options in the left sidebar. The assumed system dynamics model for the simulations is in the title, i.e. **Model_Fx_Sx** (controlled by the radio buttons on the left) The plots depict the following, from top to bottom: the mean **(a) Threat; and (b) Species** states throughout the simulated period. The mean instantaneous **(c) Reward** received at the each timestep for each depicted action, and the mean cumulative **(d) Sum of rewards** over the duration of the simulation. The sum of rewards is the optimisation criterion for the MOMDP.

In the threat plot, the high threat state corresponds with a value of 2; low threat corresponds to a value of 1. In the species plot, the high state corresponds with a value of 3, low with value 2, and locally extinct has a value of 1. The initial starting state is assumed to be high threat, low species.

2. Belief Simulation page

Opening the belief simulation page (tabs on top of the screen) will cause the app to run a belief simulation assuming the model dynamics displayed on the Simulate page (See Figure 2 for an example). This page provides further diagnostic plots to help understand the optimal policy. The top two figures show the evolution of the (marginal) belief in each threat and species model over the duration of the simulation, assuming a uniform initial belief. The plots are annotated with the terminal belief state, which is the belief state at the terminal time of the simulation—this is useful for diagnosing which plots overlap where they have identical belief evolutions over time.

The bottom plot shows the frequency that each action was selected in the simulations at each time step, providing an indication of the optimal policy. In the example in figure 2, the optimal policy is generally to select action a3.

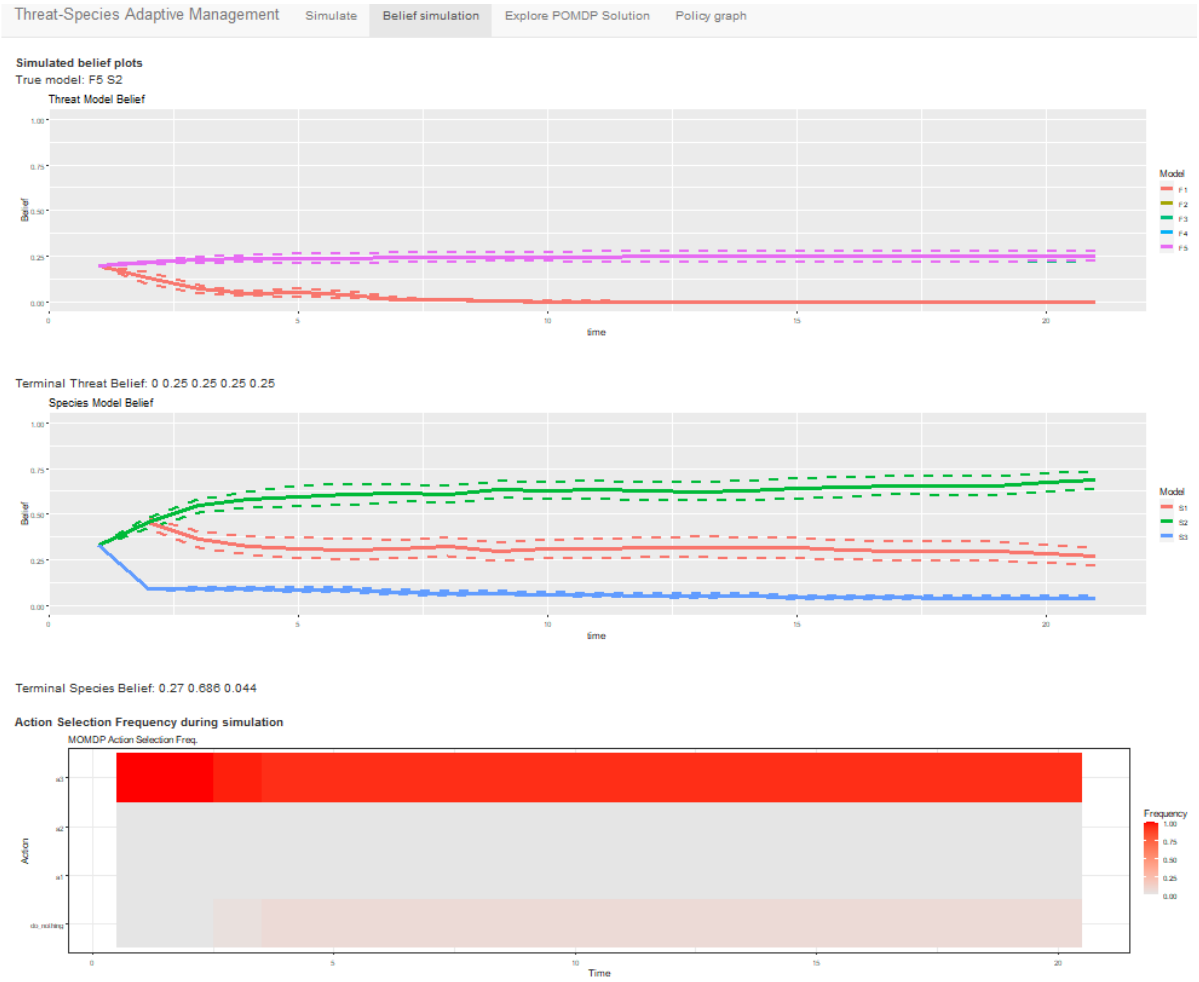
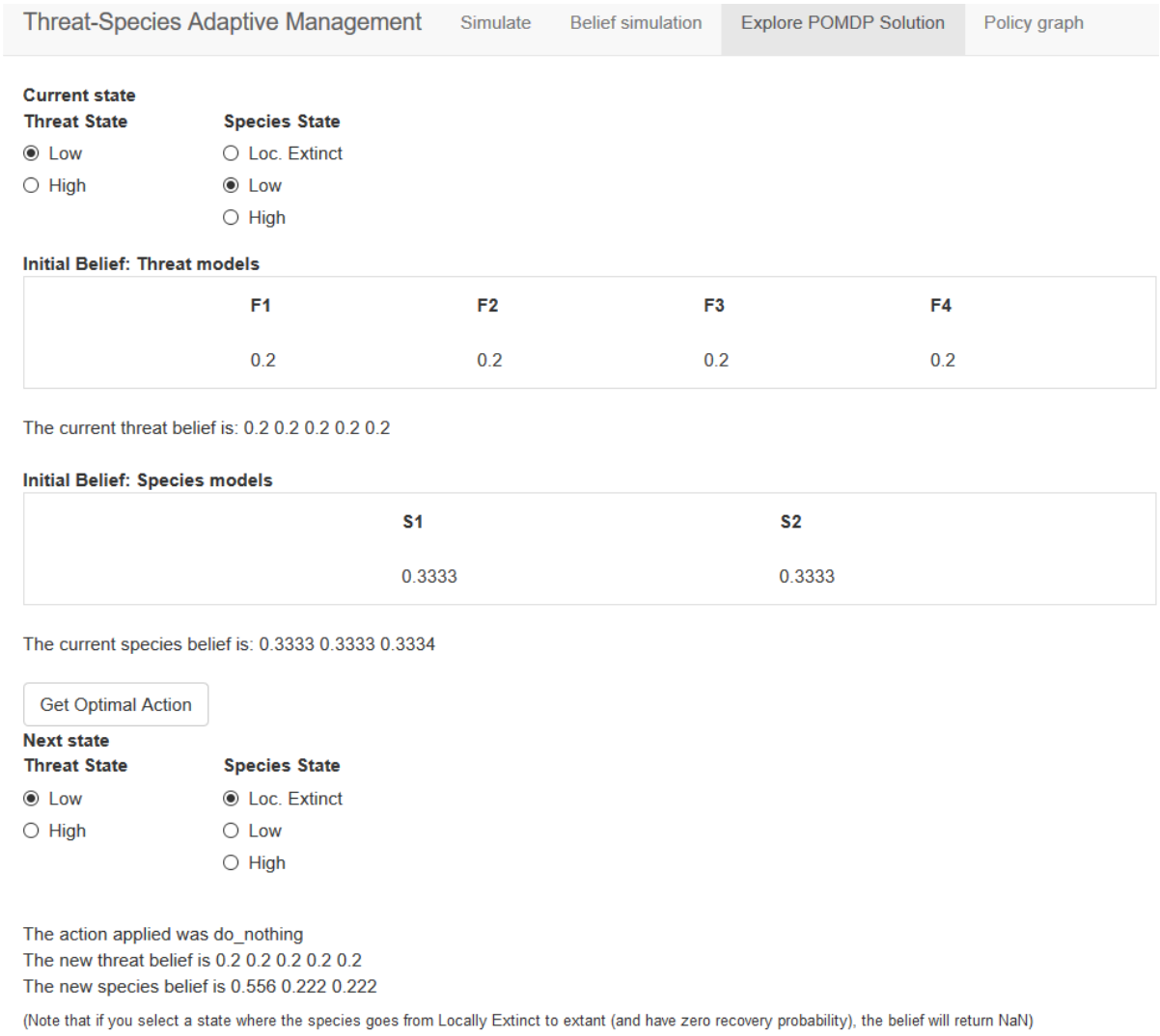


Figure 2: The Simulate Belief tab.

3. Explore POMDP Solution page

This page provides an interactive tool to simulate the belief state over one timestep. Users are given radio buttons to select the current threat and species state. Users can then input the initial belief states for the threat and species models (these have default values but they can be edited by clicking in the relevant text boxes). After doing this, pushing the "Get Optimal Action" button will interrogate the optimal policy and display the optimal action for the next timestep. Users can then specify the next observed state and the tool will print out the updated belief state (based on the current and next states and assuming that the optimal action is applied). This tool could be used sequentially by pasting the updated beliefs into the initial belief boxes and repeating the process.



4. Policy graph page

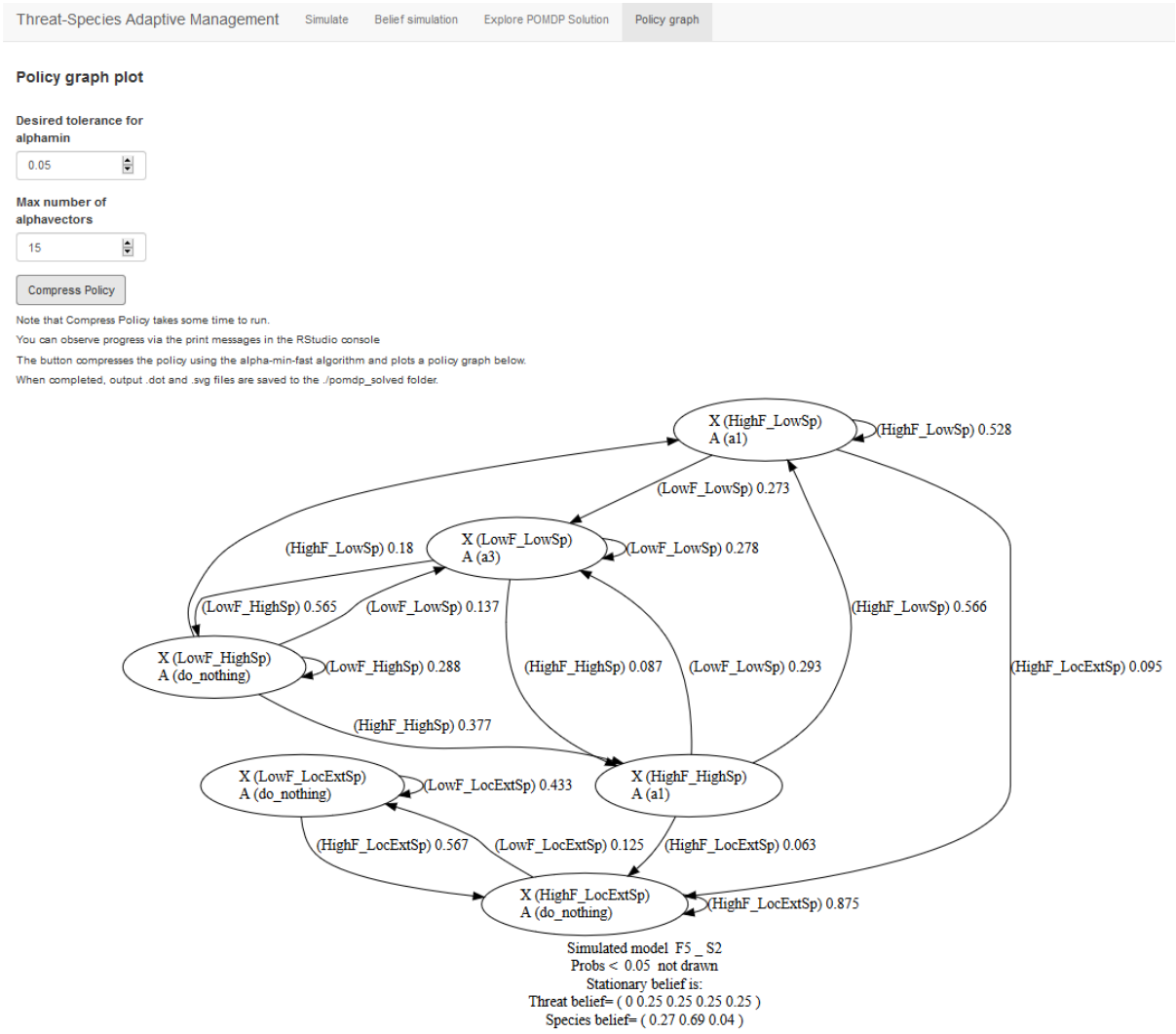
This page implements the alpha-min-fast algorithm (Dujardin et al. 2017) to compress the policy file to a desired number of alpha vectors and plot the policy graph. Users can specify the tolerance for the algorithm (lower values are better approximations to the true policy, but take longer to find) and the maximum number of alpha-vectors to include in the solution (pruning more alpha-vectors makes the solution more compact but potentially increases the error). The "Compress Policy" button runs the alpha-min-fast algorithm, which is stored in an R script named "alpha_min_fast.R". Progress messages are pasted to the console—these are useful since alpha-min-fast can run slowly on this problem and depending on how many alpha vectors are included in the policy file, it may take a substantial amount of time to complete. If you are using the alpha-min-fast page of the app, we recommend running the app through RStudio so that you can see the progress messages (if you run via the web you will have to wait until the output file appears in the pomdp_solved folder… but until the file is completed, you won't know if the algorithm will complete or is hanging because you have too many alpha-vectors). You can control the number of alpha-vectors by changing the precision of the SARSOP algorithm with the "MOMDP tolerance" option on the Setup and Simulation page. We recommend finding a precision that returns up to ~3000 alpha vectors to ensure a reasonable

solution time for alpha-min-fast (a precision of 0.5 worked well on our problem; solved in a few seconds on our laptop).

The policy graph for this problem remains complex despite the simplifications to the number of alphavectors. SARSOP contains a tool for plotting policy files using a file type called a .dot file, which is a graphviz graphical format that can be plotted using the DiagrammeR package in R. Unfortunately, SARSOP's polgraph tool is designed to plot POMDP rather than MOMDP outputs, so the resulting policy graph remains highly complex despite the simplified policy file. This is caused by the program interpreting otherwise identical states with different beliefs as different nodes in the graph. There is a need to design a plotting tool that is customised for dealing with MOMDP policies and can identify and plot identical states. We have implemented the sarsop plotting tool in our code (lies 603-614), however it is commented out because the resulting policy graph is too complex for easy use.

As an alternative to help with visualising the policy, we created a custom visualisation of the policy graph from our simulations on the previous pages. Our belief simulation showed that for all models, a stationary belief distribution is reached in relatively few steps. The stationary belief distribution is dependent on the "true model" used in the simulations, but we can plot the optimal policy for any "true model" to help users to visualise the optimal policy under different possible models. The policy graph for the current "true model" (i.e. the one selected using the "Threat model" and "Species model" radio buttons in the "Simulate" tab) is computed and displayed when the user clicks the "Compress Policy" button (see the example below). The policy graph is saved in "./pomdp_solved/polGraph_miniFx_Sy.svg", where x and y represent the threat model and species model respectively. SVG files can be viewed in a web browser and edited with vector graphics tools.

To read the policy graph, find the node corresponding to the current state (denoted by X). The optimal action will be listed in the node (denoted A). After taking the action, the probability of transition to other states is shown adjacent to the relevant edges on the graph. The text below the graph shows the "true model" that the policy graph applies to. Transition probabilities less than 0.05 are not shown for simplicity; if desired, this threshold can be changed with the 'threshold' argument of the draw.polgraph.conditional function at line 623 of the code. The stationary belief is also displayed on the policy graph. See below for example output from the compressed policy graph page.



References

Cain J. 2001. Planning improvements in natural resource management. Guidelines for using Bayesian networks to support the planning and management of development programmes in the water sector and beyond. Wallingford, UK: Centre for Ecology and Hydrology. Report no.

Dujardin Y, Deitterich T, Chades I. 2017. Three new algorithms to solve N-POMDPs. Paper presented at Thirty-first AAAI conference on Artificial Intelligence, San Francisco, USA.