MODELING STRATEGIES FOR THE CONSERVATION OF FLORIDA MANATEES USING POMDP METHODS

AA 228 / CS 238 PROJECT

Poojit Hegde
Department of Computer Science
Stanford University University
Stanford, CA 94305
phegde8@stanford.edu

Rubens Lacouture
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
rubensl@stanford.edu

Aarya Mecwan
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
amecwan@stanford.edu

June 30, 2025

ABSTRACT

In this paper, we explore the use of partially observable Markov decision process (POMDP) models for formulating conservation strategies, particularly for Florida Manatee populations. Florida Manatees are a threatened species in the waters around the Southeastern United States, and they face various environmental threats from changes in the marine climate. Various management and intervention strategies have been used for conservation of manatees, including monitoring population size, rescuing vulnerable wild manatees, rehabilitation efforts. We model these conservation strategies as a POMDP, where there is uncertainty in the state of the manatees being threatened, endangered, or extinct, based on observations of seeing wild carcasses or wild living manatees. Based on estimated transition, reward, and observation models, we perform a policy search to optimize resource allocation for Florida manatee conservation based on educated assumptions of the kinds of observations and transition matrices associated with the problem. We compare the use of multiple offline and online methods for evaluating our POMDP problem. We find that POMDP methods are promising in designing conservation strategies for vulnerable species like Florida manatees.

Keywords partially observable Markov decision processes (POMDP) · conservation

1 Introduction

For this project, we consider the problem of resource management for determining strategies for the conservation of Florida manatees. Species conservation efforts can be resource-intensive. Various conservation tasks include monitoring the population size, rescuing vulnerable wild manatees, rehabilitating manatees, and more. Resource management tasks also have significant uncertainty, particularly in monitoring the species for whether it is endangered, threatened, etc. Counting the exact number of species is a task that comes with uncertainty. Therefore, resource management strategies that account for this uncertainty are needed. Partially observable Markov decision processes can be used to model these sequential problems with uncertainty.

1.1 Florida Manatees

The Florida manatee is a marine mammal usually dispersed in several spots near waters in the Southwestern United States during summer months, but in the winter, aggregates in Florida for the warm-water habitats. Since the 1970s, the species has undergone a large fluctuation in population size and protection statuses. In the 1970s the species was endangered, with only around a few hundred manatees, but in 2018, the population size was estimated to reach almost 9 thousand, making it threatened. Beginning 2021, The Florida Fish and Wildlife Commission (FWC) has declared an Unusual Mortality Event (UME) for the manatee species. The UME attributes multiple reasons for this, including widespread loss of seagrass as a food source, unusually cold winters due to climate change, and algal bloom outbreaks. In the long term, due to climate change and its effects on the marine ecosystem, coastal areas may not be a sustainable ecosystem for manatees for much longer.

1.2 Partially Observable Markov Decision Processes (POMDP)

Partially Observable Markov Decision Processes (POMDPs) are a type of mathematical framework used to model decision-making problems that involve uncertainty and incomplete information. In a POMDP, an agent interacts with an environment that is partially observable, meaning that the agent cannot directly observe the true state of the environment. Instead, the agent observes a set of potentially noisy and incomplete observations that are generated by the underlying state. The environment's state evolves over time according to a probabilistic process, and the agent's actions affect this process. The goal of the agent is to select a sequence of actions that maximizes a long-term objective, often referred to as the policy.

1.3 POMDPs for Conservation Strategies

To apply POMDP to conservation, we need to first define the state space, action space, observation space, transition probabilities, and rewards.

- State space: The state space should include all the relevant variables that affect the conservation problem, such as the population size of the endangered species, the quality of the habitat, and the impact of human activities.
- Action space: The action space should include all the possible actions that the decision-maker can take, such as implementing a conservation measure, reducing human activities in the area, or monitoring the population of the species.
- Observation space: The observation space should include all the information that the decision-maker can obtain from the environment, such as the number of animals spotted during a survey or the amount of damage caused by human activities.
- Transition probabilities: The transition probabilities should describe how the state of the environment changes over time in response to the actions taken by the decision-maker and other external factors.
- Rewards: The rewards should be designed to encourage conservation actions that are aligned with the goals of the conservation problem. For example, a positive reward could be given for increasing the population of the endangered species, while a negative reward could be given for allowing human activities that harm the habitat.

Using the POMDP framework, the decision-maker can use a value iteration algorithm or a policy iteration algorithm to compute an optimal policy that maximizes the expected cumulative reward over a given time horizon. The optimal policy will depend on the initial state of the environment, the available information, and the goals of the conservation problem.

1.4 Related Work

Formulating conservation of strategies using POMDP models is not new, previous work has shown that this technique can be applied to various conservation strategy formulation which has gained popularity in recent years. Nicol and Chadès [2012] make use of a Continuous U-Tree (CU-Tree) to formulate optimal policies for the management of sea otters in Washington. Chadès et al. [2008] and McDonald-Madden et al. [2011] apply a POMDP fomulation to formulate the conservation strategy for managing endangered Sumatran tigers and uses an analytical approximation for the POMDP solution. In this project, we aim to apply online planning methods and offline planning methods to obtain optimal policies for the conservation of manatees problem.

Overall, the use of POMDPs in conservation strategy formulation has shown promise for addressing complex and uncertain decision-making problems. As far as we know, we did not find any previous work in the literature that

attempted to aimed to model the conservation of manatees problem using a POMDP formulation. Although our topic and problem is novel, we make use of the techniques used in prior work in the literature to solve a variation of a conservation strategy formulation. We loosely followed the proposed population model by Hostetler et al. [2021] with some modifications to represent our own population model.

2 Methods

Our problem is formulated as a POMDP, partially observable Markov Decision Process, meaning that we only have partial knowledge of the current state, since it is difficult to detect each animal.

States: [extinct, endangered, threatened]

Actions: [do nothing, survey, rescue, rehabilitate].

Observations: [see nothing, wild carcass, wild living]

Our initial state was [threatened], and the **discount factor** was 0.9.

In actions, *rescue* refers to bringing manatees that are sick or in danger to a facility for treatment. *Rehabilitate* refers to performing tasks to restore health in manatees, such as feeding them romaine lettuce. *Survey* refers to determining the manatee population, such as with aerial and water-based survey methods. *Doing nothing* refers to doing nothing.

In observations, the most common one is not seeing any species unless the action is the survey. There's also a chance to see a dead carcass or wild living animal. A wild carcass reflects a higher likelihood of a manatee being extent or endangered while seeing a wild living manatee reflects a lower likelihood of extinction or endangerment.

Risks are giving up too soon on possibly locally extinct species or spending too much resources on species that are not engangered or extinct.

T(s' | s, a)	Extinct	Endangered	Threatened
Extinct	1	0	0
Endangered, do nothing	0.2	0.8	0
Endangered, survey	0.2	0.8	0
Endangered, rescue	0.01	0.49	0.5
Endangered, rehabilitate	0.01	0.59	0.4
Threatened, do nothing	0	0.25	0.75
Threatened, survey	0	0.25	0.75
Threatened, rescue	0	0.15	0.85
Threatened, rehabilitate	0	0.05	0.95

Table 1: **Transition Matrix**

O(s' | s, a)	Seen nothing	Wild carcass	Wild living
Extinct	1	0	0
Endangered, do nothing	0.99	0.01	0
Endangered, survey	0	0.9	0.1
Endangered, rescue	0.9	0.1	0
Endangered, rehabilitate	0.99	0.01	0
Threatened, do nothing	0.99	0.01	0
Threatened, survey	0	0.1	0.9
Threatened, rescue	0.9	0	0.1
Threatened, rehabilitate	0.99	0	0.01

Table 2: **Observation Matrix**

2.1 Offline Solutions

2.1.1 QMDP

QMDP is a simple offline approximation technique, that assumes perfect observability and involves both a fully observed MDP and the action value function, Q. A set of Γ vectors are iteratively updated, and after k iterations, and the resulting

s, a	R(s, a)	s,a	R(s,a)
Extinct, do nothing	-201	Endangered, rescue	-170
Extinct, survey	-210	Endangered, rehabilitate	-170
Extinct, rescue	-270	Threatened, do nothing	-1
Extinct, rehabilitate	-270	Threatened, survey	-10
Endangered, do nothing	-101	Threatened, rescue	-70
Endangered, survey	-110	Threatened, rehabilitate	-70

Table 3: **Reward Matrix**

Γ represents a value function and policy. This result can then be used with one-step look ahead. The policy will be an approximation of the optimal solution.

The QDMP algorithm constructs an alpha vector α_a for each action, is initialized to zero and then iterated with the following equation:

$$\alpha_a^{(k+1)}(s) = R(s,a) + \gamma \sum_{s'} T(s'|s,a) \max_{a'} \alpha_{a'}^{(k)}(s')$$

2.1.2 Fast Informed Bound

Similarly to QMDP, Fast Informed Bound (FIB) is an offline technique that computes an alpha vector α_a for each action. The main difference is that it also takes the observation into account. The new iteration rule is:

$$\alpha_a^{(k+1)}(s) = R(s,a) + \gamma \sum_{o} \max_{a'} O(o|a,s') T(s'|s,a) \alpha_{a'}^{(k)}(s')$$

The FIB provides an upper bound for the optimal policy that is often tighter than QMDP.

2.2 Belief Update

Since policies rely on belief states, when evaluating policies in real-time based on observations received and actions taken, a belief updater can be used. We use a type of inference known as recursive Bayesian estimation that updates the belief state based on observation received and action taken. Since there are a discrete number of states, actions, and observations, we can use a discrete state filter. Given the previous belief b, and an agent that takes an action a and receives an observation o, the new belief b' can be calculated as

$$b'(s') = O(o|a,s') \sum_{s} T(s'|s,a) b(s).$$

2.3 Online Solutions

2.3.1 Monte Carlo Tree Search

We also employ an online solution method to evaluate our POMDP known as Partially Observable Upper Confidence Tree (PO-UCT). This method, described by Silver and Veness [2010] explores Partially Observable Monte Carlo Planning (POMCP) methods, and combines more conventional Monte Carlo Tree Search (MCTS) for partial observability with Upper Confidence Trees (UCT).

Monte Carlo Tree Search with partial observability involves a belief state *b*, depth *d*, exploration factor *c*, and rollout policy π. Due to the uncertainty, this algorithm utilizes histories instead of states, a history $h = a_1 o_2 a_2 o_2 \ldots$ being a sequence of past actions and observations.

In the POMCP algorithm we use, first Monte-Carlo sampling is used during belief state updates and planning. Here, instead of generating a mapping from all belief states to actions, we iteratively decide the next action from a given belief state during execution. POMCP is better at planning effectively for significantly large POMDPs, since the entire mapping of beliefs to actions is not necessary to construct. When searching down the tree, the algorithm takes the decision that maximizes the value estimate $Q(h,a)$ as follows:

$$Q(h,a) + c\sqrt{\frac{\log N(h)}{N(h,a)}}.$$

In this case, $N(h) = \sum_a N(h, a)$ is the total visit count for *h* and *c* is an exploration parameter. Upper confidence trees (UCT) are used to balance exploitation and exploration, with at each step of exploration, child node j is chosen as the one that maximizes

$$UCT_j = X_j + C * \sqrt{\ln(n)/n_j}.$$

Altogether, these two algorithms constitute PO-UCT.

3 Results and Discussion

For this problem, we created policies for both offline and online POMDP solvers.

Figure 1 depicts the alpha vectors for QMDP. The x-axis represents the probability of the state being *endangered*, given that the probability of the state being *extinct* is zero. That is, the state is either *endangered* or *threatened*. Removing the possibility of extinction makes it possible to model the alpha vectors for this decision problem on a two-dimensional Cartesian plane; in addition, it serves as a useful reduction to the conservation strategy of preventing endangerment. The y-axis refers to the utility function $U(b)$, where b is the belief state. For example, if $p(endangered) = 0.5$, then $b = [0, 0.5, 0.5]$, because $p(extinct) = 0$ and $p(threatened) = 1 - p(endangered) = 0.5$.

Each alpha vector is associated with an action. For this plot, the alpha vector associated with the *do nothing* action is excluded, as when $p(extinct) = 0$, it is only dominant for $p(threatened) = 1$. Therefore, for clarity, the alpha vector corresponding to this was removed.

For the QDMP policy, we can determine the optimal action to take by identifying the alpha vector with the highest utility for each $p(endangered)$ value. With accuracy to three decimals, the alpha vector corresponding to the *survey* action dominates when $p(endangered) \leq 0.071$, the alpha vector corresponding to the *rehabilitate*action dominates when $0.071 < p(endangered) \leq 0.500$, and the alpha vector corresponding to the "rescue" action dominates when $p(endangered) > 0.500$.

The optimal policy with FIB is similar, but with different bounds for which vectors are dominant. Here, the alpha vector corresponding to the *survey* action dominates when $p(endangered) \leq 0.095$, the alpha vector corresponding to the *rehabilitate* action dominates when $0.095 < p(endangered) \leq 0.566$, and the alpha vector corresponding to the *rescue* action dominates when $p(endangered) > 0.566$.

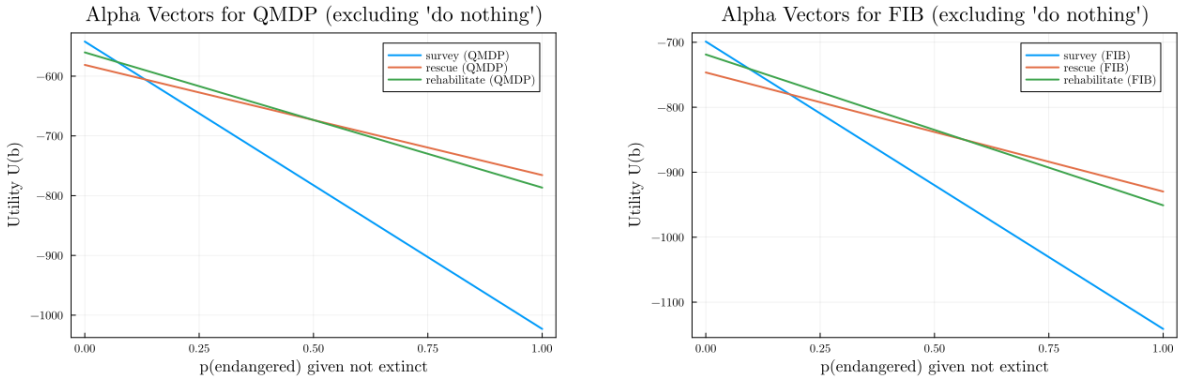

Figure 1: Graphs of the three dominant alpha vectors for QMDP and FIB Offline Policies

We also have a ternary graph with the optimal policy decision based on the full belief state. The ternary graphs in Figure 2 depict the optimal policy with the different regions corresponding to the different actions where the corresponding alpha vectors dominate. Here, the full belief state considers the probabilities of being extinct, endangered, and threatened. As seen from the graph, when the belief state probability of extinction is lower, then the distribution of actions resembles that in the previous graphs. However, when the probability is extinction is higher, then the survey action is dominant for more values.

For both of the offline methods, a discrete state filter as described in Section 3.3 can be used as an inference method to update the belief state. Then, the ternary graph can be successively used to determine the next action for an optimal policy.

Comparing the two ternary graphs for optimal policy between QMDP and FIB, we see that the regions of action are similar, but with FIB preferring *survey* and *rehabilitate* slightly more frequently than QMDP. In addition, while

this graph doesn't depict this nuance, there are some extraneous regions in the ternary graph, particularly those for which $p(extinct) > p(endangered) < p(threatened)$. Logically, this combination of probabilities makes less sense considering the context of the problem, so some regions of the ternary graph are likely to not be actually used when updating the belief state. While it is not visible in this graph, the action *do nothing* is the optimal action for some actions, particularly $[0, 0, 1]$ and $[1, 1, 0]$. That is, upon the certainty of the manatees being threatened or extinct, we do nothing.

Figure 2: Ternary Graphs for QMDP and FIB Offline Policies

For the online policy PO-UCT, which combines partially observable MCTS with UCT, we have a preliminary component of a sample tree generated. Here, Monte Carlo simulation methods identify the optimal child node given each observation. In this example shown in Figure 3, we start with the action "rescue", and we observe "seen nothing". Then, we take the action "rescue" again, and we observe "seen nothing" again. Last, we take the action "do nothing". The value function's outputs are also visible on the treee as V. A benefit of this method is that it is less computationally expensive to evaluate the policy online. However, it is more difficult to get a bigger picture of how the policy treats different belief states, as the offline policies do since they allow the creation of the graphs visualizing alpha vectors and optimal policies with the ternary graphs.

4 Conclusion

In this project, we modeled the conservation strategies for the Florida Manatee populations as a POMDP and solved it using different online and offline planning methods. For the offline methods of QMDP and FIB, we demonstrated how the policy in terms of the dominant alpha vectors as well as in a ternary graph can be useful for understanding how a policy maps the belief state to actions. For the online method, we demonstrated how a MCTS-based method can be used to construct a policy for Mantee conservation.

The particular details of our policies, in terms of their bounds and specific numbers, are a result of our problem formulation. In particular, the values we choose for our transition function, observation function, and reward function highly impact the result of both the offline and online methods. For the purposes of this problem, we chose the values we have based on rough estimates how how impactful different actions are in changing the state of Florida manatees, what they reveal about observations, and how costly they are. Consequently, limitations of the policies we constructed can be improved through more data on the actual effects of various actions. In addition, the POMDP model as a whole can be expanded to incorporate more meaningful states, actions, and observations that can more precisely reflect the challenges of Florida manatee conservation today.

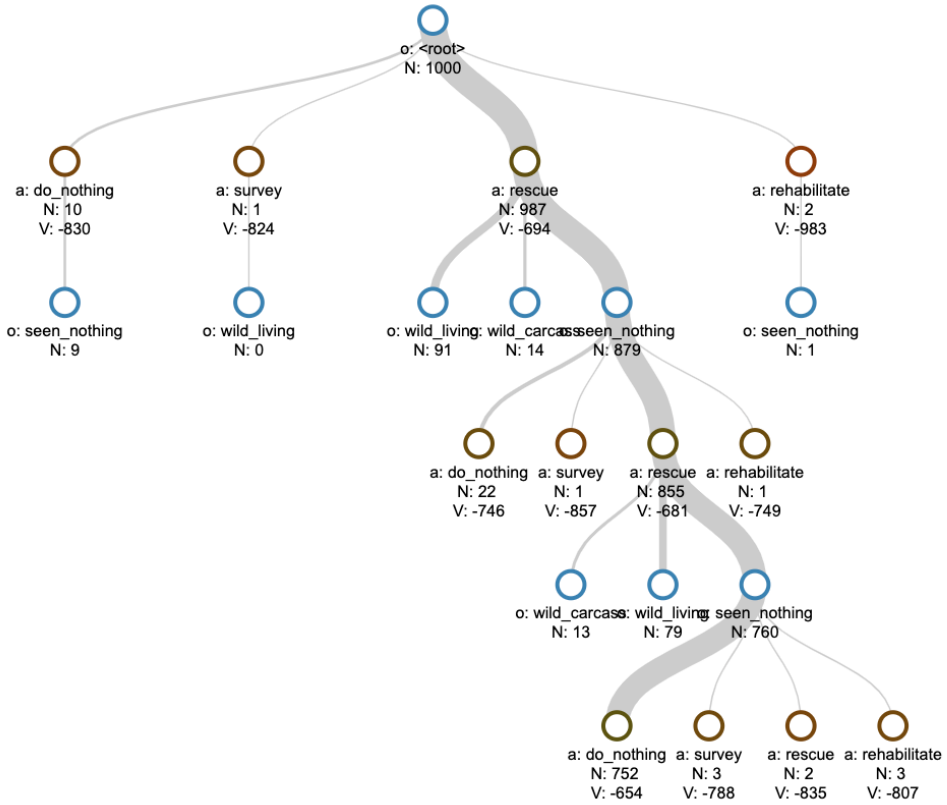

Figure 3: Tree for PO-UCT

5 Contributions

Aarya initiated the codebase and contributed to the development of the POMDP structure. Rubens was responsible for coding the POMDP formulation and provided valuable assistance with the literature review. Poojit significantly contributed in coding the offline and online methods for evaluating the POMDP, collecting and visualizing the results, and summarizing the conclusions. All team members wrote the final paper and did initial research.

References

Sam Nicol and Iadine Chadès. Which states matter? an application of an intelligent discretization method to solve a continuous pomdp in conservation biology. *PLOS ONE*, 7(2):1–8, 02 2012. doi:10.1371/journal.pone.0028993. URL `https://doi.org/10.1371/journal.pone.0028993`.

Iadine Chadès, Eve McDonald-Madden, Michael A. McCarthy, Brendan Wintle, Matthew Linkie, and Hugh P. Possingham. When to stop managing or surveying cryptic threatened species. *Proceedings of the National Academy of Sciences*, 105(37):13936–13940, 2008. doi:10.1073/pnas.0805265105. URL `https://www.pnas.org/doi/abs/10.1073/pnas.0805265105`.

Eve McDonald-Madden, Iadine Chadès, Michael A McCarthy, Matthew Linkie, and Hugh P Possingham. Allocating conservation resources between areas where persistence of a species is uncertain. *Ecological Applications*, 21(3): 844–858, 2011.

Jeffrey A Hostetler, Julien Martin, Michael Kosempa, Holly H Edwards, Kari A Rood, Sheri L Barton, and Michael C Runge. Reconstructing population dynamics of a threatened marine mammal using multiple data sets. *Scientific Reports*, 11(1):1–15, 2021.

David Silver and Joel Veness. Monte-carlo planning in large pomdps. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL `https://proceedings.neurips.cc/paper/2010/file/edfbe1afcf9246bb0d40eb4d8027d90f-Paper.pdf`.

A Appendix: Code

```
# CODE GOES HERE (change the language option to your language of choice)

#This project formulates the task of Florida manatee conservation as a POMDP problem.
#Libraries and Packages
using POMDPs
using POMDPModelTools
using QuickPOMDPs
using POMDPTools
using QMDP
using BasicPOMCP
using D3Trees
using POMDPModels
using POMDPSimulators
using FIB
using Plots

#POMDP Formulation

@enum State extinct endangered threatened
@enum Action do_nothing survey rescue rehabilitate
@enum Observation seen_nothing wild_carcass wild_living

pomdp = QuickPOMDP(
    states       = [extinct, endangered, threatened],  #
    actions      = [do_nothing, survey, rescue, rehabilitate],  #
    observations = [seen_nothing, wild_carcass, wild_living], #
    initialstate = [threatened],          # Deterministic initial state
    discount     = 0.9,             #

    transition = function T(s, a)
        if s == extinct
            return SparseCat([extinct, endangered, threatened], [1, 0, 0])
        elseif s == endangered && a == do_nothing
            return SparseCat([extinct, endangered, threatened], [0.2, 08, 0])
        elseif s == endangered && a == survey
            return SparseCat([extinct, endangered, threatened], [0.2, 0.8, 0])
        elseif s == endangered && a == rescue
            return SparseCat([extinct, endangered, threatened], [0.01, 0.49, 0.5])
        elseif s == endangered && a == rehabilitate
            return SparseCat([extinct, endangered, threatened], [0.01, 0.59, 0.4])
        elseif s == threatened && a == do_nothing
            return SparseCat([extinct, endangered, threatened], [0, 0.25, 0.75])
        elseif s == threatened && a == survey
            return SparseCat([extinct, endangered, threatened], [0, 0.25, 0.75])
        elseif s == threatened && a == rescue
            return SparseCat([extinct, endangered, threatened], [0, 0.15, 0.85])
        elseif s == threatened && a == rehabilitate
            return SparseCat([extinct, endangered, threatened], [0, 0.05, 0.95])
        end
    end,

    observation = function O(s, a, s)
        if s == extinct
            return SparseCat([seen_nothing, wild_carcass, wild_living], [1, 0, 0])
        elseif s == endangered && a == do_nothing
            return SparseCat([seen_nothing, wild_carcass, wild_living], [0.99, 0.01, 0])
        elseif s == endangered && a == survey
            return SparseCat([seen_nothing, wild_carcass, wild_living], [0, 0.9, 0.1])
        elseif s == endangered && a == rescue
            return SparseCat([seen_nothing, wild_carcass, wild_living], [0.9, 0.1, 0])
        elseif s == endangered && a == rehabilitate
            return SparseCat([seen_nothing, wild_carcass, wild_living], [0.99, 0.01, 0])
```

```
        elseif s == threatened && a == do_nothing
            return SparseCat([seen_nothing, wild_carcass, wild_living], [0.99, 0, 0.01])
        elseif s == threatened && a == survey
            return SparseCat([seen_nothing, wild_carcass, wild_living], [0, 0.1, 0.9])
        elseif s == threatened && a == rescue
            return SparseCat([seen_nothing, wild_carcass, wild_living], [0.9, 0, 0.1])
        elseif s == threatened && a == rehabilitate
            return SparseCat([seen_nothing, wild_carcass, wild_living], [0.99, 0, 0.01])
        end
    end,

    reward = function R(s, a)
        if s == extinct && a == do_nothing
            return -201
        elseif s == extinct && a == survey
            return -210
        elseif s == extinct && a == rescue
            return -270
        elseif s == extinct && a == rehabilitate
            return -270
        elseif s == endangered && a == do_nothing
            return -101
        elseif s == endangered && a == survey
            return -110
        elseif s == endangered && a == rescue
            return -170
        elseif s == endangered && a == rehabilitate
            return -170
        elseif s == threatened && a == do_nothing
            return -1
        elseif s == threatened && a == survey
            return -10
        elseif s == threatened && a == rescue
            return -70
        elseif s == threatened && a == rehabilitate
            return -70
        end
    end,
)

# Experimenting with Belief Updater as a Bayesian filter

updater(pomdp::QuickPOMDP) = DiscreteUpdater(pomdp);
b0 = uniform_belief(pomdp); b0.b

for i in 0:10
    observation_choices = [seen_nothing, wild_carcass, wild_living]
    o1 = rand(observation_choices)
    a1 = rehabilitate
    o1 = seen_nothing
    b1 = update(updater(pomdp), b0, a1, o1)
    b1.b
end

# Solve POMDP using QMDP Offline Method
qmdp_solver = QMDPSolver(max_iterations = 100)
qmdp_policy = solve(qmdp_solver, pomdp)

# Check action that QMDP policy takes for various belief states
for i in 0:10
    # Inner For-loop
    for j in 0:10-i
        b = [i/10, j/10, (10-i-j)/10]
```

```
        a = action(qmdp_policy, b)
        if i <= j || 10-i-j <= j
            println(b,a)
        end
    end
end

begin
    # Alpha Vectors for QMDP, where extinct belief is 0

    default(fontfamily="Computer Modern", framestyle=:box)
    p_extinct = 0

    #print(alphavectors(qmdp_policy))

    alpha_do_nothing = alphavectors(qmdp_policy)[1]
    alpha_survey = alphavectors(qmdp_policy)[2]
    alpha_rescue = alphavectors(qmdp_policy)[3]
    alpha_rehabilitate = alphavectors(qmdp_policy)[4]
    print(alphavectors(qmdp_policy))
    x = range(0, 1, length=100)
    y1= @.  (alpha_do_nothing[2]-alpha_do_nothing[3]) * x + alpha_do_nothing[3]
    y2 = @. (alpha_survey[2]-alpha_survey[3]) *x + alpha_survey[3]
    y3= @.  (alpha_rescue[2]-alpha_rescue[3]) * x + alpha_rescue[3]
    y4 = @. (alpha_rehabilitate[2]-alpha_rehabilitate[3]) *x + alpha_rehabilitate[3]


    plot1 = plot(x, [y2 y3 y4 y1], title="Alpha Vectors for QMDP", xlabel="p(endangered) given
     not extinct", ylabel = "Utility U(b)", label=["survey (QMDP)" "rescue (QMDP)" "
    rehabilitate (QMDP)" "do nothing (QMDP)"], linewidth=2)
    #action(qmdp_policy, [1-p_endangered - p_threatened,p_endangered, p_threatened])
    #plot_alpha_vectors(qmdp_policy, p_hungry)
end

#Simplify Plot



plot2 = plot(x, [y2 y3 y4],
  title="Alpha Vectors for QMDP (excluding 'do nothing')",
    xlabel="p(endangered) given not extinct", ylabel = "Utility U(b)",
    label=["survey (QMDP)" "rescue (QMDP)" "rehabilitate (QMDP)"], linewidth=2)



# ymax = @. max([y1,y2,y3,y4])
# plot(x, [y2 y3 y4],
#   title="Alpha Vectors",
#     label=["survey (QMDP)" "rescue (QMDP)" "rehabilitate (QMDP)"], linewidth=2)

#Solve POMDP using fast informed bound (FIB) Offline Method

fib_solver = FIBSolver(max_iterations=100)
fib_policy = solve(fib_solver, pomdp)

# Check action that FIB policy takes for various belief states


for i in 0:10
    # Inner For-loop
    for j in 0:10-i
        b = [i/10, j/10, (10-i-j)/10]
        a = action(fib_policy, b)
```

```
        if i <= j || 10-i-j <= j
            println(b,a)
        end
    end
end

begin
    # Alpha Vectors for FIB, where extinct belief is 0

    default(fontfamily="Computer Modern", framestyle=:box)
    p_extinct = 0

    #print(alphavectors(qmdp_policy))

    alpha_do_nothing_fib = alphavectors(fib_policy)[1]
    alpha_survey_fib = alphavectors(fib_policy)[2]
    alpha_rescue_fib = alphavectors(fib_policy)[3]
    alpha_rehabilitate_fib = alphavectors(fib_policy)[4]
    print(alphavectors(fib_policy))
    x_fib = range(0, 1, length=100)
    y1_fib = @.  (alpha_do_nothing_fib[2]-alpha_do_nothing_fib[3]) * x + alpha_do_nothing_fib
    [3]
    y2_fib = @. (alpha_survey_fib[2]-alpha_survey_fib[3]) *x + alpha_survey_fib[3]
    y3_fib = @.  (alpha_rescue_fib[2]-alpha_rescue_fib[3]) * x + alpha_rescue_fib[3]
    y4_fib = @. (alpha_rehabilitate_fib[2]-alpha_rehabilitate_fib[3]) *x +
    alpha_rehabilitate_fib[3]


    plot3 = plot(x_fib, [y2_fib y3_fib y4_fib y1_fib], title="Alpha Vectors for FIB", xlabel="
    p(endangered) given not extinct", ylabel = "Utility U(b)", label=["survey (FIB)" "rescue
    (FIB)" "rehabilitate (FIB)" "do nothing (FIB)"], linewidth=2)
    #action(qmdp_policy, [1-p_endangered - p_threatened,p_endangered, p_threatened])
    #plot_alpha_vectors(qmdp_policy, p_hungry)

end

#simplified FIB plot

plot4 = plot(x_fib, [y2_fib y3_fib y4_fib], title="Alpha Vectors for FIB (excluding 'do
    nothing')", xlabel="p(endangered) given not extinct", ylabel = "Utility U(b)", label=["
    survey (FIB)" "rescue (FIB)" "rehabilitate (FIB)"], linewidth=2)
    #action(qmdp_policy, [1-p_endangered - p_threatened,p_endangered, p_threatened])
    #plot_alpha_vectors(qmdp_policy, p_hungry)

 bound1_1 = (alpha_survey[3]-alpha_rehabilitate[3])/(alpha_rehabilitate[2]-alpha_rehabilitate
    [3]-alpha_survey[2]+alpha_survey[3])
println(bound1_1)
bound2_1 = (alpha_rescue[3]-alpha_rehabilitate[3])/(alpha_rehabilitate[2]-alpha_rehabilitate
    [3]-alpha_rescue[2]+alpha_rescue[3])
println(bound2_1)
function optimal_qmdp(x)
    if x >= 0 && x < bound1_1
        (alpha_survey[2]-alpha_survey[3]) *x + alpha_survey[3]
    elseif x >= bound1_1 && x < bound2_1
        (alpha_rehabilitate[2]-alpha_rehabilitate[3]) *x + alpha_rehabilitate[3]
    else
        (alpha_rescue[2]-alpha_rescue[3]) * x + alpha_rescue[3]
    end
end
```

```
bound1_2 = (alpha_survey_fib[3]-alpha_rehabilitate_fib[3])/(alpha_rehabilitate_fib[2]-
    alpha_rehabilitate_fib[3]-alpha_survey_fib[2]+alpha_survey_fib[3])
println(bound1_2)
bound2_2 = (alpha_rescue_fib[3]-alpha_rehabilitate_fib[3])/(alpha_rehabilitate_fib[2]-
    alpha_rehabilitate_fib[3]-alpha_rescue_fib[2]+alpha_rescue_fib[3])
println(bound2_2)
function optimal_fib(x)
    if x >= 0 && x < bound1_2
        (alpha_survey_fib[2]-alpha_survey_fib[3]) *x + alpha_survey_fib[3]
    elseif x >= bound1_2 && x < bound2_2
        (alpha_rehabilitate_fib[2]-alpha_rehabilitate_fib[3]) *x + alpha_rehabilitate_fib[3]
    else
        (alpha_rescue_fib[2]-alpha_rescue_fib[3]) * x + alpha_rescue_fib[3]
    end
end
#plot(x_fib, [y2_fib y3_fib y4_fib], title="Alpha Vectors for FIB", xlabel="p(endangered)
    given not extinct", ylabel = "Utility U(b)", label=["survey (FIB)" "rescue (FIB)" "
    rehabilitate (FIB)"], linewidth=2)

plot5 = plot(x, [optimal_qmdp, optimal_fib], title="Optimal Value Functions",
    xlabel="p(endangered) given not extinct", ylabel = "Utility U(b)", linewidth=3, label = ["
    QMDP" "FIB"])

# Solve POMDP using partially observable upper confidence tree (PO-UCT) online tree search
    algorithm, a Monte Carlo method

pomcp_solver = POMCPSolver()
pomcp_planner = solve(pomcp_solver, pomdp);

# Visualize beginning of tree that is constructed
a, info = action_info(pomcp_planner, initialstate(pomdp), tree_in_info=true); a
tree = D3Tree(info[:tree], init_expand=2)

# Step through decisions that were made through POMCP planner, through states, actions, and
    observations

for (s, a, o) in stepthrough(pomdp, pomcp_planner, "s,a,o", max_steps=10)
    println("State was $s,")
    println("action $a was taken,")
    println("and observation $o was received.\n")
end
```