Soft Actor-Critic for railway optimal maintenance planning under partial observability

Giacomo Arcieri
PhD Student, Institute of Structural Engineering, ETH Zürich, Zürich, Switzerland

Cyprien Hoelzl
PhD Student, Institute of Structural Engineering, ETH Zürich, Zürich, Switzerland

Oliver Schwery
Manager Strategic Asset Management, Swiss Federal Railways SBB, Bern, Switzerland

Daniel Straub
Professor, Engineering Risk Analysis Group, TUM, Munich, Germany

Konstantinos G. Papakonstantinou
Professor, Dept. of Civil & Environmental Engineering, Penn State Univ., PA, USA

Eleni Chatzi
Professor, Institute of Structural Engineering, ETH Zürich, Zürich, Switzerland

ABSTRACT: The optimal maintenance planning for railway systems forms a complex sequential decision-making problem. Optimal maintenance actions ought to be configured on the basis of updated rail condition estimates. To this end, structural health monitoring solutions can be used for reliably tracking the condition of railway infrastructure. However, the measurements gathered from continuous monitoring can only offer incomplete, often noise-corrupted, information of the real condition states, which implies the need for decision–making under uncertainty. For tackling the inherent uncertainty, the problem can be formalized as a Partially Observable Markov Decision Process (POMDP). Two families of methods are generally used to solve such formulations, namely Dynamic Programming (DP) and Reinforcement Learning (RL). In this work, we apply deep RL to solve a real-world railway maintenance planning problem modeled as a POMDP without assuming any knowledge of the problem parameters, in order to derive a full model-free solution. In particular, we employ the Soft Actor-Critic method, extended to partial observability, and compare the quality of the solution against classical DP methods analyzed in previous works.

1. INTRODUCTION

Infrastructure, such as roads and bridges, can deteriorate over time, causing safety issues and needing costs for repairs. In particular, railway assets are subjected to loads and environmental factors that produce degradation and fine material infiltration in the track (Hoelzl et al., 2021). As a consequence, the railway network can incur deterioration of service, delays, environmental costs, working accidents or derailing risks (Lidén, 2015). Maintenance policies aim to prevent or control such deterioration processes and extend the lifespan of these structures by balancing the cost of mitigation actions with the risk associated to adverse conditions of the infrastructure.

Optimal maintenance planning refers to the problem of defining the maintenance policy, i.e., the sequence of maintenance actions, that minimize expected costs and risks over the operating life-cycle (Duffuaa et al., 1999). At every decision step, the executed maintenance action will bear an immediate effect on the infrastructure condition and potentially change the optimal sequence of actions during the remaining life of the system. As such, the optimal maintenance planning forms a complex sequential decision-making problem. In addition, it is usually not feasible to precisely observe the railway condition state, given the scale of the problem and the resulting economic costs. To remedy this issue, Structural Health Monitoring (SHM) (Straub et al., 2017; Andriotis et al., 2021) tools can be exploited to provide observations, which attempt to offer estimates of the structural state, through the use of sensors and associated condition indicators. However, the observations gathered from these tools can only offer incomplete, often noise-corrupted, information of the actual condition (state) of the system. This implies that decisions must be made under presence of uncertainties. One approach is to formally cast the decision problem under uncertainty as a Partially Observable Markov Decision Process (POMDP).

In a POMDP setting, the decision maker (or agent) receives an observation, possibly through use of an SHM system, and forms a belief over the system's (in this case the railway) state based on this observation. Consequently, the agent takes an action based on the current belief, which will bear an impact on the subsequent (updated) condition. The POMDP objective is to find the optimal sequence of maintenance actions that minimizes the total costs over the prescribed horizon. Relevant examples of POMDP modeling for optimal maintenance planning can be found in Madanat and Ben-Akiva (1994); Ellis et al. (1995); Papakonstantinou and Shinozuka (2014b); Memarzadeh et al. (2015); Papakonstantinou et al. (2018).

In order to solve a POMDP problem, two main families of methods are available. These are methods based on Dynamic Programming (DP) and methods belonging to the class of Reinforcement Learning (RL). Algorithms based on DP (Bertsekas, 2012), often also referred to as "planning", bring a solid mathematical foundation with optimality convergence properties. However, planning algorithms can usually only be applied to small and medium complexity problems. In addition, they assume full knowledge of the POMDP problem, namely access to (a model of) the transitions dynamics and observation generating process of the POMDP and their underlying parameters, in order to compute the solution. This solution is hence optimal (or near-optimal) for the assumed POMDP parameters. However, the inference of the transition dynamics and observation generating process of the problem is notoriously difficult in real-world applications (Papakonstantinou and Shinozuka, 2014b).

In previous work of the authoring team (Arcieri et al., 2022), we introduced a framework to fully recover the transition and the observation models from available real-world observation data, such that distributions of plausible values over the POMDP parameters are inferred. Although the methods presented are generally applicable, a POMDP solution deriving from DP algorithms remains conditioned on the inferred model parameters. The latter may naturally deviate from the actual underlying parameters, which leads to a violation of any guarantees for optimality of the computed policy. To tackle this, we merge DP with Bayesian decision making in order to incorporate information on the probabilistic distributions of the parameters into the solution. While this framework clearly improves the robustness of the POMDP solution over model uncertainty, the solution can still be subject to model errors and the real-world parameters can still differ from the inferred distributions.

Another approach is to solve the underlying POMDP problem without assuming any prior knowledge over the transition dynamics and the observation generating process; in this case, the policy is fully learned through interactions via trial and error learning. This family of methods is commonly called RL (Sutton and Barto, 2018), or deep RL when combined with deep learning schemes. While RL overcomes the problem of availability of a POMDP model (which, however, would still be needed for testing the learned policy before the actual real-world deployment), these methods require a massive number of samples. In addition, deep RL, albeit permitting solution of high dimensional problems (Arulkumaran et al., 2017), lacks optimality guarantees for the solution (Sutton and Barto, 2018).

The two aforementioned families of methods therefore both have their pros and cons. When the complexity of the problem is not prohibitive for DP applications and thus allows for use of both options, one has to decide whether it is preferred to compute a solution with some optimality guarantees, at the cost of placing strong assumptions, or to relax those assumptions at the cost of losing optimality guarantees. In this work, we thus compare a DP solution against a deep RL solution, on the case study of a real-world problem for optimal maintenance planning of railway assets, which is modelled as a POMDP. The inference of the entire POMDP model has been showcased in Arcieri et al. (2022), where we here adopt the mean values of the inferred distributions as the POMDP problem parameters. The interested reader is referred to Arcieri et al. (2022) for a complete description of the underlying problem, its POMDP modeling, and the inferred parameters.

An alternative solution approach, which likely constitutes a hybrid between the two aforementioned options (Morato et al., 2023), would assume access to the POMDP parameters to compute beliefs via Bayes theorem, which would be then fed to the deep RL algorithm as inputs. Namely, the POMDP problem is converted into the belief-MDP (Papakonstantinou and Shinozuka, 2014a) and then solved with deep RL techniques. The premise is that in engineering problems we usually have and/or need an underlying simulation environment that can be exploited. While this can be informative and generally leads to improved RL solutions, it still suffers from the model assumptions bias that characterize DP solutions. As such, this hybrid approach is better suited to very complex problems (e.g., multi-component systems (Andriotis and Papakonstantinou, 2019)) where "pure" DP or model-free RL approaches would not likely work.

In this work, we rely on a pure model-free deep RL approach, namely the Soft Actor-Critic (SAC) method (Haarnoja et al., 2018) for discrete action settings (Christodoulou, 2019). The method is here extended by the use of Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to handle the partial observability. Although SAC is considered a state-of-the-art approach for model-free RL and the use of LSTMs has become the standard to solve POMDPs (Schmidhuber, 1990; Dung et al., 2008; Zhu et al., 2017), to the best of our knowledge, no other work has presented this extension in a full model-free RL context to solve POMDP problems. We compare this approach in this work against the planning algorithm Q_{MDP} method (Cassandra et al., 1996), whose solution to the problem is shown in Arcieri et al. (2022).

The remainder of this paper is organized as follows. Section 2 provides the theoretical background on the POMDP problem setting, the RL framework, and the Q_{MDP} and the SAC methods. Section 3 provides details of the evaluation of the experiment and reports the results. Finally, Section 4 discusses the contributions, the results, and possible limitations of the work, and outlines possible future work.

2. Preliminaries

2.1. *The POMDP framework*

A POMDP can be considered as a generalization of a Markov Decision Process (MDP) for modelling a sequential decision making problem within a stochastic control setting, with uncertainty incorporated into the observations. A POMDP is defined by the tuple $\langle S, A, Z, R, T, O, b_0, H, \gamma \rangle$, where:

- S is the finite set of hidden states that the environment can assume. Here, we assume 4 possible discrete states (condition levels) of the railway track, which are not directly observed by the agent.
- A is the finite set of available actions. We consider 3 possible real-world actions, which we refer to as a_0 do-nothing, a_1 minor repair, and a_2 major repair, for which we had access to logged data over the Swiss railway network.
- Z is the set of possible observations, generated by the hidden states and executed actions, which provide partial and noisy infor-

mation about the actual state of the system. In this work, the "fractal value" indicator (Hoelzl et al., 2021) forms the set of continuous observations of the railway tracks, for which we had access to a database of 10 years of recordings.

- $R : S \times A \to \mathbb{R}$ is the reward function that assigns the reward $r_t = R(s_t, a_t)$ for assuming an action a_t at state s_t. In an optimal maintenance problem, rewards typically assume negative values that represent costs; see Arcieri et al. (2022) for a table of the costs assumed.
- $T : S \times S \times A \to [0,1]$ is the transition dynamics model that describes the probability $p(s_{t+1}|s_t, a_t)$ to transition to state s_{t+1} if action a_t is taken at state s_t.
- $O : S \times A \times Z \to \mathbb{R}$ is the observation generating process that defines the emission probability $p(z_t|s_t, a_{t-1}, z_{t-1})$, namely the likelihood to observe z_t if the system is at state s_t and action a_{t-1} was taken.
- b_0 is the initial belief on the system's state s_0.
- H is the considered horizon of the problem, which can be finite or infinite. In this work, we consider $H = 50$ time-steps, where every time-step equals 6 months of the real-world problem. The problem is hence *episodic* and reset after the horizon is reached (called episode).
- γ is the discount factor that discounts future rewards to obtain the present value.

In the POMDP setting, the agent takes a decision based on the belief over the system's state. The belief is hence defined as a probability distribution over S, which maps the discrete finite set of states into a continuous $|S|-1$ dimensional simplex (Papakonstantinou and Shinozuka, 2014a). It, therefore, offers sufficient statistics over the complete history of actions and observations. The belief over the system's state is updated every time the agent receives a new observation according to Bayes' rule:

$$b(s_{t+1}) = \frac{p(z_{t+1}|s_{t+1}, a_t)}{p(z_{t+1}|\mathbf{b}, a_t)} \sum_{s_t \in S} p(s_{t+1}|s_t, a_t) b(s_t) \quad (1)$$

where the denominator is the usual normalizing factor.

The objective of the POMDP is to determine the optimal policy π^*, which maps beliefs to actions such that the expected sum of rewards is maximized:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t r_t\right] \quad (2)$$

where $r_t = R(s_t, \pi(b_t))$. The optimal policy can be computed via algorithms based on DP (Bertsekas, 2012), which introduce the concepts of the value function V^π, namely the expected sum of rewards of policy π from a certain state, and the Q-value function Q^π (Sutton and Barto, 2018), which outputs the state-value for taking action a_t at state s_t and then following the policy π.

2.1.1. *The Q_{MDP} method*

The exact solution of a POMDP is generally intractable and approximations have been proposed in the literature (Parr and Russell, 1995). The advent of point-based value iteration algorithms allowed to efficiently solve large scale POMDP problems with good approximation (Spaan and Vlassis, 2005; Papakonstantinou et al., 2018). However, solving POMDP problems with continuous observations remains a challenging task, even for these methods, which rely on discretization of the observation space. As such, in this work we rely on the simpler Q_{MDP} (Cassandra et al., 1996) approach, which is not negatively affected by a continuous set of observations.

The Q_{MDP} method initially ignores the observation model and computes the Q-values Q^* of the underlying MDP given the transition model. It then determines the optimal action at each step by only updating the belief $b(s)$ via Equation 1. The optimal action is then selected according to:

$$\pi_{Q_{MDP}} = \arg\max_{a \in A} \sum_{s \in S} b(s) Q^*(s, a) \quad (3)$$

This results in extremely low computational load when compared to point-based methods, at the expense of reduced accuracy, in general problems.

2.2. *Reinforcement Learning*

RL methods can solve sequential decision-making problems, and in particular POMDPs, by maximizing the objective function in Equation 2 without assuming knowledge of the POMDP parameters. Rather, RL learns optimal policies through interactions between the agent and the environment and via trial end error learning, using samples of the environment. In the RL context, the term environment is associated with the POMDP problem where the agent acts, which outputs observations and rewards resulting from the assumed actions. Transition, observation, and reward models are usually hidden from the RL agent.

Deep RL methods can be categorized in different ways. A common approach to such a categorization is dependent on what the Neural Networks (NNs) parameterize and learn through samples. For instance, in value-based methods, NNs are used to approximate Q-values (Mnih et al., 2015). The optimal policy is then computed by picking the action associated with the maximum Q-value, which works efficiently in discrete control settings. In policy-based methods (Sutton et al., 1999), NNs directly parameterize the policy, which is then improved via application of a stochastic gradient descent/ascent scheme through samples. While these methods enabled the solution of complex problems, including continuous control problems and stochastic policy learning, they suffer from high variance and very low sample-efficiency. Actor-critic methods (Mnih et al., 2016) combine the two previous approaches, with one or more NNs that evaluate the current policy (critic) by learning Q-values and a policy network that learns how to act (actor). All methods that do not have direct access to a model belong also to the family of the so-called model-free RL. In model-based RL, NNs are also exploited to learn a model of the environment, which is also used for planning and solving the underlying task (Arcieri et al., 2021).

In the POMDP setting, the belief may not be directly inferred, although a few works point to such a direction, e.g., (Igl et al., 2018). In such cases, decisions are based on some form of history, which is needed to learn hidden states, such as the current observation and a memory of past observations. In practice, a window $\mathscr{W}$ of the last k observations is passed to the agent at each time-step and the classical feed-forward NNs are generally replaced by LSTMs (Zhu et al., 2017).

2.2.1. SAC-Discrete for POMDPs

SAC is an off-policy actor-critic algorithm that optimizes a stochastic policy by changing the objective in Equation 2 to the following entropy-regularized RL objective (Haarnoja et al., 2018):

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t \left(r_t - \alpha \mathscr{H}(\pi)\right)\right] \qquad (4)$$

Namely, the SAC agent maximizes the trade-off between the classical RL objective and the entropy $\mathscr{H}$ of the policy, while α is a temperature hyper-parameter to be optimized.

In the SAC method, 5 NNs are employed, whose parameters are here labelled as ϕ to parameterize the policy π_ϕ, θ_1 and θ_2 for the Q-values Q_{θ_j}, and $\theta_{targ,1}$ and $\theta_{targ,2}$ for the target Q-values $Q_{\theta_{targ,j}}$. Given a collected replay buffer $\mathscr{D}$, the network parameters are optimized according to the objectives $J_Q(\theta_j)$ and $J_\pi(\phi)$; see Christodoulou (2019) for a detailed discussion of the computation steps, which this work accurately implemented.

In order to handle the partial observability, in this work all 5 NNs in SAC are replaced by LSTMs, which are fed with a window $\mathscr{W}$ of the last 5 observations as inputs.

3. Evaluation methodology and results

The full evaluation methodology[1] of SAC is reported in detail in Algorithm 1. The NNs, composed by 3×100 LSTM layers and ReLu activation functions, are trained with a stochastic policy, which is then tested over 500 episodes to average the results over the stochasticity of the environment. All hyper-parameters have been optimized with a thorough grid-search (other relevant ones are reported in Algorithm 1).

The full methodology is repeated over 4 random seeds and the results are reported in Table 1 in terms of average performance and Standard Error (SE). In the first row, the optimal solution of the fully observed problem is reported as a benchmark, which can be computed via DP by considering the full observability of the hidden states. It hence represents an upper bound that cannot be achieved by POMDP

[1] Code available on GitHub.

Algorithm 1 SAC evaluation algorithm

```
Initialize $Q_{\theta_1}, Q_{\theta_2}, \pi_\phi, Q_{\theta_{targ,1}}, Q_{\theta_{targ,2}}$    $\triangleright$ Initialize local and target networks
$\theta_{targ,1} \leftarrow \theta_1, \theta_{targ,2} \leftarrow \theta_2$    $\triangleright$ Equalize target and local network weights
$\mathscr{D} \leftarrow \emptyset$    $\triangleright$ Initialize an empty replay buffer that collects entire trajectories for training
for TrainingEpisode = 0 to $N$ do    $\triangleright N = 60{,}000$
    $s_0 \sim T_0$, $z_0 \sim O_0$ and $\mathscr{W} \leftarrow \emptyset$    $\triangleright$ Initialize environment and window buffer of capacity 5
    for timestep t = 0 to $H$ do
        $\mathscr{W} \leftarrow \mathscr{W} \cup \{(z_t)\}$    $\triangleright$ Store the observation in the window buffer
        $a_t \sim \pi_\phi(\mathscr{W})$    $\triangleright$ Sample action from policy network given history of obs
        $s_{t+1} \sim T(s_t, a_t), z_{t+1} \sim O(s_{t+1}, a_t, z_t)$    $\triangleright$ Execute action and sample next state and obs
        $\mathscr{D} \leftarrow \mathscr{D} \cup \{(z_t, a_t, R(s_t, a_t))\}$    $\triangleright$ Store tuple in the replay buffer and current trajectory place
    end for
    for update step do    $\triangleright$ 10 update iterations, sampled two full trajectories each, $\alpha = 0.1$
        $\theta_j \leftarrow \theta_j - \lambda_Q \hat{\nabla}_{\theta_j} J(\theta_j)$ for $j \in \{1,2\}$    $\triangleright$ Update Q-network parameters
        $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$    $\triangleright$ Update policy network parameters
        $\theta_{targ,j} \leftarrow \tau\theta_{targ,j} + (1-\tau)\theta_j$ for $j \in \{1,2\}$    $\triangleright$ Update targets with Polyak average, $\tau = 0.995$
    end for
end for
Initialize ListEpisodeRewards $\leftarrow \emptyset$    $\triangleright$ Initialize list of costs
for TestingEpisode = 0 to $K$ do    $\triangleright K = 500$
    Set EpisodeRewards $\leftarrow 0$    $\triangleright$ Initialize episode costs
    $s_0 \sim T_0$, $z_0 \sim O_0$ and $\mathscr{W} \leftarrow \emptyset$    $\triangleright$ Initialize environment and window buffer of capacity 5
    for timestep t = 0 to $H$ do
        $\mathscr{W} \leftarrow \mathscr{W} \cup \{(z_t)\}$    $\triangleright$ Store the observation in the window buffer
        $a_t = \pi_\phi(\mathscr{W})$    $\triangleright$ Compute deterministic action from policy network given history of obs
        $s_{t+1} \sim T(s_t, a_t), z_{t+1} \sim O(s_{t+1}, a_t, z_t)$    $\triangleright$ Execute action and sample next state and obs
        EpisodeRewards $+= R(s_t, a_t)$    $\triangleright$ Collect reward
    end for
    ListEpisodeRewards $\leftarrow$ ListEpisodeRewards $\cup \{(\text{EpisodeRewards})\}$    $\triangleright$ Collect episode costs
end for
```

solutions, which also deal with uncertain observations.

The planning algorithm Q_{MDP} significantly outperforms the deep RL solution SAC for this problem, taking advantage of the enhanced information that its belief inputs provide. The SAC policy clearly improves during training, as the initial policy with random initialization would result in average costs that are at least an order of magnitude higher. However, it failed to attain a more sophisticated policy than Q_{MDP} is able to deliver. The SE shows that a large number of evaluations would not alter this conclusion. Note that the smaller SE of Q_{MDP} is mainly due to the larger number of samples used for evaluating the mean (10,000 for Q_{MDP} vs 2,000 for SAC).

Table 1: Evaluation results.

Method	Average costs	SE
Optimal solution (MDP)	-13,405	-
Q_{MDP} method (POMDP)	-14,374	35
SAC method (POMDP)	-16,918	528

4. Discussion and conclusions

In this work, a deep RL algorithm, namely the SAC method, is employed to solve a real-world optimal maintenance planning problem modelled as a POMDP. The performance is compared against a planning algorithm called Q_{MDP} method. The main contribution is thus formed by the comparison and introductory analysis of the trade-off between a DP solution, which has general optimality convergence guarantees but assumes complete knowledge of the problem, and a RL solution, which does not assume any prior model knowledge, at the cost of optimality convergence guarantees, for a real-world POMDP problem. In addition, this work also represents the first extension of the SAC method to partial observability in a full model-free RL context.

Although the DP planning method achieved superior results and outperformed the RL method for the problem considered here, it should still be reminded that the former assumed complete knowledge of the POMDP parameters and the solution found is thus optimal for these parameters. However, no assurance can be offered on the behaviour of the computed planning optimal policy in the real-world, whose dynamics may potentially differ from those tested here. The RL solution may still be meaningful for this problem if model assumptions on the problem setting are not deemed reliable. In addition, it should also be noted that we fed the RL algorithm with a history of actual observations, instead of computing the beliefs via Equation 1 and using them as inputs, e.g., as in Morato et al. (2023). While this latter technique would certainly lead to increased performance, it relies on the underlying POMDP model, something we wanted to avoid in this work. As a result, the considered problem here is far more complex to be tackled with RL techniques than the belief-MDP case.

A possible limitation of this work is that the past history of actions was not passed to the RL agent as input for subsequent decisions, while this extension has been proven to sometimes increase the performance in the POMDP context (Zhu et al., 2017). In addition, only one RL method is here evaluated. Future work will extend this investigation to further RL methods and various past history of information sequences and types.

5. REFERENCES

Andriotis, C. P. and Papakonstantinou, K. G. (2019). "Managing engineering systems with large state and action spaces through deep reinforcement learning." *Reliability Engineering & System Safety*, 191, 106483.

Andriotis, C. P., Papakonstantinou, K. G., and Chatzi, E. N. (2021). "Value of structural health information in partially observable stochastic environments." *Structural Safety*, 93, 102072.

Arcieri, G., Hoelzl, C., Schwery, O., Straub, D., Papakonstantinou, K. G., and Chatzi, E. (2022). "Bridging POMDPs and Bayesian decision making for robust maintenance planning under model uncertainty: An application to railway systems." *arXiv preprint arXiv:2212.07933*.

Arcieri, G., Wölfle, D., and Chatzi, E. (2021). "Which Model To Trust: Assessing the Influence of Models on the Performance of Reinforcement Learning Algorithms for Continuous Control Tasks." *arXiv preprint arXiv:2110.13079*.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). "Deep reinforcement learning: A brief survey." *IEEE Signal Processing Magazine*, 34(6), 26–38.

Bertsekas, D. (2012). *Dynamic programming and optimal control: Volume I*, Vol. 1. Athena scientific.

Cassandra, A. R., Kaelbling, L. P., and Kurien, J. A. (1996). "Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation." *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, IEEE, 963–972.

Christodoulou, P. (2019). "Soft actor-critic for discrete action settings." *arXiv preprint arXiv:1910.07207*.

Duffuaa, S. O., Raouf, A., and Campbell, J. D. (1999). "Planning and control of maintenance systems." *John Willey and Son, New York*.

Dung, L. T., Komeda, T., and Takagi, M. (2008). "Reinforcement learning for POMDP using state classification." *Applied Artificial Intelligence*, 22(7-8), 761–779.

Ellis, H., Jiang, M., and Corotis, R. B. (1995). "Inspection, maintenance, and repair with partial observability." *Journal of Infrastructure Systems*, 1(2), 92–99.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." *International Conference on Machine Learning*, PMLR, 1861–1870.

Hochreiter, S. and Schmidhuber, J. (1997). "Long short-term memory." *Neural Computation*, 9(8), 1735–1780.

Hoelzl, C., Dertimanis, V., Chatzi, E. N., Winklehner, D., Züger, S., and Oprandi, A. (2021). "Data driven condition assessment of railway infrastructure." *Bridge Maintenance, Safety, Management, Life-Cycle Sustainability and Innovations*, CRC Press, 3251–3259.

Igl, M., Zintgraf, L., Le, T. A., Wood, F., and Whiteson, S. (2018). "Deep variational reinforcement learning for POMDPs." *International Conference on Machine Learning*, PMLR, 2117–2126.

Lidén, T. (2015). "Railway infrastructure maintenance-a survey of planning problems and conducted research." *Transportation Research Procedia*, 10, 574–583.

Madanat, S. and Ben-Akiva, M. (1994). "Optimal inspection and repair policies for infrastructure facilities." *Transportation science*, 28(1), 55–62.

Memarzadeh, M., Pozzi, M., and Zico Kolter, J. (2015). "Optimal planning and learning in uncertain environments for the management of wind farms." *Journal of Computing in Civil Engineering*, 29(5), 04014076.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). "Asynchronous methods for deep reinforcement learning." *International Conference on Machine Learning*, PMLR, 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). "Human-level control through deep reinforcement learning." *Nature*, 518(7540), 529–533.

Morato, P. G., Andriotis, C. P., Papakonstantinou, K. G., and Rigo, P. (2023). "Inference and dynamic decision-making for deteriorating systems with probabilistic dependencies through Bayesian networks and deep reinforcement learning." *Reliability Engineering & System Safety*, 109144.

Papakonstantinou, K. G., Andriotis, C. P., and Shinozuka, M. (2018). "POMDP and MOMDP solutions for structural life-cycle cost minimization under partial and mixed observability." *Structure and Infrastructure Engineering*, 14(7), 869–882.

Papakonstantinou, K. G. and Shinozuka, M. (2014a). "Planning structural inspection and maintenance policies via dynamic programming and Markov processes. Part I: Theory." *Reliability Engineering & System Safety*, 130, 202–213.

Papakonstantinou, K. G. and Shinozuka, M. (2014b). "Planning structural inspection and maintenance policies via dynamic programming and Markov processes. Part II: POMDP implementation." *Reliability Engineering & System Safety*, 130, 214–224.

Parr, R. and Russell, S. (1995). "Approximating optimal policies for partially observable stochastic domains." *IJCAI*, Vol. 95, 1088–1094.

Schmidhuber, J. (1990). "Reinforcement learning in Markovian and non-Markovian environments." *Advances in neural information processing systems*, 3.

Spaan, M. T. and Vlassis, N. (2005). "Perseus: Randomized point-based value iteration for POMDPs." *Journal of Artificial Intelligence Research*, 24, 195–220.

Straub, D., Chatzi, E., Bismut, E., Courage, W., Döhler, M., Faber, M. H., Köhler, J., Lombaert, G., Omenzetter, P., Pozzi, M., et al. (2017). "Value of information: A roadmap to quantifying the benefit of structural health monitoring." *ICOSSAR-12th international conference on structural safety & reliability*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (1999). "Policy gradient methods for reinforcement learning with function approximation." *Advances in Neural Information Processing Systems*, 12.

Zhu, P., Li, X., Poupart, P., and Miao, G. (2017). "On improving deep reinforcement learning for POMDPs." *arXiv preprint arXiv:1704.07978*.