# Solving POMDP for the Detection of Golden Cheeked Warbler using RBF-based Q-learning

Long Le
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

***Abstract*—Viterbi decoding remains the best (performance-wise) inference technique known today. However, it requires that all data being present and only runs in batch mode. Therefore it is only suitable for offline processing. A Bayesian filter, on the other hand, sequentially processes data and hence can perform inference online, but still requires all data samples. This paper investigates a different technique that is designed to both process data sequentially and control the usage of data samples optimally. Such techniques is based on the popular framework for sequential decision making called Partially Observable Markov Decision Process (POMDP). The POMDP solution technique of choice was Q-learning using radial basis function (RBF) neural network. The application is the detection of an endangered bird species named Golden Cheeked Warbler (GCW). Our result illustrated that it is possible to use only 9.18% of the samples on average while incurring only 9.07% detection error rate, compared to a full Viterbi decoding.**

## I. Introduction

The application under consideration for this paper is the detection of Golden Cheeked Warblers (GCW). GCW is an endangered bird species [1] and hence information about their behavior will be crucial for bird scientists to develop preservation strategy. Some of the GCW's behavioral information can be inferred from their acoustic data [2]. Figure 1 shows samples of the recorded acoustic data, represented in spectrogram, at multiple time scales. The first plot from the top has the broadest time scale with the shaded region indicates the time when the GCW is present. During its presence, a GCW does not call all the time but instead rests between calls, as illustrated in the second plot. Finally, the actual time-frequency structure of a GCW's call is shown in the last plot. Therefore, it is possible to infer whether a GCW is in absent, calling or resting state merely from the acoustic data.

Since the amount of data is large, it is desirable to automate the inference process. There are various automatic inference techniques such as Viterbi decoding and Bayesian filtering [3]. Viterbi decoding remains the best (performance-wise) inference technique known today. However, it requires that all data being present and only runs in batch mode. Therefore it is only suitable for offline processing. A Bayesian filter, on the other hand, sequentially processes data and hence can perform inference online, but still requires all data samples. This paper investigates a different technique that is designed to both process data sequentially and control the usage of data samples optimally. The last technique is based on a popular framework for sequential decision making called Partially Observable Markov Decision Process (POMDP). A brief review of the

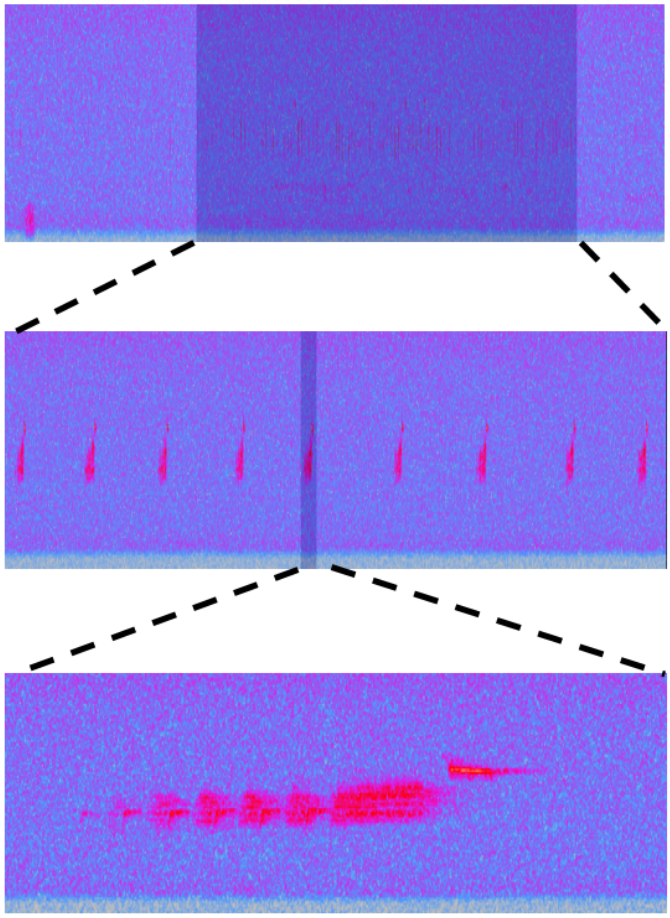

Fig. 1. Sample acoustic data represented using time-frequency analysis

POMDP framework is given in Sections II.

The reason for which the POMDP framework is chosen over other sequential decision making framework such as recurrent neural net [4] and sequential support vector machine [5] is because POMDP has a strong mathematical background that was designed specifically to solve sequential decision making problems. On the contrary, recurrent neural network is based originally on neural networks, which is a formalism for function approximation. Similarly, support vector machines was originally formulated to solve linear classification problems.

While it is possible to apply the POMDP framework for a complete state space inference of GCW, including absent, calling, resting, we decided to restrict only to the problem of detecting GCW's presence for simplicity. A complete state space inference will be a subject of future work. Despite such simplification, in general, solving POMDP is computationally hard. In fact, it has been proven to be PSPACE-complete [6]. Therefore approximated solution techniques are desirable for practical purposes. This paper proposes a novel approximated solution based on radial basis neural nets [7].

The rest of the paper is organized as follows. Sections II and III give a brief review of the POMDP framework and its solutions techniques. Section IV applies the framework to the specific problem of GCW's detection. Finally, results are given and discussed in Section V.

## II. The POMDP framework.

A thorough survey of the POMDP framework and its solution techniques can be found in [8], [9]. In this paper, only key ideas are summarized for the purpose of discussion.

Markov decision process (MDP) is the original framework for sequential decision making. It is a tuple of

$$< \mathcal{S}, \mathcal{A}, T, R >$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T$ is the transition matrix/kernel that dictates the dynamic of states and finally, $R$ is the reward that express preference of certain action in certain state. For example, in the GCW problem, the state space can be absent, calling, resting. Namely,

$$s \in \mathcal{S} = \{\text{absent}, \text{calling}, \text{resting}\} \tag{1}$$

The actions can be two-fold and the action space can be "declare present and sample next time", or "declare present and do not sample next time", or "declare absent and sample next time", or "declare absent and do not sample next time". Notationally,

$$a \in \mathcal{A} = \{\text{PS}, \text{PN}, \text{AS}, \text{AN}\} \tag{2}$$

One limitation of the MDP is that it assumes the states are completely observable. However, that is not always the case. For instance, in the detection of GCW problem, true state of a GCW is not observed, it is *inferred* from the spectrogram. To overcome this limitation, POMDP was proposed to generalize MDP with partially observable states. POMDP is a tuple of

$$< \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z >$$

where $\mathcal{O}$ is the observation space and $Z$ is the emission matrix, which specifies the probability of seeing each observation in each state.

Unlike MDP where decision maker only needs to keep track of the current state, POMDP requires the entire history of observations being kept tracked. Obviously, this is not feasible. Fortunately, [10] shows that the current *posterior distribution on the state space given the entire observation history*, or *belief*, is a sufficient statistics for the entire history. Thus POMDP can be reformulated in the form of MDP using the notion of belief state. A belief-state MDP is a tuple of

$$< \mathcal{B}, \mathcal{A}, \bar{T}, \bar{R} >$$

where $\mathcal{B}$ is the belief state space, which is a probability simplex on the original state space, $\bar{T}$ and $\bar{R}$ are the transformed versions of the original $T$ and $R$. For example,

$$b \in \mathcal{B} = \{\Pr(\text{absent}), \Pr(\text{calling}), \Pr(\text{resting})\}$$

Given the POMDP setup, solving POMDP means finding an optimal policy function $\pi^*$, which is a mapping from belief state to action. Namely,

$$\pi : \mathcal{B} \to \mathcal{A}$$

Common techniques for solving POMDP is discussed in the next section.

## III. POMDP solution techniques

This paper focuses on the infinite horizon POMDP. Therefore, a *discount* factor $\gamma$ is needed to ensure the existence of a stationary policy [11]. There exists two main classes of algorithm to find the optimal stationary policy: value iteration and policy iteration, both of which rely on the celebrated Bellman's equation [10].

### A. Value iteration

Value iteration sequentially find the optimal value function for each horizon $t$. As $t \to \infty$, $V_t^*(b) \to V^*(b)$. This infinite horizon optimal value function is then used to rank actions and thus find the best policy. This class of algorithms can be summarized using the following three equations. Various

algorithm differs only in the way they compute these equations [9].

$$V_t^*(b) = \max_{a\in\mathcal{A}}[\bar{R}(b,a) + \gamma \sum_{b'\in\mathcal{B}} \bar{T}(b,a,b')V_{t-1}^*(b')]$$
$$Q^*(b,a) = \bar{R}(b,a) + \gamma \sum_{b'\in\mathcal{B}} \bar{T}(b,a,b')V^*(b') \text{ where } V_t^* \to V^*$$
$$\pi^*(b) = \arg\max_{a\in\mathcal{A}} Q^*(b,a)$$

### B. Policy iteration

Policy iteration, on the other hand, sequentially improves the policy function $\pi_t(b)$ to yield better value $V_t^{\pi_t}(b)$ at each iteration. Ultimately the policy will converge to the optimal stationary policy. This class of algorithms can also be summarized using the following three equations.

$$V_t^{\pi_t}(b) = \bar{R}(b,\pi_t(b)) + \gamma \sum_{b'\in\mathcal{B}} \bar{T}(b,\pi_t(b),b')V_{t-1}^{\pi_t}(b')]$$
$$Q_{t+1}(b,a) = \bar{R}(b,a) + \gamma \sum_{b'\in\mathcal{B}} \bar{T}(b,a,b')V_t^{\pi_t}(b')$$
$$\pi_{t+1}(b) = \arg\max_{a\in\mathcal{A}} Q_{t+1}(b,a) \text{ until } \pi_t \to \pi^*$$

### C. Q-learning

In both value and policy iteration, the Q function, which measure the value of being in a state and performing an action, is merely an intermediate step to translate from the value function to the policy function. In the Q-learning algorithm [12], it is possible to iterate directly on the Q function without first finding the value function. The algorithm can be described by the following equations.

$$\begin{aligned}\delta(s_t,a_t) &= r_t + \gamma \max_{a\in\mathcal{A}} Q(s_{t+1},a) - Q(s_t,a_t)\\ Q(s_t,a_t) &\leftarrow Q(s_t,a_t) + \alpha\delta(s_t,a_t)\end{aligned} \tag{3}$$

where $\delta$ denotes the *temporal difference* [13] between successive iterations and $r_t$ is the immediate reward, which is a realization of the random variable $R(s,a)$. Q-learning was originally proposed as a model-free approach to solve MDP problems, as there is no need for a transition matrix/kernel in the calculation of temporal difference. It was also proven to converge to the optimal Q-function under the condition that all states are visited infinitely often [12]. This condition effectively requires that the decision maker takes action not only to *exploit* the best reward but also to *explore* the state space. Hence a stochastic policy is desirable, and a commonly chosen one is the softmax policy function, instead of a hard max in value and policy iteration. Namely,

$$\pi_t(s,a) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'\in\mathcal{A}} e^{Q(s,a')/\tau}} \tag{4}$$

where $\tau$ is the Boltzmann temperature constant.

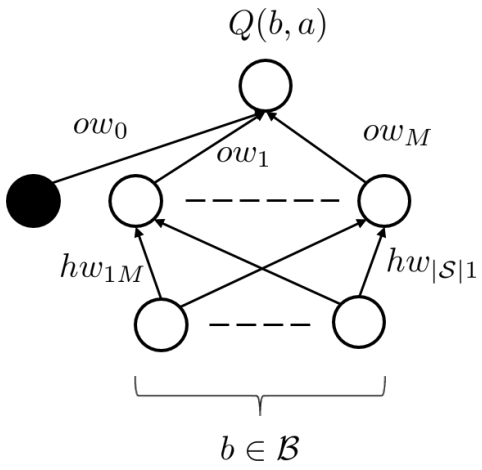


Fig. 2. Sigmoid neural net.

In order to generalize Q-learning from MDP to POMDP, syntactically, one simply needs to replace $s$ with $b$. However, since the belief state space is continuous, the Q function cannot be stored in a look-up table anymore. Instead, it needs to be approximated using function approximation techniques. One formalism of function approximator is the neural net.

### D. Neural net as Q-function approximator

The two types of neural net considered in this paper are the sigmoid neural net and the radial basis neural net [7]. Figure 2 shows the structure of a sigmoid neural net with $M$ hidden nodes. The total number of weights in the network includes $M+1$ output weights and $M \times |\mathcal{S}|$ hidden weights. The input to the network is a belief vector of dimension $|\mathcal{S}|$, while the output is the Q function for a particular action [1].

Sigmoid neural net has an interpretation based on a mathematical fact that the value function is a piece-wise linear and convex function of the belief state [14]. Namely,

$$V_t^*(b) = \max_{\alpha\in\nu_t} b^T\alpha$$

where $\nu_t$ is the space of all possible $\alpha$ vector at time $t$. Thus a hidden node's weights can be interpreted as an $\alpha$ vector. Using sigmoid neural net to approximate Q function has already been done in [13].

This paper uses a different neural net based on radial basis functions (RBF) [7], which is shown in Figure 3. For $M$ hidden nodes, there is a total of $M+1$ output weights and $M+1$ hidden centroid vectors $\Phi$. The motivation behinds using RBF is inspired by a new class of POMDP solution techniques called point-based solvers [15]. The point-based

[1]While it is possible to define a single network with multiple outputs, each for an action, it is harder to debug and therefore was not adopt in this paper

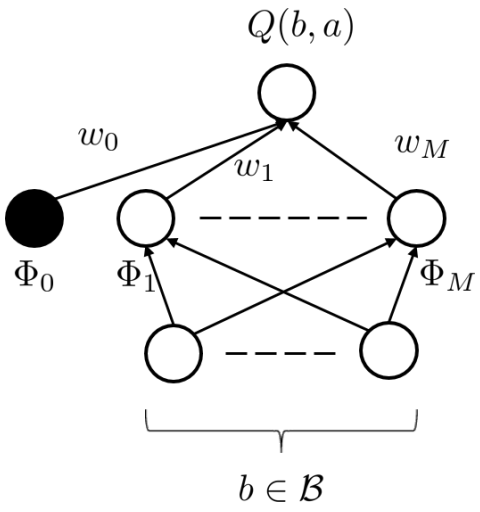


Fig. 3. Radial basis neural net.

approach keeps track of only reachable belief points instead of the set $\nu_t$ whose size grows exponentially fast as the time horizon increases. Each hidden centroid vector $\Phi$ can then be interpreted as a reachable belief point.

Training an RBF neural net to approximate the Q function has to be done using a stochastic gradient training algorithm because the next belief vector depends on the current output of RBF network. The pseudo code for training is given in Algorithm 1 [2].

## IV. Application to the detection of GCW

This section applies the POMDP framework to the specific problem of detecting GCW using acoustic data. Five hour of recorded data were analyzed using a spectrogram with Hamming window of 0.5 second length, 0.1 second increment and 128 frequency points (see Figure 4).

A portion of the data were used to construct a finite observation space $\mathcal{O}$. K-means, with $k = 4$, was used to quantized vector slices of a spectrogram into four typical observations, as shown in Figure 5. The first and last typical observations are transient noise, while the third typical observation is stationary noise. The second typical observation is the GCW's song.

The state space and action space are given by Equation (1) and (2).

Baum-Welch algorithm (EM on HMM) [3] was then used to learn the state transition and emission matrices. The result is presented in a state diagram (see Figure 6). The weight on each edge denotes the transition probability. Notice that the absent and resting states share the same observation distribution for

[2]The output weights are initialized by the uniform $[0, 1]$ distribution and the hidden centroid vectors are initialized using the Dirichlet distribution.

**Algorithm 1** Training algorithm for radial basis neural net (with a fixed spread constant $\sigma$) to approximate Q function.

**for** each iteration **do**

- $Q(b_t, a) = w_0 + \sum_{i=1}^{M} w_i \exp\{-\frac{\|b-\Phi_i\|^2}{\sigma}\}$, $\forall a \in \mathcal{A}$
- Sample an action $a_t$ according to (4) for POMDP
- Find next belief $b_{t+1}$ using Bayes' rule.
- Define target using Equation (3) for POMDP

$$t = \bar{r}_t + \gamma \max_{a \in \mathcal{A}} Q(b_{t+1}, a)$$

- Update $Q(b_t, a_t)$ using the following equations

$$y = w_0 + \sum_{i=1}^{M} w_i \exp\{-\frac{\|b-\Phi_i\|^2}{\sigma}\}$$
$$\frac{\partial \epsilon}{\partial w_i} = (y-t)\Phi_i,\ i = 0 \to M$$
$$\frac{\partial \epsilon}{\partial \Phi_i} = (y-t) w_i \exp\{-\frac{\|b-\Phi_i\|^2}{\sigma}\} \frac{b-\Phi_i}{\sigma},\ i = 1 \to M$$
$$w_i = w_i - \frac{\partial \epsilon}{\partial w_i},\ i = 0 \to M$$
$$\Phi_i = \Phi_i - \frac{\partial \epsilon}{\partial \Phi_i},\ i = 1 \to M$$

**end for**

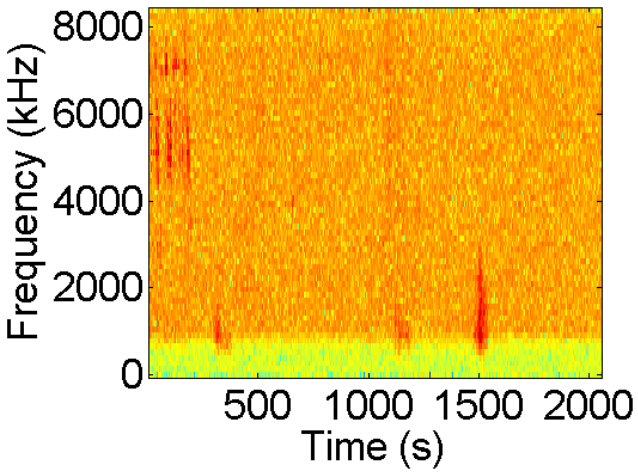


Fig. 4. Data spectrogram.

noise $P_n$ while the calling state has the observation distribution for GCW's song $P_s$. The mux on each state is used to control whether an observation is taken at the *next time step*.

The immediate reward function at each time step could be defined as follows.

$$\begin{aligned}\bar{r}_t =& \mathbb{I}(a_t = \{\text{PS, AS}\}) \times \lambda + \\ & \mathbb{I}(a_t = \{\text{PS, PN}\}) \times (b(\text{calling}) + b(\text{resting})) + \\ & \mathbb{I}(a_t = \{\text{AS, AN}\}) \times b(\text{absent})\end{aligned}$$

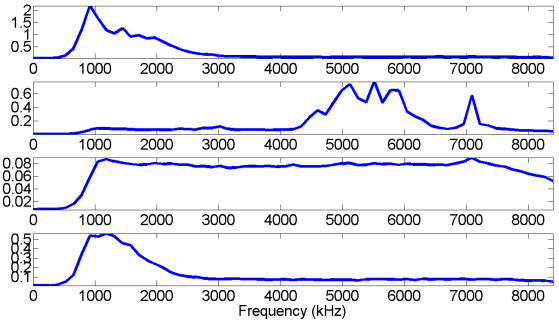


Fig. 5. Quantized observations.

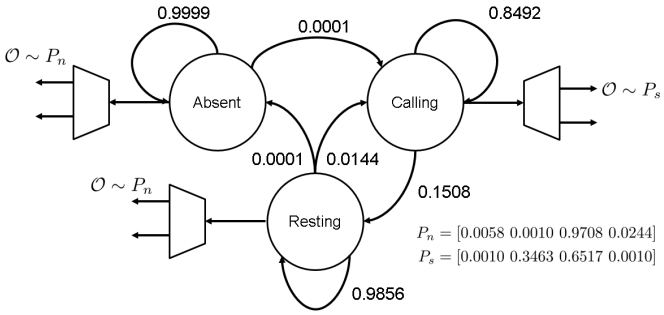


Fig. 6. State and observation model.

where $\mathbb{I}(\cdot)$ is an indicator function and $\lambda$ was selected to trade off between the number of sample usage and the detection accuracy. In this paper, $\lambda$ was set at 0.6154 to achieve roughly $10\%$ of sample usage on average. In addition, the discount factor was set at 0.66.

Finally, the spread constant of the RBF neural net $\sigma$ was set at 0.01 and the Boltzmann temperature constant $\tau$ was set at 0.06.

## V. Results and Discussion

The policy obtained by solving the POMDP problem defined in Section IV using Algorithm 1 is shown in the first plot of Figure 7. The tracked belief is also shown in the plot below. As can be readily seen, the policy is very likely to declare present and skip the next sample (the blue line) when its belief about calling or resting is high; there is still a small chance of taking new sample to update the belief (the green line). When the belief about absent is high, the policy declares absent and and skip next sample with high probability (the magenta line), leaving only a small chance of taking new sample to update the belief (the red line).

The last plot of Figure 7 shows the state estimates obtained by running the Viterbi decoding algorithm. As discussed in Section I, these estimates are the best possible and hence can

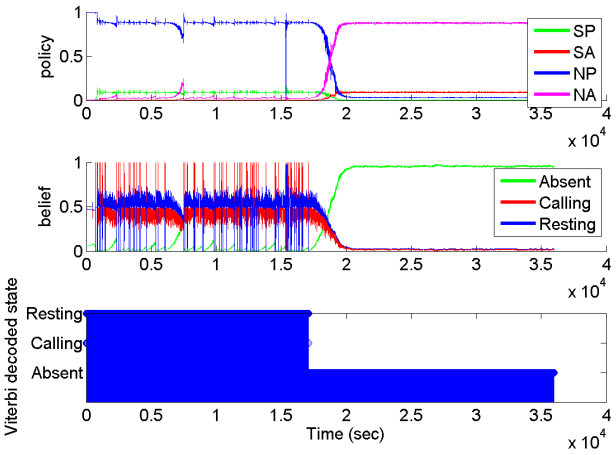


Fig. 7. Policy, belief and true state for one hour of data.

serve as a benchmark for how well the POMDP technique works. Over five hour of data, the POMDP technique demonstrates that it can use only 9.18% of the samples on average and still incurring only 9.07% detection error compared to a full Viterbi decoding.

## VI. Conclusion and Future Work

Unlike the classical Viterbi decoding and Bayesian filtering algorithms, inference algorithms based on the POMDP framework have great potential to perform inference tasks online with efficient usages of samples. One way to solve POMDP, i.e. to find the optimal policy, is to use Q-learning. This paper proposes an alternative form of the Q-learning algorithm based on RBF neural net. An application of this algorithm for the detection of GCW has demonstrated that it can performing inference tasks online efficiently. Future work will investigate on whether there is any fundamental difference in the performance between the different choices of neural net to approximate the Q function, such as the sigmoid and the RBF neural net.

## References

[1] W. Leonard, J. Neal, and R. Ratnam, "Variation of type B song in the endangered golden-cheeked warbler (dendroica chrysoparia)," *The Wilson Journal of Ornithology*, vol. 122, no. 4, pp. 777–780, 2010.

[2] D. Jun and D. Jones, "The value of sleeping: a rollout algorithm for sensor scheduling in HMMs," *GlobalSIP*, 2013.

[3] B. Hajek, "An exploration of random processes for engineers," *class notes for ECE 534*, 2009.

[4] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Computation*, vol. 4, no. 2, pp. 234–242, 1992.

[5] N. de Freitas, M. Milo, P. Clarkson, M. Niranjan, and A. Gee, "Sequential support vector machines," in *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.* IEEE, 1999, pp. 31–40.

[6] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[7] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[8] K. P. Murphy, "A survey of POMDP solution techniques," *environment*, vol. 2, p. X3, 2000.

[9] D. Braziunas, "POMDP solution methods," *University of Toronto, Tech. Rep*, 2003.

[10] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, 1995, vol. 1, no. 2.

[11] E. J. Sondik, "The optimal control of partially observable markov processes over the infinite horizon: Discounted costs," *Operations Research*, vol. 26, no. 2, pp. 282–304, 1978.

[12] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[13] M. A. Wiering and T. Kooi, "Region enhanced neural q-learning for solving model-based pomdps," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–8.

[14] E. J. Sondik, "The optimal control of partially observable markov processes." DTIC Document, Tech. Rep., 1971.

[15] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based pomdp solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.