Human-Machine Collaboration for Online Target Classification

Eli Kravitz
University of Colorado Boulder
Smead Department of Aerospace Engineering Sciences
eli.kravitz@colorado.edu

***Abstract*— Real-time characterization of threats is an important task in the realm of missile defense. Machine learning algorithms are a valuable tool for target classification, but often lack contextual knowledge in a complicated and evolving environment. Humans are equipped with context that is difficult to program into machine learning models, but are limited in their ability to monitor and classify many potential targets simultaneously. This work presents a solution to this problem by using a Hidden Markov Model (HMM) to classify targets given data from sensors, while using a Partially Observable Markov Decision Process (POMDP) framework to query a human operator when the machine learning algorithm is uncertain in its classification. The "hard data" from sensors is then fused with "soft data" from human operators to update target classification probabilities. Using these paradigms, human workload is limited while maintaining access to soft data.**

I. Introduction

The US Military possesses a constellation of eight Overhead Persistent Infrared (OPIR) satellites whose mission is to maintain surveillance on targets of interest. In this context, surveillance generally entails detection, tracking and object classification. This system is able to start and maintain tracks autonomously, but requires significant user input for object classification. As such, achieving the full capability of this system entails considerable effort from highly-trained operators.

The general workflow is that personnel in the OPIR Battlespace Awareness Center (OBAC) interrogate tracks before making a making a classification and releasing these Track IDs downstream to subsequent operators. During this process, users are able to consult with an online classification system that is updated without human input. The classifier is ad hoc in nature (it was specifically developed for this application) and is not probabilistic (it simply reports the most likely target type). Upon achieving a desired amount of certainty regarding the object type, tracks are manually released. This current workflow poses two significant obstacles: the algorithms for characterization lack both sophistication and human visibility/input (algorithms are not aware of general battlespace context that humans are privy to), and the user is tasked with intimate interaction with every track. This workload is manageable during normal operations, but can become overly cumbersome as visible IR signatures become more abundant across the globe.

Due to the current limitations of the classification process, the University of Colorado's Cooperative Human-Robot Intelligence (COHRINT) Lab is under contract to update the characterization workflow. This work presents a candidate solution that allows for better incorporation of contextual information known to humans (but not computer-based algorithms), while also limiting the load of tasking that operators are currently subject to. The proposed solution involves object discrimination using a Hidden Markov Model (HMM) framework. A Partially Observable Markov Decision Process (POMDP) is solved every time the HMM provides belief update to the state to determine whether to wait for more data, query an operator, or release a track downstream. In the event that the algorithm determines that the best action is to query a user, a simple Bayesian fusion step allows the human input to be integrated into the probabilistic model. This feature allows backend algorithms to incorporate "soft data" without being explicitly aware of the evolving battlespace.

Due to the classified nature of real-world track data, this paper will show the validity of the proposed approach on six illustrative models: Spiderman, Venom, Thor, Loki, IronMan, and Hawkeye. The structure of the developed software platform is quite modular, and will thus scale well as more (and real) target types are added in a classified environment. Integration with real-world data will not require a full-scale overhaul, but rather training of new models for the HMM and reward weight tuning for the POMDP.

II. Background & Related Work

This section presents relevant background concerning the formulation of the problem. Specifically, HMMs and POMDPs are discussed in detail.

A. Hidden Markov Models

An HMM represents a hidden stochastic process that can be inferred, with some statistical certainty, by an observable process (i.e. we may observe the weather, and subsequently make an inference as to whether the beach will be crowded). Note that for this framework to provide meaningful context, the Markov Property must hold, as described in Equation 1:

$$P(s_t \mid s_{t-1}, ..., s_1) = P(s_t \mid s_{t-1}) \tag{1}$$

Rabiner and Juang [1] provide an in-depth overview of HMMs. Their work is specifically focused on speech recognition, but the concepts readily apply to any classification-type problem. Firstly, HMMs are comprised of the following characteristics:

- L = length of observation sequence

- S = possible states
- O = possible observations
- $T(s' \mid s)$ = state transition probability
- $Z(o \mid s')$ = observation probability
- π = initial state distribution

Given an HMM definition, $\lambda = (T, Z, \pi)$, as described above, a model can be trained for each target type. Given a pre-trained model, the "forward-backward" algorithm can be used to solve for $P(O \mid \lambda)$, the probability of a string of observations given a model. The forward-backward algorithm computes posterior marginals for all hidden state variables. Through a normalization process, the probability of each classification type can be determined, as follows:

$$P(O_{1:L} \mid \lambda_i) = \frac{P(O_{1:L} \mid \lambda_i)}{\sum_j P(O_{1:L} \mid \lambda_j)},\ j \in [1, n_{models}] \quad (2)$$

A graphical model of an HMM is shown in Figure 1. Solutions reason about an entire string of observations, o, to determine the likelihood that they came from a specific model, λ.

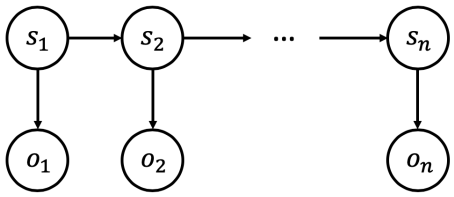

Fig. 1. Graphical diagram of HMM

While Rabiner and Juang apply this theory to a simple toy problem, Boussemart and Cummings [2] use HMMs to define and solve a realistic problem concerned with predicting the actions of an operator controlling an unmanned vehicle (UV). Using this theory, they are able to predict whether an operator is acting in an expected or unexpected manner.

It is important to tune the number of hidden states appropriately in order to not underfit the data, while also not overfitting and expending needless computation. Pal et al. [3] use a Bayesian Information Criterion (BIC) score to determine the optimal number of hidden states. In concept, a BIC score rewards fitting the observation string appropriately, while penalizing using an excessive number of hidden states. This process allows for an adequate data fit without overcomplicating the models. BIC score is given by:

$$BIC = -2logL + plog(T) \quad (3)$$

$$p = m^2 + km - 1 \quad (4)$$

In the above, $logL$ is the log likelihood of a model, m is the number of states in the Markov chain model, k is the number of parameters for the underlying distribution (two for a Gaussian), and T is the length of the observation series.

B. Partially Observable Markov Decision Processes

POMDPs provide a framework for automatically selecting optimal actions given a specific model. As with HMMs, POMDP models must obey the Markov property, described in Equation 1. Kochenderfer et al. [4] provide a succinct framework for POMDP problems:

- S = possible states
- A = possible actions
- O = possible observations
- $R(s, a)$ = reward for state action pairs
- $T(s' \mid a, s)$ = state transition probability
- $Z(o \mid a, s')$ = observation probability
- γ = discount factor

The graphical structure for a POMDP is shown in Figure 2. While the state and observation sequence has the same structure as an HMM, actions, a, and rewards, r, are present at each time step. Unlike an HMM, a POMDP reasons about the current belief about the true state in order to determine an optimal action at each time index.

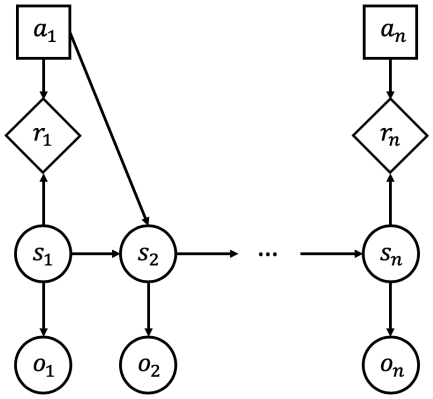

Fig. 2. Graphical diagram of POMDP

POMDPs allow for the determination of an optimal action given the model described above. Given a large state space, exact solutions are generally computationally intractable. As such, approximate solutions to POMDPs are used most often in practice. Furthermore, because optimal actions are reasoned about for a specific belief about the true state, $b(s)$, solutions over an entire belief space can become intractable even for relatively small state spaces. This problem is solved by the use of online solvers, which must only reason about beliefs that are reachable from the current belief (i.e. only a small portion of the overall POMDP is solved at a given time). Silver and Veness [5] proposed a Monte-Carlo approach to solving large POMDPs called Partially Observable Monte-Carlo Planning (POMCP). This algorithm employs a tree-based approach where similar to Monte-Carlo Tree Search (MCTS) [4] for fully-observable Markov Decision Processes (MDPs). The main deviation from MCTS is that POMCP traverses the tree based on observation/action history pairs $h(o, a)$ rather than deterministic states. POMCP uses all of the characteristics of a POMDP, along with the following:

- $N(h, a)$ = number of visits to specific node
- $Q(h, a)$ = action-value function
- d = maximum depth of tree
- m = number of Monte-Carlo simulations
- c = exploration constant
- $V(s)$ = utility

Note that d, m, and c are tuning parameters that are problem-specific and therefore require domain knowledge to be accurately chosen. The utility $V(s)$ is often initialized with some heuristic value estimate. This can be solved for using Value Iteration, a solution to the deterministic MDP that relies on the Bellman Backup Equation to solve for state utility [4].

III. Problem Statement

This problem is formulated as a classification problem. We assume that classification does not deal with raw IR data, but rather outputs from a tracker (generally a Kalman Filter or Particle Filter), where Track IDs have already been assigned. The goal of this work is to take received tracker outputs and classify targets with some probability. The work done in this paper assumes that all target types are known apriori, a necessary caveat for the validity of Equation 2.

Given the complexity of the battlespace, it is assumed that a machine-learning model with knowledge only of tracker data will lack contextual information necessary to make accurate classifications for similar target types. As such, the underlying classification algorithm must be augmented with access to information only known by expert human operators.

The goal of this work is fundamentally to attempt automated object typing, and to query a human when it is deemed that hard data alone will not lead to state certainty great enough for releasing a track. This framework is intended to keep humans in the loop while not burdening them with excessive tasking.

IV. Solution Approach

This problem can be distilled into several sub-components, as shown in Figure 3. Compartmentalizing the problem allows for both improved readability of the software, as well as modularity as new targets are added. Each of these blocks are discussed in detail as a means of creating a full description of the solution.

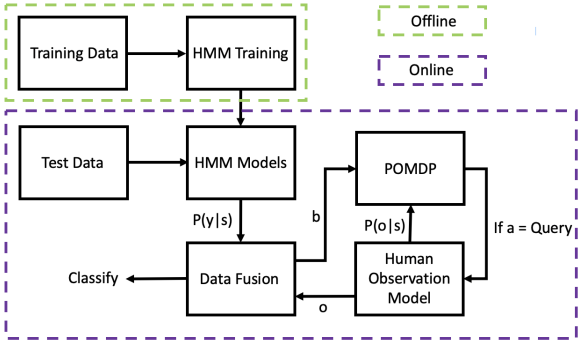

Fig. 3. System-level block diagram

A. Training Data

Training data contains all of the tracker data that is used to train the HMM models. For this proof-of-concept, each model (Spiderman, Venom, Thor, Loki, IronMan, Hawkeye) was trained with 10 tracks. This is a limited dataset, and full-scale implementations should train with more data whenever possible. The simplified dynamics are described as follows:

- Spiderman: stepwise-sine wave for varying intensity, constant velocity
- Venom: varying intensity, constant velocity
- Thor: constant intensity, stepwise-sine wave for varying velocity
- Loki: constant intensity, varying velocity
- IronMan: stepwise-sine wave for varying intensity, stepwise-sine wave for varying velocity
- Hawkeye: varying intensity, varying velocity

B. HMM Training

This block is used to create models for each target type. Through the use of BIC score, it was determined that the optimal number of hidden states for each model was 9. Training is done offline, and the models are then used along with training data to do real-time object classification. Due to the simple nature of the data, the only features trained on were velocity magnitude, velocity uncertainty, and IR intensity. The HMM models were created with a Gaussian Mixture Model (i.e. data is generated from finite number of Gaussian distributions with unknown parameters).

C. Test Data

Test data is in the same format as the training data, but is used as a means of model validation rather than model training.

D. HMM Models

The HMM models, created offline, are used online. Each time new data is received, the entire data sequence (from test data) is fed through all HMM models. Log likelihood of each model is returned, which can be converted to a probability and normalized according to Equation 2. The output of this block is the probability of a data string given a specific model $P(y|\lambda_i)$, $i \in [1, n_{models}]$.

E. POMDP

The POMDP for this problem can be defined as follows:

- S = {Spiderman, Venom, Thor, Loki, IronMan, Hawkeye, Released}
- A = {Wait, Query, Release Spiderman, Release Venom, Release Thor, Release Loki, Release IronMan, Release Hawkeye}
- O = {Spiderman, Venom, Thor, Loki, IronMan, Hawkeye, None}

- $R(s,a)$ =

$$\begin{cases} 0 & s = Released \\ 50 & a = Wait \\ -700 & a = Query \\ 500 & Release\ correctly \\ -15000 & Release\ incorrectly \end{cases}$$

- $T(s' \mid a, s)$ =

$$\begin{cases} Released & a = Release \\ s & Otherwise \end{cases}$$

- $Z(o \mid a, s')$ =

$$\begin{cases} According\ to\ human\ belief & a = Query \\ None & Otherwise \end{cases}$$

- γ = 0.6

For the POMCP solver, the following tuning parameters were used:

- d = 50
- m = 1000
- c = 1000

Note that, in this model, transitions are deterministic, and a meaningful observation is only received if the chosen action is to query the operator. Also, a small positive reward is assigned when the algorithm chooses to wait for more data. This is case because it is assumed that the HMM classifier will improve upon the belief as more data is accounted for (this is essentially assuming that the belief is monotonically increasing - a major conjecture that drastically simplifies the solver). The discount factor, γ, was chosen to be significantly less than 1 in order to prioritize increasing belief before releasing a track (this discount factor lowers the likelihood of releasing a track with a very short string of hard data).

For the POMCP solver, the utility, $V(s)$, was calculated by solving Value Iteration for the underlying MDP. This provides a heuristic estimate for the utility that leads to faster convergence of the algorithm than initializing $V(s)$ with no domain knowledge. In the solver, as the history tree is traversed, an increase in depth corresponds to an increase in length of the data sequence. As such, as the tree depth grows, the observation probability, $P(o \mid s)$, evolves according to the function described in Section IV-F. This means that the POMCP solver is explicitly aware of the changing observation probability (through the observation model) and implicitly aware of the improved HMM classification capability over time (through the positive reward associated with waiting for more data).

F. Human Observation Model

As simple human observation model, $P(o|s)$, was used in this work. We assume that the user is likely to give a correct observation 30% of the time at the first data step, improving linearly to 90% accuracy after receiving 30 data samples (and never exceeding 90% accuracy). All other observations are equally likely to be chosen by the user, with the total marginal probability of all observations adding to 1. The POMDP is aware of this model for its internal sampling of observations, and when the POMDP decides to query the operator, an observation, o, is sampled from this distribution to be used in the data fusion step. It should be noted that this is not a particularly sophisticated model, but was used in this work due to lack of domain knowledge about operator behavior.

G. Data Fusion

From the HMM model output and human observation model, $P(y_{1:n} \mid s_i)$ and $P(o \mid s_i)$ are known, respectively, at each time step. For the purposes of classification, we are interested in the quantity $P(s_i \mid y_{1:n}, o_{1:m})$, where i is the index of each target type, n is the length of the data sequence, and m is the number of human observations. This term can be solved for using Bayesian data fusion. Expressing the joint probability in terms of a chain of conditional probabilities, we can say (dropping subscripts for cleanness):

$$P(s, y, o) = P(s \mid y, o)P(y, o) \tag{5}$$

$$P(s, y, o) = P(o \mid y, s)P(y, s) \tag{6}$$

Combining, we have:

$$P(s \mid y, o)P(y, o) = P(o \mid y, s)P(y, s) \tag{7}$$

If we make the assumption that $y \perp\!\!\!\perp o \,|\, s$, we can simplify Equation 7 and rearrange to say:

$$P(s \mid y, o) = \frac{P(o \mid s)P(y \mid s)P(s)}{P(y, o)} \tag{8}$$

Above, the normalizing term, $P(y, o)$, is the same for each target type. If we make the assumption of a uniform prior for target typing, Equation 8 simplifies to:

$$P(s \mid y, o) \propto P(o \mid s)P(y \mid s) \tag{9}$$

Finally, if we make the assumption that $o_k \perp\!\!\!\perp o_1, ..., o_{k-1}, o_{k+1}, ..., o_m \mid s$, Equation 9 simplifies to (with subscripts for clarity):

$$P(s_i \mid y_{1:n}, o_{1:m}) \propto \prod_j \left[P(o_j \mid s_i)\right] P(y_{1:n} \mid s_i),\ j \in [1, m] \tag{10}$$

As stated above, both $P(o \mid s)$ and $P(y \mid s)$ are known quantities, and the left hand side of Equation 10 gives the un-normalized probability of each target type. Normalizing such that $\sum_i P(s_i|y_{1:n}, o_{1:m}) = 1$ completes the data fusion step. Note that Equation 10 specifically relies on a string of observations to fuse the probabilities. In the absence of human data, this can be simplified to:

$$P(s_i \mid y_{1:n}) \propto P(y_{1:n} \mid s_i) \tag{11}$$

H. General Workflow

In this section, the overall workflow of the online portion of the algorithm is discussed. In sequential order, the following occurs at each data step:

1) Solve for $P(y_{1:n}|x_i)$ for all models from HMM models and normalize
2) Update human belief probability given length of data sequence
3) Solve POMDP where belief is output of data fusion from previous time step
   - If a = Release, break
   - If a = Query, sample observation and add to list of observations along with observation probability at that time
   - Otherwise, do not sample observation and continue
4) Update fused probability

V. Results

This section presents simulation results for representative data. Figure 4 shows the HMM classification over time for a target whose truth is "Spiderman". It should be noted that while the HMM correctly types the target, it does not do so with strong certainty. This is because the HMM models are only trained on three features, creating poor separation between the different target types.

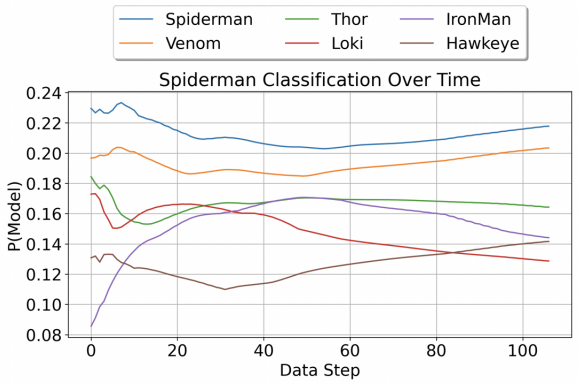

Fig. 4. HMM classification over time, in the absence of human input

Figure 5 shows classification over time of "Spiderman" (top), along with the associated actions chosen by the POMDP (bottom). It can be seen that an incorrect observation is given at the second data step, but due to the relatively low probability of the operator correctly classifying the target, the probability for "Hawkeye" only jumps slightly. Correct observations at data steps 5, 9, and 10, where the operator was more likely to answer correctly, causes the POMDP to release the track at the eleventh data step with a high classification likelihood.

Due to the incomplete nature of the data the HMM models were trained on, a more comprehensive evaluation of the POMDP behavior was investigated by solving for the optimal action over the entire belief space. Note that this analysis

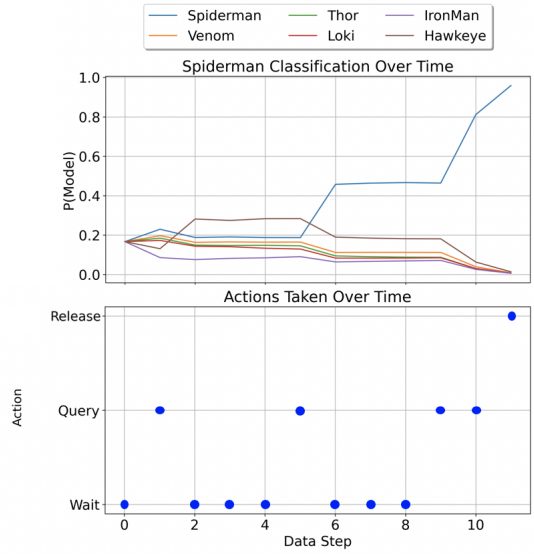

Fig. 5. Spiderman classification over time, along with associated actions

was only completed as chosen action as a function of the belief about the target being of type "Spiderman". This is a sufficient analysis due to the symmetric nature of the human observation model (i.e. similar results would be expected for all targets).

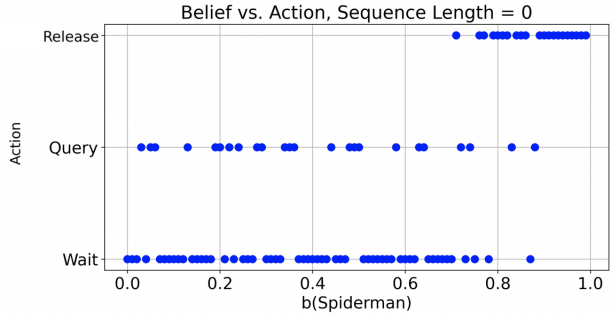

Fig. 6. Chosen actions vs. belief for data sequence length = 0, probability of correct observation = 30%

Figure 6 shows the chosen action vs. belief given that the data sequence is of length 0, indicating that the operator has a 30% chance of choosing the correct target type if queried at that time. Figure 7 shows the same results given a data sequence length of 50, indicating that the operator has a 90% chance of choosing the correct target type if queried at that time. In both cases, as the belief exceeds 80%, the POMDP is likely to release the track to downstream users. This behavior is expected based on the reward function, and is the desired system functionality. When $b(Spiderman) < 0.8$, there is a distinct difference in the POMDP behavior between the two plots. In Figure 6, the most frequent action choice across the belief space (excluding the region with a strong belief) is to "Wait". This makes intuitive sense, since the information gathering "Query" action returns observations with poor certainty. As such, POMCP assigns high utility

to gathering more hard data rather than asking an operator to provide unreliable soft data. On the contrary, in Figure 7, the most frequent action across the belief space (again excluding the region with a strong belief) is to "Query". This makes sense given the high likelihood of gathering correct observations due to the longer data sequence. These two examples were shown to present the upper and lower bounds on human observation probability, while intermediate data sequence lengths exhibit behavior between these two edge cases.

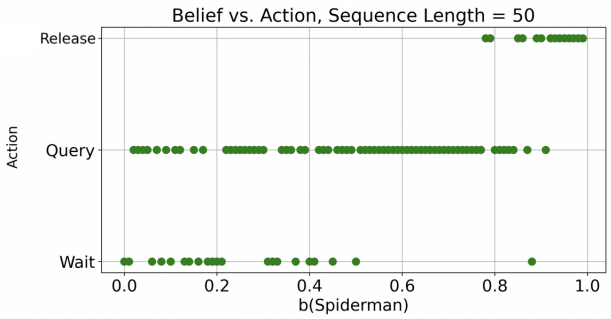

Fig. 7. Chosen actions vs. belief for data sequence length = 50, probability of correct observation = 90%

VI. Conclusions & Future Work

This paper presents a means of automated object classification with the ability to incorporate soft human data in situations where the automation is insufficient. This framework required HMM models for each target type, a POMDP model for decision making, a human observation model, and Bayesian data fusion to combine state probabilities from both machine and human sources.

Future work will include fine-tuning many of the blocks outlined in Figure 3. Firstly, the HMM models will be trained accounting for more features in order to achieve better separation for different target types. The POMDP model will also be updated to meet mission CONOPS when the intended system functionality is discussed in more detail (this may include running the POMDP every k steps to limit operator tasking). Also, due to this work being completed in a simulation environment, the algorithm assumes that the human operator will respond immediately upon being queried. This expectation must be relaxed in future iterations of this problem. Finally, a simple piece-wise linear function was used to determine human accuracy. This is likely not representative of true operator inference ability, and was used simply as a proof-of-concept. Moving forward, a more sophisticated model will be used to determine human accuracy, which may be different across a wide array of users.

VII. Contributions

This work was completed as an individual project. However, as it is part of my research, contributions were made by members of my lab. Nicholas Conlon contributed to the parsing of data files, and Hunter Ray contributed to the framework for the HMM. I worked on state optimization for the HMM, and created all other blocks in Figure 3. Note that all code was implemented in Python. Given the available libraries, the HMM was implemented using open-source code, but everything related the the POMDP was written from scratch.

VIII. Release

The authors grant permission for this report to be posted publicly.

References

[1] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. ieee assp magazine, 3(1):4–16, 1986.

[2] Yves Boussemart and ML Cummings. Behavioral recognition and prediction of an operator supervising multiple heterogeneous unmanned vehicles. Humans operating unmanned systems, 2008.

[3] Pal, et. al (2014). Modeling Winter Rainfall in Northwest India using a Hidden Markov Model: Understanding Occurrence of Different States and their Dynamical Connections. Climate Dynamics. 44. 10.1007/s00382-014-2178-5.

[4] Kochenderfer, Mykel, et al. Algorithms for Decision Making, MIT PRESS, S.l., 2022.

[5] D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In Advances in Neural Information Processing Systems (NIPS). 2010.