# Investigation of Inspection and Maintenance Optimization with Deep Reinforcement Learning in Absence of Belief States

Daniel Hettegger
*PhD Student, Chair of Robotics, Artificial Intelligence and Embedded Systems, Technical University of Munich, Germany*

Carmen Buliga
*PhD Student, Engineering Risk Analysis Group, Technical University of Munich, Germany*

Florian Walter
*Postdoctoral Scientist, Chair of Robotics, Artificial Intelligence and Embedded Systems, Technical University of Munich, Germany*

Elizabeth Bismut
*Senior Researcher, Engineering Risk Analysis Group, Technical University of Munich, Germany*

Daniel Straub
*Professor, Engineering Risk Analysis Group, Technical University of Munich, Germany*

Alois Knoll
*Professor, Chair of Robotics, Artificial Intelligence and Embedded Systems, Technical University of Munich, Germany*

ABSTRACT: Maintenance of deteriorating infrastructure is a major cost factor for owners and society. Therefore, efficient inspection and maintenance (I&M) strategies are of paramount importance. Deep reinforcement learning (DRL) has been proposed for maintenance optimization of deteriorating systems. For good performance, DRL relies on information rich state representations, but information about the state may only be available through costly inspections. One option to alleviate this is by use of belief states, however this might not always be possible due to incomplete model knowledge or computational constraints. Hence, there is potential for DRL approaches using only the information which is already available. In this work, we investigate several observation representations and compare them by training multiple DRL agents. Our experiments show that the choice of informative observation representations has a strong effect on the performance of the resulting optimized maintenance strategy.

## 1. Introduction

Inspection and maintenance (I&M) of deteriorating infrastructure represent major costs to owners. With increasing demand and age, more efficient I&M strategies are increasingly important. I&M planning can be efficiently formulated as a sequential decision-making problem for which, in recent years, deep reinforcement learning (DRL) has shown promising results by solving complex problem scenarios, such as mastering the game of Go (Silver et al., 2017). For good performance, reinforcement learning (RL) relies on an information rich state representation, to make informed decisions about actions. In the case of partial observability, such a representation might, however, be unavailable. One way to address the uncertainty of the system state is, to use the current belief state $b_t$. It serves as a probability distribution over the state space, and can be updated using observations. This has been successfully used to solve several partially observable I&M problems with DRL by Andriotis and Papakonstantinou (2019, 2021). However, in order to calculate the belief state, precise knowledge about the model is required. If this model information is not available during training or deployment of the agent, other options have to be considered. If just providing a single observation is insufficient, the agent can be provided with additional observations from previous time steps, as well as previously taken actions as proposed by Lin and Mitchell (1993). Depending on the system, this can result in very high dimensional observation representations, which can make training challenging. To this end, we investigate several observation representations in this work, which are created using only the information already available through inspections. The resulting observation spaces are then used to train multiple DRL agents, to compare the impact of different representation options.

## 2. Partially Observable Markov Decision Models

In order to model the deterioration problem and its possible interactions with a maintenance strategy, it is formally described using a Markov Decision Process (MDP), first proposed by Bellman (1957). In this setting, the deterioration problem is referred to as the environment, and a maintenance strategy as the agent. In the case of I&M optimization, the true state of the environment is not available to the agent, and information about it can only be gained using observations yielded through inspections. To model this inherent uncertainty about the state, Partially Observable Markov Decision Processes (POMDPs) can be used, which are a generalization of the MDP. The POMDP is formally defined using a 7-tuple $(S,A,Z,T,O,R,\gamma)$, where we follow the same notation as Memarzadeh et al. (2015):

- $S$ defines the hidden state space of the system, within which current states $s_t \in S$ lie,
- $A$ refers to the action space of the system, from which the agent can choose current actions $a_t \in A$,
- $Z$ is the observation space of the system, in which current observations $z_t \in Z$ lie,
- $T : S \times A \times S \to [0,1]$ is the transition probability function of the system,
- $O : S \times Z \to [0,1]$ is the observation probability function of the system,
- $R : S \times A \to \mathbb{R}$ is the reward function, which determines the reward $r_t$ at each time step,
- $\gamma$ is the discount factor for future rewards.

## 3. Deep Reinforcement Learning

A short introduction to reinforcement learning is provided here; for a more detailed introduction, see (Sutton and Barto, 2018), on which this summary is based. In the RL framework, tasks are formulated as a sequential decision-making problem: An agent performs actions in an environment and receives observations and rewards depending on its behavior. These interactions are formally described by the aforementioned MDP. The goal of RL is to find a policy $\pi$ that maximizes the expected future reward by choosing appropriate actions. This is achieved by interacting with the environment, and modifying the policy according to the observed states, actions and rewards. Formally, the goal is to maximize the expected return $G_t$, which is the discounted sum of future rewards.

$$G_t = r_t + \gamma\, r_{t+1} + \gamma^2\, r_{t+2} + \cdots = \sum_{i=0}^{T-t-1} \gamma^i\, r_{t+i} \quad (1)$$

The expected future return $G_t$ of taking an action $a$ in a state $s$, when following a policy $\pi$, is given by the so called action-value or Q-function $q_\pi(s,a)$.

$$q_\pi(s,a) = \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right] \quad (2)$$

When following an optimal policy $\pi_*$, the corresponding action-value function is referred to as the optimal action-value function $q_*(s,a)$.

$$q_*(s,a) = \max_{\pi} q_\pi(s,a) \tag{3}$$

It can be observed that by knowing the optimal action-value function, the optimal policy can be extracted in discrete action spaces, through always choosing the action corresponding to the maximum optimal action-value.

$$\pi_*(s) = \arg\max_{a} q_*(s,a) \tag{4}$$

So the optimal policy might either be learned explicitly or implicitly through the value functions.

One distinguishes approaches which learn or use a transition model $T$, so-called model-based RL approaches, from approaches which do not use a model of the environment, referred to as model free RL. A second important distinction can be made by the way the policy or value network is trained. If the interaction data between an agent and the environment has to originate from the policy that is currently trained, the approach is classified as on-policy. If this is not the case, the approach is referred to as an off-policy approach.

DRL refers to a set of approaches that apply artificial neural networks to approximate either a value function, a policy, or both. This allows problems with continuous and large discrete state-action spaces to be solved using RL. In an influential work, which revitalized the field, an agent was able to play several Atari games with super human performance by learning an approximate optimal action-value function using an artificial neural network (Mnih et al. (2013)).

As previously noted, most RL and especially model free RL approaches were created to solve MDP problems. Given observations with enough information, however, these approaches are still able to find policies with good performance.

## 4. Models and Methods

To investigate the impact of choosing different observation features, we use a deteriorating system as an example and perform several experiments in simulation with differing observation spaces for the same problem.

### 4.1. *Deterioration Model*

For our experiments, we investigate a single component system, whose deterioration is described by the gamma process. The gamma process is an additive process, whose increments are gamma distributed: $\Delta D(t_{\text{age}}) \sim \text{Gamma}(k(t_{\text{age}}), \lambda)$. $k(t_{\text{age}})$ is the shape parameter, which depends on the age $t_{\text{age}}$ of the component, and $\lambda$ is the rate parameter. At every time step, the deterioration state is updated according to the following rule:

$$D(t) = D(t-\Delta T) + \Delta D(t_{\text{age}}) \tag{5}$$

The shape parameter of the gamma distribution is described by

$$k(t_{\text{age}}) = a\lambda \left( (t_{\text{age}} + \Delta T)^b - t_{\text{age}}^b \right) \tag{6}$$

The parameters of the model are summarized in Table 1. The parameter $a$ is uncertain and is modeled by a lognormal distribution with mean $\mu_a$ and variance $\sigma_a^2$.

At $t = 0$ the system is initialized with the following values: $D(0) = 0$, $t_{\text{age}} = 0$. We consider a time horizon of 200 time units and discretize time in steps of $\Delta T = 0.1$, resulting in 2000 total time steps per episode.

Failure occurs if the damage $D(t)$ exceeds a threshold $d_{crit}$. When the component fails it is immediately replaced, the initial deterioration state $D(t)$ and age $t_{\text{age}}$ of the new component are set to zero and the process proceeds with the same gamma process (with the same value of $a$). Failure also results in a failure cost of $c_{\text{failure}}$, to model the cost of replacement and costs related to unavailability of the component.

### 4.2. *Maintenance Actions*

The system has several maintenance actions that can be selected by the maintenance policy. The available actions at every time step are the following: *Do nothing*, *Inspect* and *Repair*. In the *Do nothing* action, no maintenance or inspection is performed. When the *Inspect* action is taken, the current deterioration state of the component $D(t)$ is observed, with perfect information and a cost

| Parameter | Type | Value |
|---|---|---|
| $a$ | stochastic | $a \sim \ln \mathcal{N}\left(\mu_a, \sigma_a^2\right)$ |
| $\mu_a$ | Fixed | 0.002 |
| $\sigma_a^2$ | Fixed | 0.001 |
| $b$ | Fixed | 2 |
| $\lambda$ | Fixed | 10 |
| $d_{crit}$ | Fixed | 25 |
| $r_e$ | Fixed | 0.5 |
| $T$ | Fixed | 200 |
| $\Delta T$ | Fixed | 0.1 |
| $c_{\text{inspection}}$ | Fixed | 5 |
| $c_{\text{repair}}$ | Fixed | 50 |
| $c_{\text{failure}}$ | Fixed | 1500 |

*Table 1: Overview of deterioration model parameters.*

$c_{\text{inspection}}$ is incurred[1]. When a *Repair* is performed, the current state of the system $D(t)$ is improved by a factor $r_e$ (repair effect): $D(t) = D(t) \cdot r_e$. This results in a cost of $c_{\text{repair}}$. When an action is chosen by a policy, the algorithm will first update the internal deterioration state $D(t)$ according to Eq. 5 and check for failure.

## 4.3. *Reinforcement Learning Environment*

In order to enable reinforcement learning for the I&M problem, the deterioration model is adapted in the following ways: The agent is able to perform four distinct actions, including a *Repair and Inspect* action, which allows the agent to perform maintenance and inspection in the same time step. The effect of this action is equivalent to first performing the *Repair* action, followed by the *Inspect* action, yielding the deterioration state after the repair.

As described in section 4.1, observations of the system state are available through inspections, meaning there is no information gain if no inspection is performed. We investigate several options of implementing these observations for the inspection and maintenance task without use of a belief state $b_t$. The simplest observation, which can be given to the agent, is just the current system state $d_c$. If the agent has performed an inspection in this time step, $d_c$ will take the value of the component state $D(t)$, otherwise $d_c$ takes a placeholder value of -1, to enable the agent to distinguish it from real state observations. The observation of the current time step within the episode is denoted as $t_c$, starting at zero and increasing by one each possible interaction cycle ($t_c = \frac{t}{\Delta T}$). By holding the information of the last inspection state $d_c$ in memory, the last known state $d_l$ of the system can be provided to the agent. If an inspection has been performed $d_l$ will take the value of the component state, if no inspection is performed $d_l$ will remain constant according to the last inspection value. Lastly, by storing the time step of the last inspection, we may provide the time steps since inspection $t_i$ to the agent. $t_i$ takes the value of zero if an inspection has been performed in the current time step, and increases by one in each following time step if no inspection is performed.

### 4.3.1. *Investigated Observation Spaces*

In order to investigate the effect of different observation features, we use the deterioration model described in section 4.1 and define four DRL setups, each with a distinct observation space. Table 2 shows an overview of the observation spaces for each of the four investigated setups. Both setups one and two rely on the current inspection state $d_c$, whereas setups three and four use a representation that retains the value $d_l$ and age of $t_i$ previous inspection results.

| | $Z$ | $d_c$ | $t_c$ | $d_l$ | $t_i$ |
|---|---|---|---|---|---|
| Setup 1 | $Z_1$ | $\times$ | | | |
| Setup 2 | $Z_2$ | $\times$ | $\times$ | | |
| Setup 3 | $Z_3$ | | | $\times$ | $\times$ |
| Setup 4 | $Z_4$ | | $\times$ | $\times$ | $\times$ |

*Table 2: Observation spaces for setups one to four.*

During training and evaluation of the DRL agents, the uncertain parameter $a$, which controls the deterioration of the system, is sampled according to its distribution (Table 1) at the beginning of an episode and remains constant throughout. The value of $a$ is hidden, meaning the agent has to learn a policy which is optimal considering the full distribution of $a$.

[1]While inspection yields perfect information about the deterioration state, it should be noted that the problem remains partially observable, as this information is only available through inspections and the hidden state of $a$ is never observed.

The reward function $R$ is defined by the negative sum of costs incurred in a time step, encouraging the agent towards minimizing total life-cycle cost.

$$r_t = -\left(c_{t,\text{inspection}} + c_{t,\text{repair}} + c_{t,\text{failure}}\right) \quad (7)$$

The associated costs in a time step $t$ are denoted as $c_{t,\text{inspection}}, c_{t,\text{repair}}, c_{t,\text{failure}}$ and take the values described in Table 1 if the respective action was taken or if failure has occurred, otherwise their corresponding value is zero. For the discount factor, $\gamma$ we choose a value of 1, meaning no discounting is performed.

## 4.4. *Deep Reinforcement Learning Approach*

For the DRL agent, which is trained to perform maintenance on the system, we choose Proximal Policy Optimization (PPO) proposed by Schulman et al. (2017). PPO is a model free, on policy, actor-critic method, which is able to increase training stability by constraining policy updates within a certain region to prevent changes which degrade the current policy too much. All training runs were performed using state and reward normalization. The specific implementation details can be found in our accompanying code repository[2].

## 4.5. *Baseline Policies*

As a baseline for the DRL agents' performance, the following three policies are implemented and evaluated.

### 4.5.1. *Corrective Replacement*

This policy performs only the *Do nothing* action, meaning the system is left to deteriorate without performing any maintenance or inspections and is only replaced upon failure. This is chosen over a random policy to perform as a neutral baseline, due to the high associated cost of performing repairs and inspections in almost every time step. This policy achieves an average life cycle cost (LCC) of $C_{LC} = 1593.75 \pm 363.09$.

[2]https://github.com/INFRA-RELEARN/ICASP14-Submission

### 4.5.2. *Condition Based Maintenance*

The *Condition Based Maintenance* (CBM) policy $\pi_{CBM}(s|t_{ins}, d_{thr})$ inspects the system every $t_{ins} \in [0, T]$ time steps. If the observed system state $d_c$ is at or above the repair threshold $d_{thr} \in [0, d_{crit}]$ $d_c \geq d_{thr}$, the repair action is chosen by the policy. Such a heuristic policy is common in risk-based I&M planning (Straub and Faber, 2006). To find the optimal parameters $t^*_{ins}$ and $d^*_{thr}$ we employ a grid search over the parameter space with $N_{MC} = 128$ Monte Carlo simulations for every pair of values. The optimal parameters are chosen according to the lowest average value of the LCC.

$$t^*_{ins}, d^*_{thr} = \underset{t_{ins}, d_{thr}}{\arg\min} \underset{\pi_{CBM}(s|t_{ins}, d_{thr})}{\mathbb{E}} [C_{LC}] \quad (8)$$

This policy achieves an average LCC of $C_{LC} = 452.23 \pm 333.7$ using optimal parameters $t_{ins} = 11.8$, $d_{thr} = 16$. The high standard deviation is due to the sampled parameter $a$, which can create episodes in which repairs have to be performed with high frequency to keep the system from failure. The optimal policy according to the minimum average LCC does therefore accept failures in some situations, to achieve a better overall performance.

### 4.5.3. *Free Inspection Policy*

To provide a lower bound on the cost of maintaining the system, a *Condition Based Maintenance* strategy with free inspections (CBM-FI) is considered. This policy has perfect information of the deterioration state of the system at all times without inspections and will always perform repairs just before failure would occur at a threshold of $d_{thr} = 24$. This policy achieves an average LCC of $C_{LC} = 234.38 \pm 65.47$.

## 5. RESULTS

Using the four setups described in section 4.3.1, ten DRL agents with different initialization were trained for 12000 episodes. The resulting training curves can be seen in Figure 1. It can be seen that within about 1000 to 2000 training episodes, most agents reach the performance of the *Corrective replacement* policy. Agents trained in setup one and two are not able to outperform this cost, and consistently learn to not perform any actions.

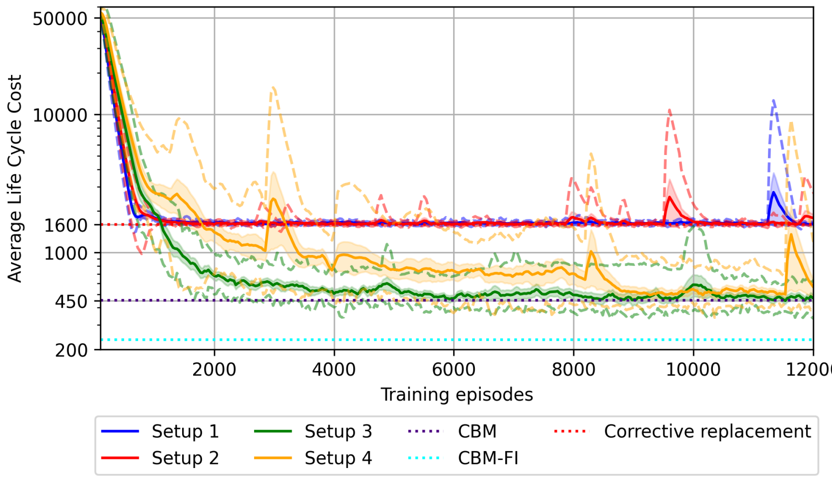

*Figure 1: Average life cycle cost $C_{LC}$ of the DRL agents trained in four observation spaces. The minimum and maximum $C_{LC}$ for each setup are indicated using dashed lines, the std. error is indicated by the transparent regions in the respective colors. Note that setup one and two are not able to produce policies which exceed a "Corrective Replacement" strategy. Agents using richer observations spaces, like setup three and four, are able to produce maintenance strategies that keep the system maintained.*

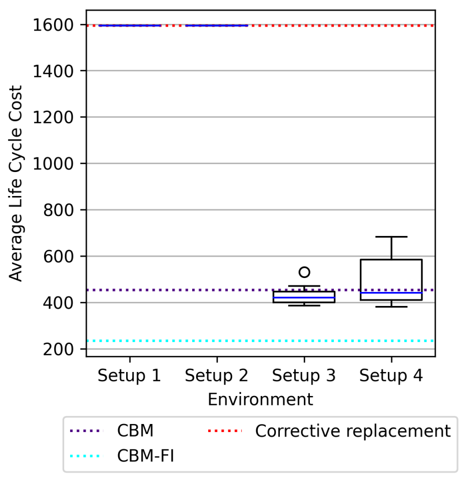

*Figure 2: Box plot of the life cycle costs $C_{LC}$ for the final model in each of ten training runs of each observation space. Agents trained in setups three and four with observation spaces retaining inspection information often even outperform the CBM policy.*

Agents trained in setups three and four, however, are able to quickly surpass this threshold and go on to reach and even outperform the *CBM* strategy. Setup four exhibits slower convergence compared to setup three.

To show the distribution of final agent performances, we select the best model from each training process and evaluate it using 128 test episodes with fixed parameters. All heuristics are also evaluated using the same 128 episodes to serve as a baseline. The resulting box plot, can be seen in Figure 2. Agents trained using the observation spaces of setup one and two consistently learn to perform no actions at all, achieving the same final LCC as the *Corrective replacement* heuristic. By contrast, agents trained using setups three and four in most cases exceed the *CBM* strategy. Table 3 shows the best, median and worst LCC for each of the four setups.

## 6. Discussion

The experiments show that the choice of the observation space for the DRL Agent can have a profound impact on final performance and training stability. In setups one and two, where the observation spaces do not contain information about past inspection states, the DRL agents are not able to find a policy that keeps the system maintained. In setup one, merely the current inspection value when available was provided to the agents. Intuitively, it can be observed that a deterministic agent, can only have a single action in response to the value of no observation ($d_c = -1$), therefore if no inspection was performed, the agent has to perform the same exact action in any circumstance, leading to either a policy that inspects at least every second time step or to a policy of no inspections. The performance of agents trained using these observations is therefore not surprising. To alleviate this, it might be possible to make use of recurrent neural networks for the policy, which are able to store relevant information over many time steps. It can also be noted that a CBM policy similar to the one described in section 4.5.2 could not be implemented using this observation space without the agent internally keeping track of the time steps which have passed since the last inspection.

Interestingly, even though setup two provides in-

| | | Setup 1 | Setup 2 | Setup 3 | Setup 4 |
|---|---|---|---|---|---|
| Best | $C_{LC}$ | 1593.75 ± 363.09 | 1593.75 ± 363.09 | 386.60 ± 325.32 | **380.27 ± 238.03** |
| Median | $C_{LC}$ | 1593.75 ± 363.09 | 1593.75 ± 363.09 | **419.63 ± 325.84** | 441.82 ± 251.73 |
| Worst | $C_{LC}$ | 1593.75 ± 363.09 | 1593.75 ± 363.09 | **531.05 ± 196.10** | 683.32 ± 97.88 |

*Table 3: Best, median and worst average LCC for each of the four setups. Best values in a row are highlighted.*

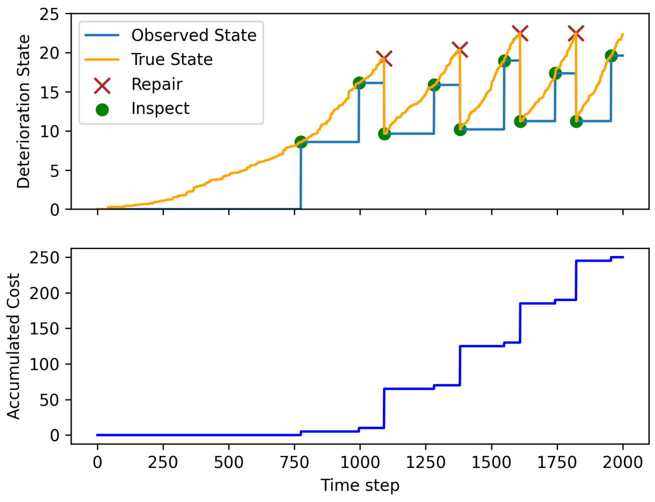

*Figure 3: Sample deployment of DRL agent trained in observation space $Z_3$. (Life cycle cost $C_{LC} = 250$)*

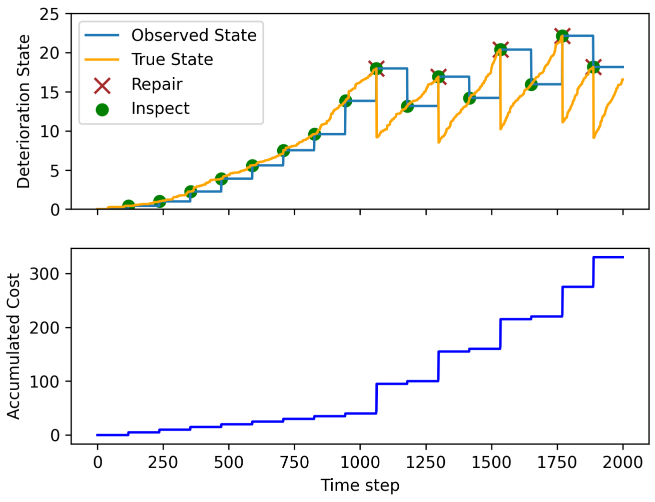

*Figure 4: Sample deployment of the Condition Based Maintenance heuristic. (Life cycle cost $C_{LC} = 330$)*

formation about the current time step, it is also unable to produce meaningful DRL policies. Conceptually, using this observation space, policies with inspections in fixed time steps could be realized. After inspections, a policy would be able immediately take action to repair the system if necessary. A CBM policy as described in section 4.5.2 can be implemented using this observation space. Agents trained in this way, however, fail to learn such a policy. This might indicate that this kind of information is not as fit for use in DRL as other representations.

Setups three and four provide observations, which retain information of past inspections, such as their value $d_l$ and age $t_i$. Agents can therefore choose actions based on information that might lie several hundreds of time steps in the past. Enabling agents trained with these observations to perform on par or better than the CBM baseline.

DRL models trained using the observation space of setup three overall seem to perform the best and most consistent. The best model produces 85% of the cost compared to the CBM baseline. Figure 3 shows a sample episode of the best policy. Note that the agent performs its first inspection only after around 750 time steps, where the condition of the component is most likely still very good. After this, the agent inspects more frequently and repairs when the state reaches a value of about 20. Repairs performed by this agent are always accompanied by an inspection immediately after the repair. This updates the last inspection value $d_l$ and allows the agent to react to the new deterioration state. For comparison, Figure 4 shows a sample of the same episode using the CBM policy. Here, repairs are not accompanied by inspections, because this policy solely relies on time intervals to decide when to perform the next inspection. It can also be observed that many inspections are performed in the beginning of the episode, even though the state of the system is still very good, which increases overall cost. The CBM policy also needs one additional repair to keep the system maintained, as repairs are done at lower deterioration state values compared to the DRL policy, which decreases their efficiency.

While setup four contains the additional information of the current time step within an episode, it seems that this hinders the training process more than the extra information could improve the policy. Intuitively, the time step might allow the agent to create policies that vary inspection and repair frequencies depending on the current progress within the episode. The extra information does however not impact final performances very much, with only slight increases in median and worst average LCC, and a comparable best average LCC.

None of the setups can reach the CBM-FI policy, which is expected. Even an exact solution would not be able to reach this performance without at least performing one inspection, in order to estimate the parameter $a$ and the deterioration state $D(t)$.

## 7. Conclusion

To investigate the effects of choosing different observation representations for I&M problems, we conducted several experiments with different observation spaces and compared the resulting optimized maintenance strategies to relevant baselines. Our experiments have shown that agents with only access to current inspection values are unable to learn a meaningful policy in our problem setting. By contrast, agents trained on systems with access to the last known inspection value and time since inspection, are able to find policies that perform comparable and even better than the CBM baseline. This shows that simply providing observations gained through inspections may not lead to good performance of DRL in environments, in which optimal inspection and maintenance decisions rely on information that can lie many time steps in the past. Observations, which retain information on the result and time of inspections, provide richer information to the agent. This allows the creation of more complex DRL maintenance policies able to dynamically react to given observations, without providing belief states or any additional knowledge about the model. To continue the investigation of observation representations, one direction for future research is augmenting the observations to include multiple previous inspection values and times, as well as also providing timing information of previous repairs.

## 8. Acknowledgements

The research presented is supported by the TUM Georg Nemetschek Institute Artificial Intelligence for the Built World.

## 9. REFERENCES

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

Andriotis, C. and Papakonstantinou, K. (2021). "Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints." *Reliability Engineering & System Safety*, 212, 107551.

Bellman, R. (1957). "A markovian decision process." *Journal of Mathematics and Mechanics*, 679–684.

Lin, L.-J. and Mitchell, T. M. (1993). "Reinforcement learning with hidden states." *From Animals to Animats*, 2, 271–280.

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., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). "Playing atari with deep reinforcement learning." *arXiv preprint arXiv:1312.5602*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). "Mastering the game of go without human knowledge." *Nature*, 550(7676), 354–359.

Straub, D. and Faber, M. H. (2006). "Computational aspects of risk-based inspection planning." *Computer-Aided Civil and Infrastructure Engineering*, 21(3), 179–192.

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