# Life-cycle policies for large engineering systems under complete and partial observability

C. P. Andriotis
*PhD Candidate, Dept. of Civil & Environmental Engineering, The Pennsylvania State University, University Park, PA, USA*

K. G. Papakonstantinou
*Assistant Professor, Dept. of Civil & Environmental Engineering, The Pennsylvania State University, University Park, PA, USA*

ABSTRACT: Management of structures and infrastructure systems has gained significant attention in the pursuit of optimal inspection and maintenance life-cycle policies that are able to handle diverse deteriorating effects of stochastic nature and satisfy long-term objectives. Such sequential decision problems can be efficiently formulated along the premises of Markov Decision Processes (MDP) and Partially Observable Markov Decision Processes (POMDP), which describe agent-based acting in environments with Markovian dynamics, equipped with rewards, actions, and complete or partial observations. In systems with relatively low dimensional state and action spaces, MDPs and POMDPs can be satisfactorily solved using different dynamic programming algorithms, such as value iteration with or without synchronous updates and point-based approaches for partial observability cases. However, optimal planning for large systems with multiple components is computationally hard and severely suffers from the curse of dimensionality. Namely, the system states and actions can grow exponentially with the number of components, in the most general and adverse case, making the problem intractable by conventional dynamic programming schemes. In this work, Deep Reinforcement Learning (DRL) is implemented, with emphasis in the development and application of deep architectures, suitable for large engineering systems. The developed approach leverages component-wise information to prescribe component-wise actions, while maintaining global optimality on the system level. Thereby, the system life-cycle cost functions are efficiently parametrized for large state and action spaces through nonlinear approximations, enabling adept planning in complex decision problems. Results are presented for a multi-component system, evaluated against various condition-based policies.

## 1. INTRODUCTION

Sequential long-term optimal or near-optimal decision making is vital in infrastructure management. In large multi-component systems, this process entails significant computational and modeling challenges, as the possible states, actions and action times create vast combinatorial decision spaces. This curse of dimensionality impedes detailed implementation of traditional threshold-based and time-based decision frameworks, as well as more advanced Markov Decision Process (MDP) and Partially Observable Markov Decision Process (POMDP) schemes, without plain modeling simplifications that can, however, in many cases compromise optimal and realistic solutions. This work addresses the need for advanced decision-making frameworks that have the capacity to provide comprehensive management solutions for complex engineering domains. We articulate the decision problem along the premises of MDPs and POMDPs and provide an efficient Deep Reinforcement Learning (DRL) approach for their solution, in order to support maintenance and inspection planning in multi-component systems which form large state and action spaces.

Standard MDP and POMDP solution procedures are well suited for certain low-dimensional domains, typically corresponding to system components or simple systems with

tractable system state and action spaces (Papakonstantinou & Shinozuka, 2014b; Papakonstantinou, et al., 2016). However, they can often become impractical when large-scale multi-component domains are considered, since Markovian transition matrices become extremely large, and computational complexity of action-value function evaluations per decision step severely deteriorates. For example, a stationary system with 20 components, 5 states and 5 actions per component is fully described by nearly $10^{14}$ states and actions! This issue renders the problem practically intractable by any conventional solution scheme or advanced MDP or POMDP algorithm (Shani, et al., 2013), unless domain knowledge and simplified modeling can possibly suggest drastic state and action space moderations. A straightforward modeling approach facilitating convenient state and action spaces reductions is to exploit similarity of components. This assumption may suffice in systems where components are highly homogeneous and structurally independent, e.g. wind farms where turbines can be assumed to share similar properties with negligible component interactions (Memarzadeh, et al., 2014). In other cases, it is feasible to properly engineer macro-states and -actions in order to achieve practical problem-specific state and action space reductions, e.g. in (Fereshtehnejad & Shafieezadeh, 2017), or to take advantage of compressed state space representations that enable the applicability of traditional solvers (Poupart, 2005).

Reinforcement Learning (RL) is able to alleviate the curse of dimensionality related to the state space (Wiering & Van Otterlo, 2012). In RL, the decision-maker, also called the agent, does not synchronously update the value function over the entire state space, but merely conducts updates at certain states that are visited while probing the environment, without the need for a priori explicit knowledge of the environment characteristics and transition dynamics. Classical RL techniques have also been implemented for maintenance of engineering systems, providing approximate solutions in various settings, e.g. in (Durango-Cohen, 2004). Unfortunately, RL exhibits several limitations in practice when deployed in high-dimensional and complex stochastic domains, mainly manifesting algorithmic instabilities with solutions that significantly diverge from optimal regions, or exhibiting slow value updates at infrequently visited states.

Nonetheless, with the aid of deep learning, RL has recently achieved remarkable breakthroughs in autonomous control and decision-making. DRL has brought unprecedented algorithmic capabilities in providing adept solutions to a great span of complex learning and planning tasks, even outperforming human experts in several domains (Mnih, et al., 2015). Similarly to the great progress that deep learning has enabled in machine learning and artificial intelligence (Goodfellow, et al., 2016), DRL agents are capable of discovering meaningful parametrizations in immense state spaces through appropriate deep neural network architectures, and learning near-optimal control policies by interacting with the environment.

In this work, we propose a DRL solution scheme for stochastic control and management of large engineering systems, based on our recently introduced *Deep Centralized Multi-agent Actor Critic* (*DCMAC*) approach (Andriotis & Papakonstantinou, 2018). DCMAC is a deep off-policy actor-critic algorithm with experience replay, providing efficient life-cycle policies in otherwise practically intractable problems of multi-component systems operating in high-dimensional state and action spaces. DCMAC is favorably constructed for providing comprehensive individualized component- and subsystem-level decisions, while maintaining and improving a centralized value function for the entire system. In DCMAC, two deep networks (actor and critic) co-exist and are trained in parallel, based on environment signals and replayed transitions that are retrieved from the agent's experience. DCMAC can handle complex systems under complete and incomplete informa-

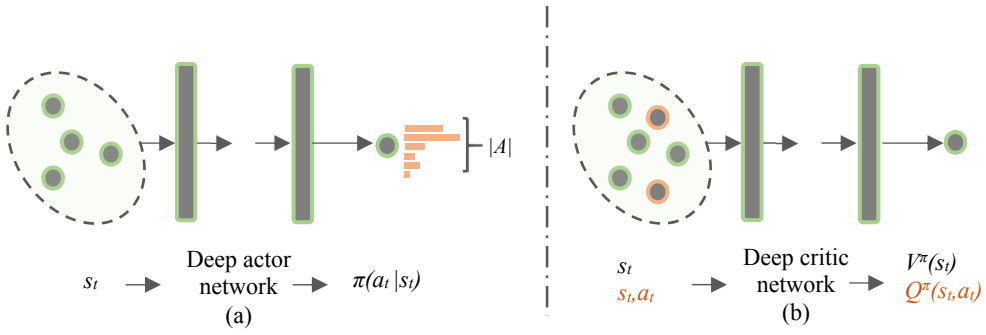


*Figure 1: (a) Actor networks approximate the policy distribution over all available actions using the state as input. (b) Critic networks approximate the V-function using the state as input or the Q-function using the state and selected action as input.*

tion, in both MDP and POMDP settings, and is evaluated against various optimized baseline policies, in system settings featuring highly heterogeneous component structures with varying cost functions and transitions, component interactions, and non-stationary environment dynamics.

## 2. DEEP POLICY GRADIENTS

The key concept of DRL is to combine RL with deep neural networks that parametrize the action-value, value and policy functions involved in MDP and POMDP formulations. A concise review can be found in (Andriotis & Papakonstantinou, 2018). DRL can be devised in two major families, namely Deep Q-Network (DQN) (Mnih, et al., 2015) and deep policy gradient approaches, e.g. (Schulman, et al., 2015; Wang, et al., 2016). In this work, we develop our algorithm and architecture along the lines of deep policy gradient methodologies, as they can support our suggested specialized DRL scheme that allows for drastically improved scaling of system actions. Deep policy gradient algorithms are established on the basis of the policy gradient theorem (Sutton, et al., 2000) and approximate the policy function with a deep neural network. The policy function can be generally described by a discrete probability distribution, $\pi(a_t \mid s_t)$: $S \to P(A)$, where $S$ is the set of states, $s_t$, and $A$ is the set of actions, $a_t$, at different time steps $t$. Policy updates follow the policy gradient, given by:

$$g_{\theta^\pi} = \mathbb{E}_{s_{t\geq 0}, a_{t\geq 0}}\left[\sum_{t\geq 0} \nabla_{\theta^\pi} \log \pi\left(a_t \mid s_t, \theta^\pi\right) Q^\pi\left(s_t, a_t\right)\right] \quad (1)$$

where $\theta^\pi$ is a set of real-valued parameters and $Q^\pi$ is the action value function. The action value function in Eq. (1) can be replaced by the advantage function (Wang, et al., 2016). The advantage function, $A^\pi$, can be seen as a measure of how advantageous an action at each state is, defined as:

$$A^\pi\left(s_t, a_t\right) = Q^\pi\left(s_t, a_t\right) - V^\pi\left(s_t\right) \quad (2)$$

where $V^\pi$ is the value function. In Eq. (1), computation of the policy gradient requires gradient $\nabla_{\theta^\pi} \log \pi\left(a_t \mid s_t, \theta^\pi\right)$ that is given from the network of Figure 1(a). Except for that, a complementary estimate related to a certain value of Eq. (2) is required. Generally, either one of the value, $V^\pi$, or the action-value, $Q^\pi$, function is approximated to provide the necessary estimates for Eq. (2), as depicted in the critic network of Figure 1(b). As shown in the figure, in case of an action-value critic, the actions are also required as input to the network (in red). This class of methods is thus referred to as actor-critic methods, as the parameters of the policy approximator (actor) are trained with the aid of a value approximator (critic). Other methods to

compute the relevant value use Monte Carlo estimates from experience trajectories (Schulman, et al., 2015). Methods relying on function approximations reduce variance but may suffer from increased bias, whereas methods relying on sampling have low bias but high variance. To trade-off bias and variance, some methods in the literature combine both techniques. Along these lines, as proposed in (Mnih, et al., 2016), an approximate form of the advantage function in Eq. (2) can be given by:

$$A^{\pi}\left(s_t, a_t \mid \theta^{V}\right) \simeq \sum_{i=0}^{k-1} \gamma^{i} r\left(s_{t+i}, a_{t+i}\right) + \gamma^{k} V^{\pi}\left(s_{t+k} \mid \theta^{V}\right) - V^{\pi}\left(s_t \mid \theta^{V}\right) \tag{3}$$

where $\gamma$ is the discount factor, $r$ the rewards, $\theta^{V}$ a set of real-valued parameters, and $k$ the length of the sampled trajectory the agent actually experienced while probing the environment. Another important distinction in the computation of the policy gradient is also the differentiation between on-policy and off-policy approaches. The gradient in Eq. (1) corresponds to on-policy algorithms. On-policy algorithms are sample inefficient, as opposed to their off-policy counterparts (Wang, et al., 2016). An efficient method to compute off-policy gradient estimators with samples generated by a behavior policy different than $\pi$ is using importance sampling. In this case, Eq. (1) becomes (Degris, et al., 2012):

$$g_{\theta^{\pi}} = \mathbb{E}_{s_t, a_t}\left[ w_t \nabla_{\theta^{\pi}} \log \pi\left(a_t \mid s_t, \theta^{\pi}\right) A^{\pi}\left(s_t, a_t\right) \right] \tag{4}$$

with $w_t$ denoting the importance sampling weight. Although this estimator is unbiased, its variance is high due to the arbitrarily large values $w_t$ can practically take. Bounded importance weights through truncated importance sampling is a standard approach in these cases.

## 3. DEEP CENTRALIZED MULTI-AGENT ACTOR CRITIC (DCMAC)

Maintenance and inspection of engineering systems comprises great challenges regarding scaling of large problems with multiple components and component states and actions. In the most comprehensive control cases, the problem should be ideally formulated in state and action spaces that scale exponentially with the number of components, which makes it practically intractable by conventional planning and learning algorithms, without simplified modeling approaches that reduce complexity. DRL can provide a valuable framework towards advanced management solutions in high-dimensional domains. However, although DQNs and deep policy gradients provide great parametrization capabilities for vast state spaces, similarly vast discrete action spaces are difficult to model, as they are related to the dimensionality of the output layer of the Q- and the actor-networks, respectively, making the involved parameters hard to train.

To simultaneously tackle both the state and action scalability issues, we use a different architecture here, the Deep Centralized Multi-agent Actor Critic (DCMAC) (Andriotis & Papakonstantinou, 2018), essentially modifying deep policy gradient to support a specific action probability structure that can drastically alleviate the complexity related to the output layer. This structure assumes that component actions, as well as various possible sub-system actions (compound actions with effects on greater parts of the system) are conditionally independent of each other, given their state and the state of all others, namely the state of the system:

$$\pi\left(a_t^{(1)}, a_t^{(2)}, ..., a_t^{(n)} \mid s_t\right) = \prod_{i=1}^{n} \pi\left(a_t^{(i)} \mid s_t\right) \tag{5}$$

where $n$ is the number of *control units*, which include components and combined sub-system parts on which individualized actions apply. This technically means that every control unit is seen as an autonomous agent that utilizes centralized system-state information to decide about its actions. The benefit of this representation becomes clear when we substitute Eq. (5) in Eq. (4), to obtain the policy gradient:

$$g_{\theta^{\pi}} = \mathbb{E}_{s_t, a_t}\left[ w_t \sum_{i=1}^{n} \nabla_{\theta^{\pi}} \log \pi\left(a_t^{(i)} \mid s_t, \theta^{\pi}\right) A^{\pi}\left(s_t, a_t\right) \right] \tag{6}$$

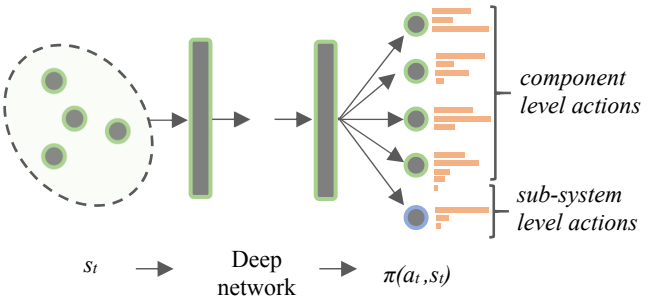


*Figure 2: Deep Centralized Multi-agent Actor network architecture.*

Eq. (6) offers a particularly compact actor representation, as shown in Figure 2, allowing system actions to scale linearly with components. Let us consider, for example, a system with 10 control units, each of which is equipped with 5 available actions to choose from. This system with a total of $5^{10}$ actions, is scaled down to an output action space of $5 \cdot 10$ dimensions with the DCMAC actor, without any approximations.

The advantage function here is evaluated with the aid of a critic that provides a value function approximation. An off-policy implementation of this advantage would require a product of $k$ importance sampling weights, as $k$ transition samples are involved. This fact, along with the required product of weights resulting from the factorized representation of the actor output could increase variance significantly. Thus, the advantage function utilized here is computed by Eq. (3) for $k=1$:

$$A^{\pi}\left(s_t, a_t\right) \simeq r\left(s_t, a_t\right) + \gamma V\left(s_{t+1} \mid \theta^{V}\right) - V\left(s_t \mid \theta^{V}\right) \quad (7)$$

Apart from the actor network, the critic network is also centralized, as implied by Eq. (6). The critic approximates the value function over the entire system space, thus providing a global measure for the DCMAC policy updates. The critic network is updated through the value function gradient:

$$g_{\theta^{V}} = \mathbb{E}_{s_t, a_t}\left[ w_t \nabla_{\theta^{V}} V^{\pi}\left(s_t \mid \theta^{V}\right) A^{\pi}\left(s_t, a_t\right) \right] \quad (8)$$

DCMAC operates off-policy, as indicated in Eqs. (6) and (8). This choice is driven by the fact that, as previously outlined, off-policy algorithms are more sample efficient than their on-policy counterparts. Low sample-complexity is critical in large engineering systems, as samples are often drawn from computationally expensive nonlinear and/or dynamic structural models and demanding finite element simulations, thus taking advantage of this attribute is of paramount importance. In addition, as in most standard DRL approaches, experience replay is utilized here.

Following the concept of belief-MDPs for POMDPs, any DRL network can be implemented as a belief DRL network, if a model for *component* transition, $p\left(s_{t+1}^{(i)} \mid s_t^{(i)}, a_t\right)$, and observation matrices, $p\left(o_t^{(i)} \mid s_t^{(i)}, a_t\right)$, is known, where $o_t^{(i)}$ is the observation outcome for component $i$ at time $t$. This is a valid and frequently met assumption for observations in engineering systems, since inspection methods and monitoring instruments are in general accurate up to known levels of precision. In this case, all Eqs. (1)-(8) hold, except that states are substituted by probability distributions over states (beliefs), which are continuously updated over the life-cycle, based on the selected actions and observations. This idea has also been applied for DQNs in (Egorov, 2015), where it is shown in small, benchmark POMDP applications that belief DQNs are able to provide adequate nonlinear approximations of the piecewise linear value function over the belief space.

## 4. RESULTS

In this paper, a system with 10 components and 4 damage states per component is considered. State 1 corresponds to *no damage*, state 2 to *minor damage*, state 3 to *severe damage*, and state 4 to *failure*. Interstate transitions are described by different triangular transition matrices, consistent with deteriorating environments, whereas component cost functions also vary, as described in (Andriotis & Papakonstantinou, 2018). Transitions are non-stationary depending on the degradation rate of the component. Combined behavior of individual components is considered to trigger failure modes of different severity,

*Table 1: System life-cycle cost estimates and 95% confidence intervals for different inspection accuracies (normalization with respect to MDP DCMAC solution).*

| Observability | DCMAC | CBM-I | CBM-II | CBM-III | CBM-IV |
|---|---|---|---|---|---|
| $p = 1.0$ | 1.0000 ± 0.0077 | 1.4433 ± 0.0098 | 1.2268 ± 0.0062 | 1.4433 ± 0.0098 | 1.2200 ± 0.0109 |
| $p = 0.9$ | 1.0442 ± 0.0074 | 1.5296 ± 0.0092 | 1.2877 ± 0.0137 | 1.4821 ± 0.0093 | 1.2466 ± 0.0092 |
| $p = 0.8$ | 1.0790 ± 0.0076 | 1.6132 ± 0.0093 | 1.2863 ± 0.0125 | 1.5358 ± 0.0095 | 1.2793 ± 0.0129 |
| $p = 0.7$ | 1.1036 ± 0.0081 | 1.6855 ± 0.0094 | 1.2833 ± 0.0118 | 1.5767 ± 0.0089 | 1.2710 ± 0.0114 |

following a k-out-of-n activation function. More specifically, two discrete modes are considered; the first is activated when the number of components in a state greater than state 3 exceeds 50% of the total number of system components, and the second when the number of components in state 4 is more than 30%. Component-wise costs are penalized by a factor of 2.0 and 12.0, when modes (i) and (ii) are activated, respectively. As such, higher states of damage continue to be less desirable from a component perspective, whereas system modes (i) and (ii) reveal some states of severe damage and failure damage from a system perspective, intensifying the component state costs. Episode length is 50 years, and discount factor is $\gamma = 0.99$.

In this problem setting, 4 available actions per component are available, i.e. *do nothing* (action 1), *minor repair* (action 2), *major repair* (action 3), and *replace* (action 4). Do nothing leaves component state and degradation rate unchanged, minor repair only reduces component state by 1 with a success probability of 0.95, major repair has the same effect on state as minor repair, but in addition reduces component degradation rate by 5 years, whereas replace sends component to an as-good-as-new condition, namely back to state 1 and the initial degradation rate. All actions are assumed to have been completed before the next system transition. As discussed in (Papakonstantinou & Shinozuka, 2014a) such actions can effectively describe typical maintenance decisions in certain types of concrete structures. As degradation rate essentially reflects the effective age of the components, major repairs not only improve component states but also suspend their aging process, which, in the uncontrolled case, due to the non-stationarity of the environment, intensifies transitions to more severe states as time progresses.

The system is examined both under complete and partial observability. In the former case, the problem is formulated as a MDP, which means that the agent observes the exact state of the system at every decision step. In the latter case, observations do not reveal the exact state of the system with certainty, so the agent forms a belief about its state. Four observation cases are considered, reflecting the accuracy of the inspection instruments, for accuracy levels $p = 1.0, 0.9, 0.8, 0.7$. The value of $p$ indicates the probability of observing the correct component state, thus $p = 1.0$ defines the MDP case. For more details about the full observation matrix see (Andriotis & Papakonstantinou, 2018).

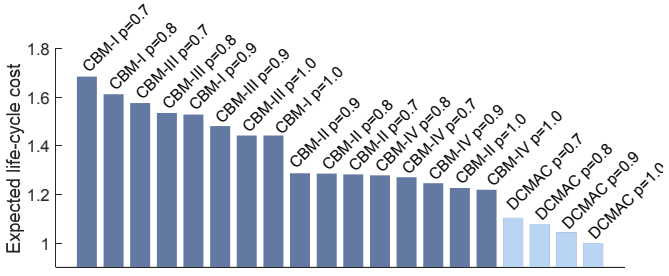


*Figure 3: Expected life-cycle cost estimates of DCMAC solutions and baseline policies, for different observability accuracies.*

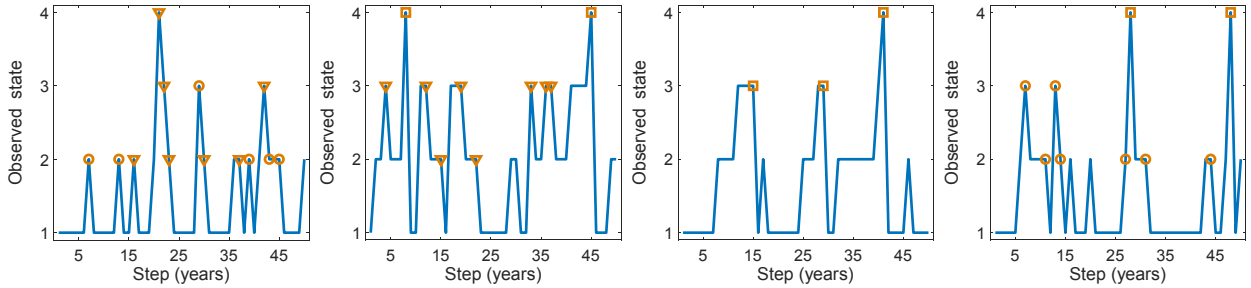


*Figure 4: Observed damage states and respective sequential actions for 4 components, for a single life-cycle policy realization of the entire system and p=0.90 (markers '○','▽', '□' indicate actions 2, 3, 4 respectively).*

For $p = 0.9, 0.8, 0.7$, partial observability is merely applied to component damage states, whereas the degradation rate is considered to be known, thus overall a MOMDP environment is established (Papakonstantinou, et al., 2018). For this system, an exact solution is not available due to the dimensionality of total states and actions. As a result of different component damage state configurations, combined with different possible deterioration rates, the total number of states is more than $10^{23}$, whereas the total number of system actions are more than $10^{6}$. This total number of states and actions in a system of 10 components is indicative of how the curse of dimensionality can impede comprehensive maintenance and inspection solutions in multi-component systems, and underscores the imperative algorithmic strengths DRL with DCMAC provides. The implemented network specifications are discussed in detail in (Andriotis & Papakonstantinou, 2018).

To assess and trace the quality of the DCMAC solutions, we formulate and evaluate 4 straightforward condition-based maintenance (CBM) baseline policies. CBM-I is a replacement policy (best states to replace), whereas CBM-II also allows for repair actions. CBM-III and CBM-IV optimize state-action pairs, as in CBM-I and CBM-II respectively, however, they both also account for two additional decision variables, namely the periodic time interval of interventions, as well as the number of components to be maintained at each intervention step. The specific components to be maintained are selected by component ranking based on their probability of failure. The results of the best DCMAC runs and the abovementioned baselines are presented in Table 1, where values are calculated through $10^3$ Monte Carlo policy realizations and are normalized with respect to the results of the MDP case, namely for $p = 1.0$. The baselines are optimized in all observability cases, and their implementation in the partially observable domains is based on the observed damage state. For the various observability levels examined, DCMAC discovers substantially better policies. The competence of DCMAC policies is also verified by the fact that they are superior to the baseline ones even when these operate under better observability conditions, as can be seen in Figure 3. As results indicate, observability is a crucial factor for the attained life-cycle cost in near-optimal POMDP solutions (Andriotis, et al., 2019). This pattern of increased life-cycle costs as observations become less precise is also in general apparent in the baseline policies, as shown both in Table 1 and Figure 3. In Figure 4, a policy sample is depicted, for $p = 0.90$, for 4 of the components. Each component contributes to the overall system policy with its detailed sequential decision life-cycle plan, which is adjusted according to its specific deterioration dynamics, cost functions, and related system interactions.

## 5. CONCLUSIONS

In this work, the problem of planning efficient life-cycle maintenance and inspection policies for large deteriorating engineering systems is formulated within the premises of MDPs and POMDPs. To solve the problem, a new DRL

approach is suggested, which has the modeling capacity to handle immense state and action spaces, alleviating the curse of dimensionality, and to provide competent near-optimal solutions to otherwise intractable learning problems. Our developed DCMAC approach is concisely presented and explained, and numerical results showcase its exceptional capabilities in the most general and complex settings, described by non-stationary, multi-state, multi-component, stochastic engineering environments, either under complete or partial observability.

## ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under CAREER Grant No. 1751941.

## 6. REFERENCES

Andriotis, C. P., & Papakonstantinou, K. G. (2018). Managing engineering systems with large state and action spaces through deep reinforcement learning. *arXiv preprint, arXiv:1811.02052*.

Andriotis, C. P., Papakonstantinou, K. G. & Chatzi, E. N. (2019). Value of structural health monitoring quantification in partially observable stochastic environments. *Structural Safety,* Under review.

Degris, T., White, M. & Sutton, R. (2012). Off-policy actor-critic. *arXiv preprint, arXiv:1205.4839.*

Durango-Cohen, P. (2004). Maintenance and repair decision making for infrastructure facilities without a deterioration model. *Journal of Infrastructure Systems, 10*(1), 1-8.

Egorov, M. (2015). *Deep Reinforcement Learning with POMDPs.* Technical Report, Stanford University.

Fereshtehnejad, E., & Shafieezadeh, A. (2017). A randomized point-based value iteration POMDP enhanced with a counting process technique for optimal management of multi-state multi-element systems. *Structural Safety, 65*, 113-125.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* Cambridge: MIT Press.

Memarzadeh, M., Pozzi, M., & Kotler, Z. (2014). 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., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *International Conference on Machine Learning*, 1928-1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., . . ., Petersen, S. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.

Papakonstantinou, K. G., & Shinozuka, M. (2014a). Optimum inspection and maintenance policies for corroded structures using partially observable Markov decision processes and stochastic, physically based models. *Probabilistic Engineering Mechanics, 37*, 93-108.

Papakonstantinou, K.G., & 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.

Papakonstantinou, K. G., Andriotis, C. P., & Shinozuka, M. (2016). Point-based POMDP solvers for life-cycle cost minimization of deteriorating structures. *Proceedings of the 5th International Symposium on Life-Cycle Civil Engineering (IALCCE 2016)* (p. 427). Delft, The Netherlands: CRC Press.

Papakonstantinou, K. G., Andriotis, C. P., & 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.

Poupart, P. (2005). *Exploiting structure to efficiently solve large scale partially observable Markov decision processes.* PhD Dissertation, University of Toronto.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. *International Conference on Machine Learning*, 1889-1897.

Shani, G., Pineau, J. & Kaplow, R. (2013). A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems,* pp. 1-51.

Sutton, R., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 1057-1063.

Wang, Z., Bapst, V., Heess, N., Munos, R., Kavukcuoglu, K., & De Freitas, N. (2016). Sample efficient actor-critic with experience replay. *arXiv preprint, arXiv:1611.01224*.

Wiering, M. & Van Otterlo, M. (2012). *Reinforcement learning - Adaptation, learning, and optimization.* Springer.