Assessing the Optimality of Decentralized Inspection and Maintenance Policies for Stochastically Degrading Engineering Systems

Bhustali, P.; Andriotis, C.

Publication date
2023

Document Version
Submitted manuscript

Published in
BNAIC BeNeLearn 2023

Assessing the Optimality of Decentralized Inspection and Maintenance Policies for Stochastically Degrading Engineering Systems

Prateek Bhustali[✉][0009-0000-6027-8571] and Charalampos P. Andriotis[0000-0002-0140-5021]

Faculty of Architecture and the Built Environment, TU Delft, Netherlands
{P.Bhustali,C.Andriotis}@tudelft.nl

Abstract. Long-term inspection and maintenance (I&M) planning, a multi-stage stochastic optimization problem, can be efficiently formulated as a partially observable Markov decision process (POMDP). However, within this context, single-agent approaches do not scale well for large multi-component systems since the joint state, action and observation spaces grow exponentially with the number of components. To alleviate this curse of dimensionality, cooperative decentralized approaches, known as decentralized POMDPs, are often adopted and solved using multi-agent deep reinforcement learning (MADRL) algorithms. This paper examines the centralization vs. decentralization performance of MADRL formulations in I&M planning of multi-component systems. Towards this, we set up a comprehensive computational experimental program focused on k-out-of-n system configurations, a common and broadly applicable archetype of deteriorating engineering systems, to highlight the manifestations of MADRL strengths and pathologies when optimizing global returns under varying decentralization relaxations.

Keywords: Inspection and maintenance planning · Decentralized partially observable Markov decision processes · Multi-agent deep reinforcement learning · Actor-critic methods · Stochastic deterioration

1 Introduction

Inspection and maintenance (I&M) planning of deteriorating engineering systems, such as bridges, roads, aircraft, etc., is an optimization problem of seeking a policy that minimizes the expected life-cycle cost over a given time horizon. The problem is particularly challenging for engineering systems with multiple heterogeneous components that exhibit distinct system-level and component-level behaviours because the space of possible policies is intractably large. To make the problem tractable, risk-based and condition-based approaches use heuristics to confine the policy space over which the objective is minimized [44,12,12]. Although these methods provide fast and explainable policies, it is well understood that they can be far from optimal [4,32].

Alternatively, since I&M is a problem of decision-making under uncertainty, several works leverage the partially observable Markov decision process (POMDP) framework to model the environment and state uncertainties inherent to real-world systems [32,32,24,8,29,7,26]. This enables us to leverage single-agent deep reinforcement learning (SADRL) algorithms, which unlike heuristics, access the entire policy space. They also circumvent many complexity limitations of traditional Bellman backup operator-based POMDP solvers [37,38] when applied to I&M planning problems [33,6].

Although, in theory, neural networks can tackle arbitrarily high-dimensional spaces, SADRL approaches do not scale well under practical computational constraints because the joint space of states, observations and action spaces grow *exponentially* with the number of agents. A natural way to address this is to decentralize the problem by assigning an agent to each component or subsystem, thus articulating the problem as a cooperative multi-agent task. This relaxation is an extension of the single-agent POMDP to multi-agent systems and is formally known as a decentralized POMDP (Dec-POMDP) [30,31,3]. Likewise, we can move from SADRL to multi-agent deep reinforcement learning (MADRL) to solve the Dec-POMDP.

On the one hand, decentralization in reinforcement learning is key to addressing scalability. On the other hand, it can introduce certain side effects, such as environment non-stationarity, equilibrium selection, multi-agent credit assignment, and other issues, which encumber the learning task and may hinder convergence to optimal policies [14,27,11]. These can become potentially predominant in I&M settings due to strong global reward signals, risk costs, and other dependencies common in the mathematical description of deteriorating engineering systems. It, therefore, remains to be understood how the various multi-agent learning pathologies manifest in various I&M planning settings.

In this work, we focus on understanding the optimality characteristics of the three major MADRL paradigms: centralized training with centralized execution (CTCE), centralized training with decentralized execution (CTDE) and decentralized training with decentralized execution (DTDE) in the context of I&M planning. Specifically, we study this through the lens of a 5-component k-out-of-n system, a common and broadly applicable archetype of deteriorating engineering systems, modelled as a (Dec-)POMDP. This allows us to conduct extensive numerical experiments with several MADRL architectures. Finally, we obtain empirical insights on the performance of decentralized algorithms, which can ultimately impede the convergence of MADRL algorithms to global optima.

2 Related Work

Some recent works in the maintenance planning context have leveraged the Dec-POMDP framework to tackle the component multiplicity challenge. Andriotis & Papakonstantinou exploit the independence of agent actions and factorize the policy network output agent-wise, effectively converting the Dec-POMDP into a multi-agent POMDP (MPOMDP) [4,5] (similar to [18]). Although action factorization curbs the exponential growth of the policy network output to linear in the number of agents, the policy input space can still grow exponentially, unless dependencies are properly broken down into likewise factorized representations [4,5,28].

CTDE and DTDE are two practical paradigms for solving Dec-POMDPs, which try to address this challenge [31,2,17,42]. Various CTDE approaches such as COMA [13], QMIX [36], FACMAC [35], VDN [39] have been proposed that can address the credit assignment problem. Leroy et al. extensively compare the performance of various CTDE approaches in the maintenance planning context for k-out-of-n systems with varying n and demonstrate empirically the limitations of current CTDE algorithms in a large system with more than $n = 50$ components [23].

An extensive empirical comparison of the CTDE and DTDE paradigms in cooperative settings has been carried out on standard MARDL environments by Papoudakis et al. to benchmark the efficacy of MADRL algorithms in practice [34]. Lyu et al. provide a theoretical and empirical study on the effectiveness of centralized and decentralized critics and show that both critics have the same policy gradient in expectation. However, the variance in centralized critics is at least as much as their decentralized counterparts. They empirically demonstrate the performance of the CTDE and DTDE paradigms in several environments to highlight shortcomings of training under centralized critics [25]. Guillaume et al. were among the first to compare all MADRL paradigms to solve Dec-POMDPs [9]. They empirically compare CTCE, CTDE and DTDE paradigms on benchmark problems and use RNN-based policy networks, among others, to encode the action-observation history.

In this work, we also compare all MADRL paradigms in the context of I&M planning, emphasizing on k-out-of-n deteriorating systems. However, unlike [9], we encode action-observation histories using beliefs over component states. We do this for three reasons: transition and observation models are often available in this context; belief-based policies are more explainable than RNN-based agents that learn to encode action-observation histories, which is particularly important in risk-sensitive settings; and learning stability is improved as training directly on the belief space is more robust to environment noise. The goal is to understand the strengths and shortcomings of MADRL approaches, specifically when solving real-world I&M planning problems.

3 Background

3.1 Dec-POMDPs

We decentralize the I&M planning problem for a multi-component system by assigning an agent to each component and requiring the agents to cooperatively minimize the inspection and maintenance cost over the system's lifetime. The objective can be extended to include risk thresholds, minimum component health states, budget constraints, etc. [5]. This decentralized cooperative multi-agent setting is formally called a Dec-POMDP [30,31], defined by the tuple

$\langle \mathcal{M}, S, \mathbf{A}, T, C, \mathbf{O}, \mathit{\Omega}, t_H, \gamma \rangle$, where $\mathcal{M} := \{1, \ldots, M\}$ is the set of agents, S is the state space, $\mathbf{A} = \times A_m$ is the joint action space, $T : S \times \mathbf{A} \times S \to [0,1]$ is the transition model, $C : S \times \mathbf{A} \to \mathbb{R}$ is the system cost model, $\mathbf{O} = \times O_m$ is the joint observation space, $\mathit{\Omega} : \mathbf{O} \times S \times \mathbf{A} \to [0,1]$ is the observation model, t_H is the time horizon, and $\gamma \in [0,1)$ is the discount factor.

The solution to the Dec-POMDP is an *optimal joint policy*, $\boldsymbol{\pi}^*$, that minimizes the expected sum of discounted costs:

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}} \left[\sum_{t=0}^{t_H-1} \gamma^t c_t | \boldsymbol{\pi} \right] \tag{1}$$

where $c_t \in C(s_t, \mathbf{a}_t)$ is the cost following the joint action $\mathbf{a}_t \in \mathbf{A}$ prescribed by the joint policy $\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_M \rangle$ in state s_t. The policy $\boldsymbol{\pi}$ can be stochastic, mapping the agent's action-observation history to a probability distribution over actions, or deterministic, mapping the agent's observation history to agents' actions. It is known that every Dec-POMDP has at least one optimal deterministic joint policy [31].

In several I&M planning scenarios, the system components deteriorate independently, enabling a factorization of the state space, $S = \times S_m$, transition model, $T_m : S_m \times A_m \times S_m \to [0,1]$ and observation model, $\mathit{\Omega}_m : S_m \times A_m \times O_m \to [0,1]$ making it a Dec-POMDP with transition and observation independence [15,21]. In scenarios where components are correlated or dependent, they can be transformed into independent representations through proper reconstruction of the underlying dynamic Bayesian network [28].

The deterioration models are often available or learnable offline, which enables us to maintain beliefs over the states of each agent [4,45,19]. When an inspection action is taken, we use the observation o_m to update the belief over a component's state s'_m using the Bayes' rule:

$$b_{m,t+1}(s'_m) = \frac{\mathit{\Omega}_m(o_m|s'_m, a_m) \cdot \sum_{s_m \in S_m} T_m(s'_m|s_m, a_m) b_{m,t}(s_m)}{\sum_{s'_m \in S_m} \mathit{\Omega}_m(o_m|s'_m, a_m) \sum_{s_m \in S_m} T_m(s'_m|s_m, a_m) b_{m,t}(s_m)} \tag{2}$$

An agent's policy, typically tied to a specific component, maps component beliefs to actions $\pi_m : B_m \to A_m$.

3.2 Baseline Performance

We establish baseline performance using the classical time-periodic inspections with condition-based maintenance (TPI-CBM) approach as in [16,24,5]. In this strategy, we inspect components at fixed time intervals ($\mathit{\Delta} t_{\text{insp}}$) and, at the inspection step, take maintenance actions based on observations ($\boldsymbol{o}_{\text{insp}}$), denoted by a sub-policy $\hat{\pi}(\boldsymbol{o}_{\text{insp}})$.

$$\boldsymbol{\pi}^*_{TC}(\mathit{\Delta} t^*_{\text{insp}}, \hat{\pi}^*(\boldsymbol{o}_{\text{insp}})) = \underset{\mathit{\Delta} t_{\text{insp}}, \hat{\pi}(\boldsymbol{o}_{\text{insp}})}{\arg\min} \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}_{TC}} \left[\sum_{t=0}^{t_H-1} \gamma^t c_t | \boldsymbol{\pi}_{TC} \right] \tag{3}$$

Table 1. Single- and multi-agent DRL architectures studied in this work.

Paradigm	*Mathematical Framework*	*Algorithm*	*Observation*	*Action*	*Critic*	*Actor*
CTCE	POMDP	**JAC** (SADRL)	Joint	Joint	Centralized	Shared
	MPOMDP	**DCMAC**		Factored		Shared
		DDMAC		Factored		Separate
CTDE	Dec-POMDP	**IACC**	Independent	Independent	Centralized	Separate
		IACC-PS		Independent		Shared
DTDE		**IAC**	Independent	Independent	Decentralized	Separate
		IAC-PS		Independent		Shared

Intuitively, we formulate the inspection and maintenance planning problem as a combinatorial optimization problem by evaluating the objective defined in Eq.(3) for each policy $\boldsymbol{\pi}_{TC}(\Delta t_{\text{insp}}, \hat{\pi}(\boldsymbol{o}_{\text{insp}}))$ using Monte Carlo rollouts. Additionally, failed components are immediately repaired, enabling the strategy to retrogress to simple corrective maintenance when necessary.

3.3 Multi-agent Deep Reinforcement Learning (MADRL)

In this work, we study the 7 variants of MADRL as listed in Table 1. In all of them, we employ an off-policy actor-critic approach as in ACER [41] for sample efficiency. Following standard nomenclature [17], we categorize these algorithms based on information available during training and execution, as explained below and summarized in Table 1.

Centralized training with centralized execution (CTCE). This paradigm essentially converts the Dec-POMDP either into a single-agent POMDP to directly learn the mapping between the joint belief space and joint action space or a multi-agent POMDP (MPOMDP) to learn the mapping between the joint belief space and the agent-wise factored action space. Although the single-agent POMDP approaches are not scalable, they theoretically encompass global optima in their solution space. MPOMDP solutions relax the requirement for joint action spaces and can be seen as a semi-CTCE paradigm that can approximate well the POMDP solution space under mild conditions. Specifically, we consider three actor-critic algorithmic approaches:

- **Joint actor-critic (JAC)**, where the actor learns the joint stochastic policy $\boldsymbol{\pi}(\mathbf{a}|\mathbf{b};\theta)$, mapping the joint beliefs to joint actions, and the critic learns the value function $V^{\boldsymbol{\pi}}(\mathbf{b};\phi)$ under that policy.
- **Deep centralized multi-agent actor-critic (DCMAC)**, where we exploit the factorizable nature of the joint action space and learn a policy for each agent that maps the joint belief to agents' actions, $\boldsymbol{\pi} = \langle \pi(a_m|\mathbf{b};\theta)\rangle_{m=1}^{M}$, and a centralized critic $V^{\boldsymbol{\pi}}(\mathbf{b};\phi)$ guides each agent as in [4,18].

- **Deep decentralized multi-agent actor-critic (DDMAC)**: Like DC-MAC, we exploit the factorizable action space but employ independent networks for each agent mapping the joint belief to component action, $\boldsymbol{\pi} = \langle \pi(a_m|\mathbf{b};\theta_m)\rangle_{m=1}^{M}$, and a centralized critic $V^{\boldsymbol{\pi}}(\mathbf{b};\phi)$ guides each agent as introduced in [5]. This approach aims to model policy distributions independently, thus alleviating potential training complexities stemming from parameter sharing in the centralized paradigm.

Centralized training with decentralized execution (CTDE). This paradigm assumes that the agents may have access to centralized information signals, such as joint observations, joint actions and critic gradients, only during training but act independently based on local observations during execution/inference. We consider two information accessibility scenarios:

- **Independent actor centralized critic (IACC)**, where each agent only has access to its own component's beliefs and learns the local stochastic policy $\pi_m(a_m|b_m;\theta_m)$ guided by a centralized critic $V^{\boldsymbol{\pi}}(\mathbf{b};\phi)$.
- **IACC with parameter sharing (IACC-PS)**: A special case of IACC where agents share the same policy network $\pi(a_m|b_m,m;\theta)$ but the beliefs are indexed (using one-hot encoding) to enable the network to learn distinct policies.

Decentralized training with decentralized execution (DTDE). Agents only have access to local information during training and execution and never have access to centralized information as in the previous cases. We consider two cases:

- **Independent actor-critic (IAC)**, where each agent learns a stochastic policy mapping its belief to actions $\pi_m(a_m|b_m;\theta_m)$ and is guided by a decentralized critic $V^{\pi_m}(b_m;\phi_m)$.
- **IAC with parameter sharing (IAC-PS)**, where agents share both policy networks $\pi(a_m|b_m,m;\theta)$ and critic networks $V^{\pi}(b_m,m;\phi)$ but the beliefs are indexed (using one-hot encoding) to enable distinct outputs.

Generally, the parameter-sharing approach can be extended to agents with heterogeneous action space cardinalities [40]. Several works have empirically demonstrated the benefits of parameter sharing in environments with homogeneous agents (in terms of observations and actions) [43,18,13]. However, Christianos et al. demonstrate empirically that these benefits are environment-specific, and such parameter sharing can become detrimental in certain environments [10]. Therefore, we study both variants in this work.

Decentralization presents several challenges due to the presence of other agents that impede convergence to optimal policies. From the perspective of a single agent, the presence of other agents affects [27,2,14]:

- Transitions: The environment is perceived as non-stationary by one agent due to the actions of other agents;

– Rewards: Reward received due to joint actions must be disentangled to identify an individual agent's contribution (known as multi-agent credit assignment problem), and exploratory actions by agents obfuscate the reward signal (also known as alter-exploration issue);
– Agent policy: Action selection must be coordinated when multiple optimal joint policies exist (referred to as equilibria selection). Agents can converge to a sub-optimal equilibrium because miscoordination due to unilateral deviation from an optimal equilibrium has lower gains/higher penalties than unilateral deviation from the sub-optimal equilibria. In such cases, a sub-optimal equilibrium is said to *shadow* the optimal equilibrium.

4 Experimental Setup

4.1 Environment: k-out-of-n system

This work studies the optimality characteristics of MADRL approaches, focusing on a k-out-of-n system with heterogeneous components. The k-out-of-n:G (G: good) is a common archetype of deteriorating systems in which the system is functional when at least k out of its n components are working and, therefore, $n \geq k$. The system has two notable special cases, namely $k = n$, a series system, and $k = 1$, representing a parallel component configuration. Examples of such systems include road networks, transmitters in communication networks, human kidneys, etc. [20,22].

We consider a system with $n = 5$ components and assign an agent to each component, thus $M = n$. The state space of each component (S_m) describes its range of health states, $S_m := \{s^1 = \text{no-damage}, s^2 = \text{minor-damage}, s^3 = \text{major-damage}, s^4 = \text{failure}\}$ and similarly the component-wise actions are given by $A_m := \{a^1 = \text{do-nothing}, a^2 = \text{repair}, a^3 = \text{inspect}\}$. Each component m has a unique and stationary deterioration model T^d_m, the natural deterioration of the environment (e.g. ageing due to corrosion, fatigue or other stressors):

$$\overset{T_1^d}{\begin{bmatrix} 0.82 & 0.13 & 0.05 & 0 \\ 0 & 0.87 & 0.09 & 0.04 \\ 0 & 0 & 0.91 & 0.09 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \overset{T_2^d}{\begin{bmatrix} 0.72 & 0.19 & 0.09 & 0 \\ 0 & 0.78 & 0.18 & 0.04 \\ 0 & 0 & 0.85 & 0.15 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \overset{T_3^d}{\begin{bmatrix} 0.79 & 0.17 & 0.04 & 0 \\ 0 & 0.85 & 0.09 & 0.06 \\ 0 & 0 & 0.91 & 0.09 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$\overset{T_4^d}{\begin{bmatrix} 0.8 & 0.12 & 0.08 & 0 \\ 0 & 0.83 & 0.12 & 0.05 \\ 0 & 0 & 0.89 & 0.11 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \overset{T_5^d}{\begin{bmatrix} 0.88 & 0.12 & 0 & 0 \\ 0 & 0.9 & 0.1 & 0 \\ 0 & 0 & 0.93 & 0.07 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

The above synthetic transition models capture the main characteristics of a deterioration model, namely, they are upper-triangular (component state cannot be improved without repair) and unit-mass at the terminal state (failure is an absorbing state).

The *repair* action restores the component back to state s^1, but this action only succeeds with a probability r_m and the transition model corresponding to this action is given by:

$$T^r_m(s'_m|s_m, repair) := \begin{bmatrix} 1 & 0 & 0 & 0 \\ r_m & 1-r_m & 0 & 0 \\ r_m & 0 & 1-r_m & 0 \\ r_m & 0 & 0 & 1-r_m \end{bmatrix} \tag{4}$$

where r_m is the repair accuracy of agent m, reflecting the uncertainty in the duration the action requires to be completed, or other. We use the following value of r_m for each of the m components $1, 0.9, 0.95, 0.85, 0.8$, respectively. The chance of unsuccessful repair actions aims to capture the inevitable human error during maintenance activities in practice.

We summarize the transition model for a component as follows:

$$T_m := \begin{cases} T^d_m, & \text{if } a = \text{do-nothing or } a = \text{inspect} \\ T^r_m \times T^d_m, & \text{if } a = \text{repair} \end{cases} \tag{5}$$

The cost model is divided into component-level costs (repair and inspection costs) and system-level costs. For each component m, the repair costs c^{repair}_m are 30, 90, 80, 250, 350, respectively, and inspections costs $c^{inspect}_m$ are 20, 40, 25, 50, 100, respectively. System failure leads to a penalty $c^{failure}$ equal to 3 times the sum of component repair costs. The cost model can be summarized as follows:

$$C(s,a) = \mathbb{1}_{failure}(s) \cdot c^{failure} + \sum_{m=1}^{M} \mathbb{1}_{a^2}(a_m) \cdot c^{repair}_m + \mathbb{1}_{a^3}(a_m) \cdot c^{inspect}_m \tag{6}$$

where $\mathbb{1}_y(x)$ denotes the indicator function, taking a value of 1 if x=y, and 0 otherwise. This penalty necessitates the agents to cooperatively coordinate maintenance actions to ensure global system functionality.

Each component has a unique observation model Ω_m through which the respective agent obtains imperfect observations of its true state when the inspection action is chosen and is given as follows:

$$\Omega_m := \begin{bmatrix} p_m & 1-p_m & 0 & 0 \\ \frac{(1-p_m)}{2} & p_m & \frac{(1-p_m)}{2} & 0 \\ 0 & 1-p_m & p_m & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

where p_m describes the observation accuracy of component m for the inspection action and takes values of 0.8, 0.85, 0.9, 0.95, and 0.8 for the five agents. Component failure, however, is assumed to be self-announcing, i.e. information about failed components is noise and cost-free. The objective is to minimize the operation cycle cost of the system over $t_H = 50$ with $\gamma = 0.99$.

Table 2. Comparison of the best performance observed for each algorithm (with $\pm$ indicating the 95% confidence interval when evaluating the optimization objective (1) with 10,000 MC rollouts. Bold indicates best in the respective k-out-of-n setting.

		1-out-of-5	2-out-of-5	3-out-of-5	4-out-of-5	5-out-of-5
Baseline	TPI-CBM	1485.74 ($\pm$8.43)	1498.8 ($\pm$8.62)	1654.19 ($\pm$14.11)	3998.41 ($\pm$43.51)	19728.42 ($\pm$95.45)
CTCE	JAC	365.51 ($\pm$10.90)	**735.96 ($\pm$13.59)**	1686.41 ($\pm$18.56)	**3413.96 ($\pm$33.88)**	**12595.17 ($\pm$67.70)**
	DCMAC	363.64 ($\pm$10.96)	**735.56 ($\pm$14.08)**	1789.00 ($\pm$21.25)	**3396.92 ($\pm$32.64)**	**12509.88 ($\pm$66.73)**
	DDMAC	364.10 ($\pm$10.67)	**732.62 ($\pm$13.49)**	1858.60 ($\pm$24.16)	**3384.11 ($\pm$32.95)**	12764.02 ($\pm$67.13)
CTDE	IACC	**298.01 ($\pm$7.96)**	820.21 ($\pm$13.80)	**1600.01 ($\pm$15.00)**	3460.20 ($\pm$35.57)	12685.70 ($\pm$66.53)
	IACC-PS	**285.64 ($\pm$7.99)**	833.55 ($\pm$10.28)	**1578.24 ($\pm$14.88)**	**3421.71 ($\pm$35.39)**	**12543.43 ($\pm$69.51)**
DTDE	IAC	563.99 ($\pm$21.91)	1422.94 ($\pm$18.13)	**1603.05 ($\pm$14.76)**	**3421.37 ($\pm$32.43)**	13291.65 ($\pm$73.41)
	IAC-PS	**296.57 ($\pm$8.42)**	850.90 ($\pm$16.11)	1646.18 ($\pm$14.46)	3395.78 ($\pm$34.13)	13370.26 ($\pm$76.00)

4.2 MADRL algorithms and baselines

For brevity, we only describe the algorithm for JAC in detail in Appendix 1. The same off-policy features are used for all seven multi-agent approaches. All MADRL algorithms hyperparameters are tuned on the 4-out-of-5 setting, and the best-performing ones are reported in Table 4 (see appendix). We use these tuned hyperparameters and train ten instances of each agent with random seeds on all k-out-of-n settings. To pick the best policy, we evaluate the agent periodically every 4,000 training episodes using 10,000 Monte Carlo rollouts. A large number of rollouts is needed due to the high variance of expected discounted cost Eq. (1), as also highlighted in [23].

For the TPI-CBM heuristic, the policy space has $t_H \times (|S_m| - 2) = 50 \times 3 = 150$ policies, assuming policy uniformity over components. We subtract 2 since the initial and final states' actions are fixed, i.e., do-nothing and repair, respectively. However, if we were to allow different component-wise policies, the search would grow to $150^5 = 7.5 \times 10^{10}$, rendering the combinatorial optimization of Eq. (3) intractable since evaluating the objective requires a large number of Monte Carlo evaluations. Therefore, we optimize the heuristic, assuming the same policy for all components and find an optimum over the smaller search space of 150 policies.

All code and experimental data are publicly available: https://github.com/omniscientoctopus/optimality-of-decentralization

5 Results and Discussion

We summarize the performance of the MADRL algorithms in all k-out-of-n settings using box plots and the test performance of agents during training in Figure 1. Additionally, we report the best performance observed for each algorithm in Table 2. For a more detailed comparison, we report mean performance with 95% confidence intervals over ten random seeds in Table 3 in the appendix. To enable a concise performance comparison of the paradigms, we aggregate the results across settings using mean, interquartile mean (IQM) and median in Figure 2 as

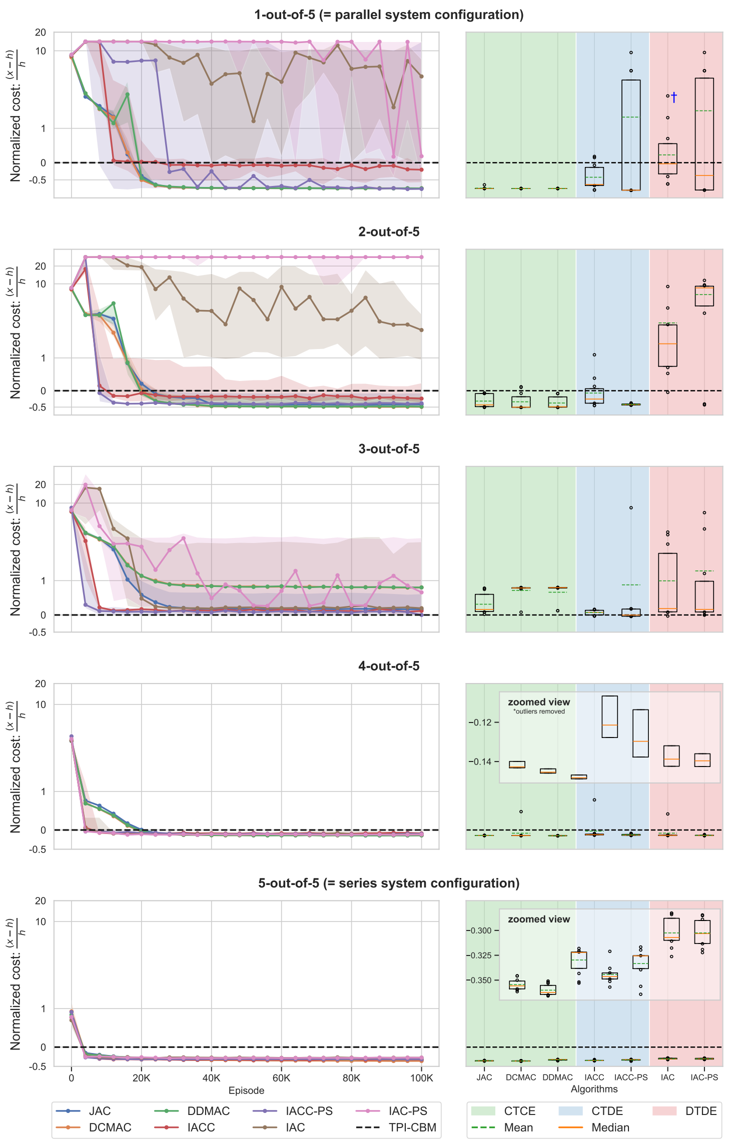

Fig. 1. Performance of CTCE, CTDE and DTDE algorithms in different k-out-of-n settings, normalized w.r.t. the heursitic (h) (lower is better). **Left**: Performance of the algorithms evaluated every 4,000 episodes during training, using 10,000 rollouts. The curve for each algorithm is aggregated over ten training instances. The bold line and the shaded region indicate the median and interquartile range respectively. **Right**: Each box plot summarizes the performance of the best policies across ten training instances, where the box represents the interquartile range and circles denote the outliers.

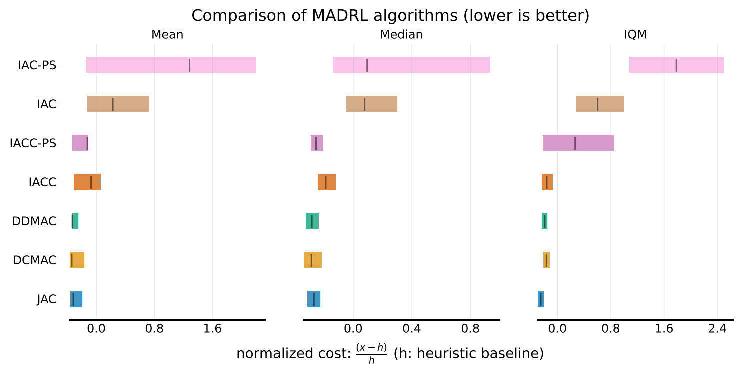

Fig. 2. Aggregate performance over the different k-out-of-n settings: mean, median and interquartile mean (IQM) (lower is better). Results for individual settings are normalized with respective heuristic baselines. Vertical lines indicate point estimates, and bands show 95% confidence intervals.

demonstrated by Agarwal et al. in [1]. Before aggregating the performance, we normalize the results with the respective baselines in each setting.

CTCE: Figure 1 shows that JAC, DCMAC, and DDMAC are all able to consistently outperform the heuristic in most settings, and the equivalence of their individual aggregate performance is shown in Figure 2, where the confidence intervals have a significant overlap. However, in the 3-out-of-5 setting, while their best instance remains competitive, we observe discrepancies in their performance, as reported in Table 2. Sub-optimalities in the CTCE paradigm have been previously reported by Bono et al. and have been attributed to the difficult exploration problem imposed by the joint state and action spaces [9]. We concur with this observation, especially because, unlike [9], we encode action-observation histories using beliefs and thus eliminating any additional complexities arising due to alternate approximations. The observed discrepancy cannot be merely an artifact of low neural network representational capacity as long as the same network architecture performs well in other settings. Another observation underpinning this claim is the best-performing training instance (random seed) of the IACC, IACC-PS and IAC algorithms, reported in Table 2, that are able to surpass the performance of CTCE algorithms and prove the existence of better policies.

CTDE: Both IACC and IACC-PS exhibit performance equivalent to CTCE algorithms in the 5-out-of-5 setting, but begin to exhibit variability as we move towards the 1-out-of-5 setting as shown in Figure 1. However, surprisingly, the best training instances of IACC and IACC-PS algorithms outperform JAC, the best CTCE training instance, in the 3-out-of-5 setting by about 7%. We examine this closely by comparing the histogram of returns from 10,00 rollouts for the IACC-PS and JAC policies in Figure 5 and contrast it with the corrective repair heuristic policy. We clearly observe how the environment under JAC policy is

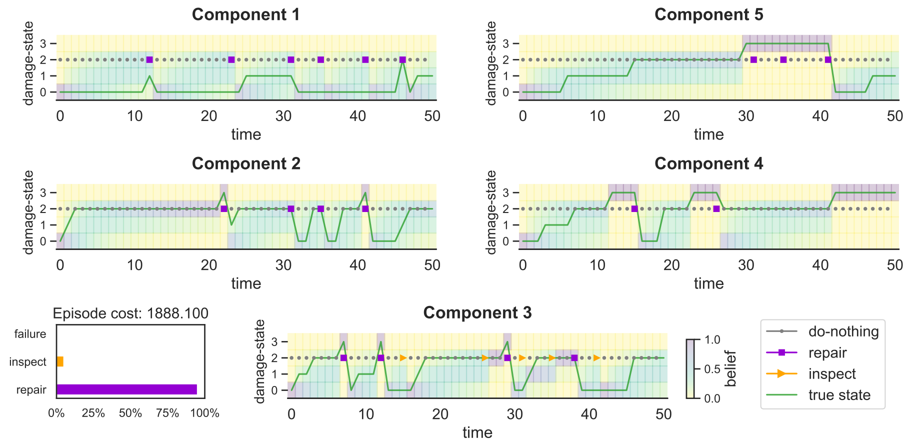

Fig. 3. Sample rollout of the best JAC policy in the 3-out-of-5 setting. The JAC agent chooses to monitor component 3 via inspections, and uses a combination of predictive and corrective repair actions to keep the system functional.

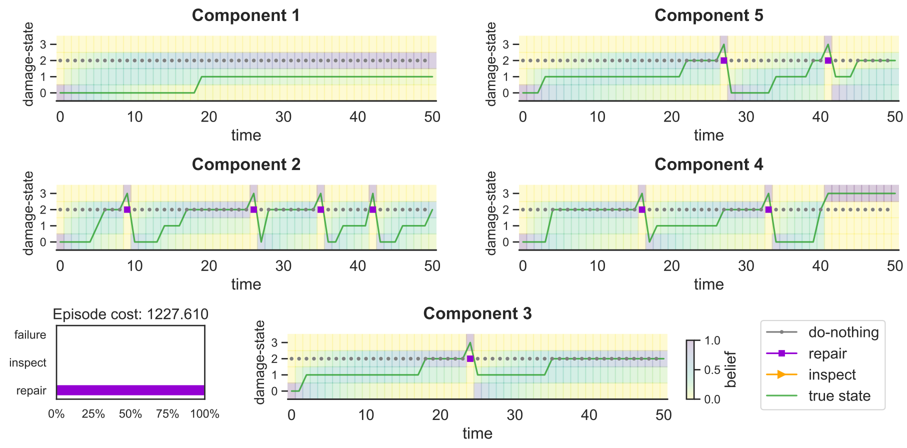

Fig. 4. Sample rollout of the best IACC-PS policy in the 3-out-of-5 setting. In most time steps, the agents choose simple corrective repair actions, however, they are able to marginally outperform the corrective replace heuristic policy by avoiding costly repair actions closer to the end of the life-cycle. We observe this for component 4 after $t = 40$, where the agent chooses do-nothing despite component failure.

slightly more likely to witness at least one system failure. Additionally, we also plot individual rollouts under these polices in Figures 3 and 4, respectively. In most time steps, the IACC-PS policy chooses corrective repair actions, however, it is able to marginally outperform the corrective repair heuristic by avoiding costly repair actions closer to the end of the life-cycle, such as for component 4 after $t = 40$. In contrast, the JAC agent chooses to monitor component 3 via inspections, and chooses a combination of predictive and corrective repair actions to keep the system functional. A possible explanation is that this component combines aggressive deterioration and low inspection to repair cost ratio.

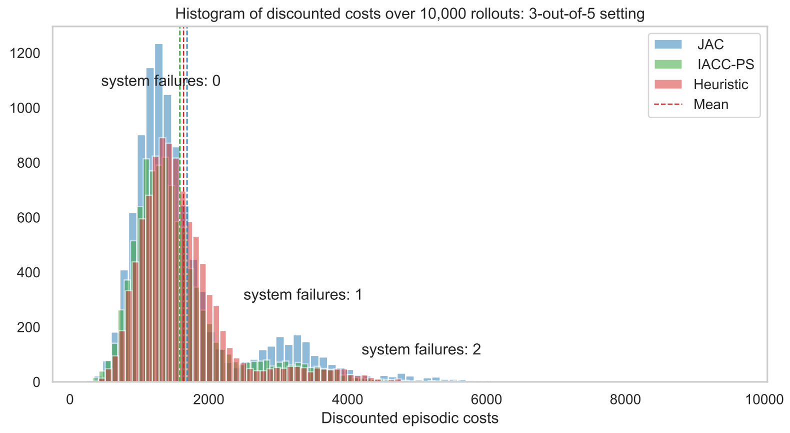

Fig. 5. Histograms of the discounted episodic costs of the best performing training instances (as highlighted in Table 2) in the 3-out-of-5 setting compared against a corrective repair heuristic policy over 10,000 environment rollouts. The modes in the histograms correspond to episodes in which system failures occur. Note that the IACC-PS policy closely resembles the corrective repair policy, but is marginally better as explained in Figure 4.

DTDE: In terms of aggregate performance, fully decentralized algorithms are outperformed by both CTCE and CTDE algorithms, as shown in Figure 2. Similar to CTDE algorithms, DTDE algorithms exhibit increasing variability in performance as we move from the 5-out-of-5 setting to the 1-out-of-5 setting, i.e. towards environments requiring more coordination among agents, as shown in Figure 1. However, the performance of DTDE algorithms is more adversely impacted as the need for cooperation in the environment increases. We illustrate this using a sample rollout of an outlier policy in the 1-out-of-5 setting (indicated with † in Figure 1) in Figure 6. Although the best performing instance is able to outperform the heuristic, this training instance succumbs to multi-agent learning pathologies and results in a poor policy. The performance of the parameter-sharing variants show similar variability, but can sometimes stumble into good policies, as reported in Table 2 for the 1-out-of-5, 4-out-of-5 and 5-out-of-5 settings.

Overall, as we move from the 5-out-of-5 setting to the 1-out-of-5 setting, environments demand more coordination among agents. In Figure 2, we observe

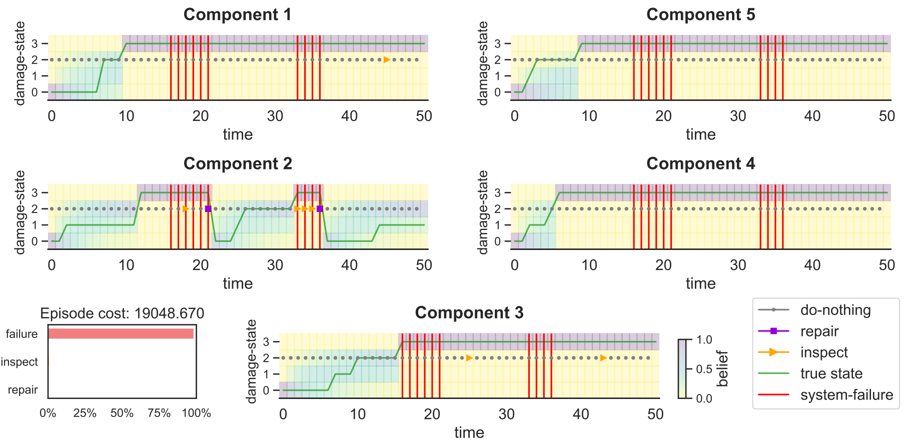

Fig. 6. Sample rollout of an outlier IAC policy in the 1-out-of-5 setting (indicated with † in Figure 1), where the agents succumb to multi-agent learning pathologies, resulting in a poor policy. Agents 1,3,4,5 choose do-nothing actions even during system failure, and only agent 2 takes necessary repair actions.

a clear trend: CTCE algorithms are unaffected by coordination requirements and thus show little variability in aggregate performance. In contrast, DTDE algorithms show large variability highlighting their susceptibility to multi-agent learning pathologies. CTDE algorithms are able to mitigate these and foster coordination among agents, thus performing better than their DTDE counterparts. Lastly, in Figure 2, we also observe that the performance of the parameter sharing variants is comparable to their independent counterparts, in addition to being significantly more computationally efficient to train.

6 Conclusions

Multi-agent deep reinforcement learning provides a scalable approach to optimal inspection and maintenance (I&M) planning for multi-component systems. In this work, we study the three major multi-agent learning paradigms, i.e., centralized training with centralized execution (CTCE), centralized training with decentralized execution (CTDE) and decentralized training with decentralized execution (DTDE), through the lens of a k-out-of-5 system to investigate their efficacy and limitations in I&M planning. Our key conclusions are listed below:

– While CTCE methods are shown to be superior overall, they also remain susceptible to sub-optimal policies due to exploration challenges induced by the joint state/action spaces. More importantly, the large joint spaces hinder scalability of these algorithms in real-world I&M problems.

- CTDE and DTDE approaches, although more scalable, face challenges stemming from decentralization, such as the emergence of non-stationary environments, multi-agent credit assignment issues, shadowed equilibria, etc. Vanilla CTDE approaches are observed to mitigate these pathologies and consistently outperform their decentralized counterparts. However, due to the large variability in performance, DTDE algorithms are sometimes still able to converge to competitive policies.
- The overall performance of parameter-sharing variants in both CTDE and DTDE paradigms is comparable to their independent counterparts, despite heterogeneity in system components in terms of costs, deterioration characteristics, and action effects. Additionally, parameter-sharing variants are significantly more computationally efficient to train.
- Finally, with increasing number of redundant components in the system, more cooperation among agents is needed to maximally exploit these redundancies. As such, environments with redundant system components can serve as a good benchmark to test agent cooperation in the context of I&M planning.

We aim to continue this line of work and study more rigorously the influence of component connectivity configurations in deteriorating engineering systems, such as series-parallel systems, and homogeneity/heterogeneity of system components on policy learning.

Acknowledgements This material is based upon work supported by the TU Delft AI Labs program. The authors gratefully acknowledge this support.

References

1. Agarwal, R., Schwarzer, M., Castro, P.S., Courville, A.C., Bellemare, M.: Deep Reinforcement Learning at the Edge of the Statistical Precipice. In: Advances in Neural Information Processing Systems. vol. 34, pp. 29304–29320. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html
2. Albrecht, S.V., Christianos, F., Schäfer, L.: Multi-Agent Reinforcement Learning: Foundations and Modern Approaches. MIT Press (2023)
3. Amato, C., Chowdhary, G., Geramifard, A., Ure, N.K., Kochenderfer, M.J.: Decentralized control of partially observable Markov decision processes. In: 52nd IEEE Conference on Decision and Control. pp. 2398–2405. IEEE, Firenze (Dec 2013), http://ieeexplore.ieee.org/document/6760239/
4. Andriotis, C.P., Papakonstantinou, K.G.: Managing engineering systems with large state and action spaces through deep reinforcement learning. Reliability Engineering and System Safety **191**(March), 106483 (2019), https://doi.org/10.1016/j.ress.2019.04.036
5. Andriotis, C.P., Papakonstantinou, K.G.: Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints. Reliability Engineering and System Safety **212**(March), 107551 (2021), https://doi.org/10.1016/j.ress.2021.107551

6. Andriotis, C.P., Papakonstantinou, K.G., Chatzi, E.N.: Value of structural health information in partially observable stochastic environments. Structural Safety **93**, 102072 (Nov 2021). https://doi.org/10.1016/J.STRUSAFE.2020.102072, publisher: Elsevier
7. Arcieri, G., Hoelzl, C., Schwery, O., Straub, D., Papakonstantinou, K.G., Chatzi, E.: Bridging POMDPs and Bayesian decision making for robust maintenance planning under model uncertainty: An application to railway systems. Reliability Engineering & System Safety **239**, 109496 (Nov 2023), https://www.sciencedirect.com/science/article/pii/S0951832023004106
8. Bismut, E., Straub, D.: Optimal adaptive inspection and maintenance planning for deteriorating structural systems. Reliability Engineering & System Safety **215**, 107891 (Nov 2021), https://www.sciencedirect.com/science/article/pii/S0951832021004063
9. Bono, G., Dibangoye, J.S., Matignon, L., Pereyron, F., Simonin, O.: Cooperative Multi-agent Policy Gradient. In: Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., Ifrim, G. (eds.) Machine Learning and Knowledge Discovery in Databases, vol. 11051, pp. 459–476. Springer International Publishing (2019), http://link.springer.com/10.1007/978-3-030-10925-7_28
10. Christianos, F., Papoudakis, G., Rahman, A., Albrecht, S.V.: Scaling Multi-Agent Reinforcement Learning with Selective Parameter Sharing (Jun 2021), http://arxiv.org/abs/2102.07475, arXiv:2102.07475 [cs]
11. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence. pp. 746–752. AAAI '98/IAAI '98, American Association for Artificial Intelligence, USA (Jul 1998)
12. Deodatis, G., Fujimoto, Y., Ito, S., Spencer, J., Itagaki, H.: Non-periodic inspection by Bayesian method I. Probabilistic Engineering Mechanics **7**(4), 191–204 (Jan 1992), https://www.sciencedirect.com/science/article/pii/026689209290023B
13. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual Multi-Agent Policy Gradients (Dec 2017), http://arxiv.org/abs/1705.08926, arXiv:1705.08926 [cs]
14. Fulda, N., Ventura, D.: Predicting and preventing coordination problems in cooperative Q-learning systems. In: Proceedings of the 20th international joint conference on Artifical intelligence. pp. 780–785. IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (Jan 2007)
15. Goldman, C.V., Zilberstein, S.: Decentralized Control of Cooperative Systems: Categorization and Complexity Analysis. Journal of Artificial Intelligence Research **22**, 143–174 (Nov 2004). https://doi.org/10.1613/jair.1427
16. Grall, A., Bérenguer, C., Dieulle, L.: A condition-based maintenance policy for stochastically deteriorating systems. Reliability Engineering and System Safety **76**(2), 167–180 (2002). https://doi.org/10.1016/S0951-8320(01)00148-X
17. Gronauer, S., Diepold, K.: Multi-agent deep reinforcement learning: a survey. Artificial Intelligence Review **55**(2), 895–943 (Feb 2022), https://doi.org/10.1007/s10462-021-09996-w
18. Gupta, J.K., Egorov, M., Kochenderfer, M.: Cooperative Multi-agent Control Using Deep Reinforcement Learning. In: Sukthankar, G., Rodriguez-Aguilar, J.A. (eds.) Autonomous Agents and Multiagent Systems, vol. 10642, pp. 66–83. Springer International Publishing, Cham (2017), http://link.springer.com/10.1007/978-3-319-71682-4_5

19. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence **101**(1-2), 99–134 (May 1998). https://doi.org/10.1016/S0004-3702(98)00023-X, publisher: Elsevier
20. Kapur, K.C., Pecht, Michael: Reliability Engineering. John Wiley & Sons, Inc, first edn. (2014)
21. Kochenderfer, M.J., Wheeler, T.A., Wray, K.H.: Algorithms for decision making. The MIT Press, Cambridge, Massachusetts (2022)
22. Kuo, W., Zuo, M.: Optimal Reliability Modeling: Principles and Applications. John Wiley & Sons. John Wiley & Sons, Inc., Hoboken, New Jersey (Jan 2003), https://catalogimages.wiley.com/images/db/pdf/047139761X.07.pdf
23. Leroy, P., Morato, P.G., Pisane, J., Kolios, A., Ernst, D.: IMP-MARL: a Suite of Environments for Large-scale Infrastructure Management Planning via MARL (Jun 2023), http://arxiv.org/abs/2306.11551, arXiv:2306.11551 [cs, eess] version: 1
24. Luque, J., Straub, D.: Risk-based optimal inspection strategies for structural systems using dynamic Bayesian networks. Structural Safety **76**(June 2017), 68–80 (2019), https://doi.org/10.1016/j.strusafe.2018.08.002, publisher: Elsevier
25. Lyu, X., Baisero, A., Xiao, Y., Daley, B., Amato, C.: On Centralized Critics in Multi-Agent Reinforcement Learning. Journal of Artificial Intelligence Research **77**, 295–354 (May 2023), https://www.jair.org/index.php/jair/article/view/14386
26. Madanat, S., Ben-Akiva, M.: Optimal Inspection and Repair Policies for Infrastructure Facilities. Transportation Science **28**(1), 55–62 (Feb 1994). https://doi.org/10.1287/trsc.28.1.55, https://pubsonline.informs.org/doi/10.1287/trsc.28.1.55, publisher: INFORMS
27. Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. The Knowledge Engineering Review **27**(1), 1–31 (Feb 2012), https://www.cambridge.org/core/product/identifier/S0269888912000057/type/journal_article
28. Morato, P.G., Andriotis, C.P., Papakonstantinou, K.G., Rigo, P.: Inference and dynamic decision-making for deteriorating systems with probabilistic dependencies through Bayesian networks and deep reinforcement learning. Reliability Engineering & System Safety **235**, 109144 (Jul 2023), https://www.sciencedirect.com/science/article/pii/S0951832023000595
29. Morato, P.G., Papakonstantinou, K.G., Andriotis, C.P., Nielsen, J.S., Rigo, P.: Optimal inspection and maintenance planning for deteriorating structural components through dynamic Bayesian networks and Markov decision processes. Structural Safety **94**(August 2021), 102140 (2022), https://doi.org/10.1016/j.strusafe.2021.102140
30. Oliehoek, F.A., Amato, C.: A Concise Introduction to Decentralized POMDPs. SpringerBriefs in Intelligent Systems, Springer International Publishing, Cham (2016)
31. Oliehoek, F.A., Spaan, M.T.J., Vlassis, N.: Optimal and Approximate Q-value Functions for Decentralized POMDPs. Journal of Artificial Intelligence Research **32**, 289–353 (May 2008). https://doi.org/10.1613/jair.2447
32. Papakonstantinou, K.G., Shinozuka, M.: Planning structural inspection and maintenance policies via dynamic programming and Markov processes. Part II: POMDP implementation. Reliability Engineering and System Safety **130**, 214–224 (2014), http://dx.doi.org/10.1016/j.ress.2014.04.006, publisher: Elsevier
33. Papakonstantinou, K.G., Andriotis, C.P., Shinozuka, M.: POMDP and MOMDP solutions for structural life-cycle cost minimization under partial and mixed ob-

servability. Structure and Infrastructure Engineering **14**(7), 869–882 (Jul 2018), https://doi.org/10.1080/15732479.2018.1439973
34. Papoudakis, G., Christianos, F., Schäfer, L., Albrecht, S.V.: Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks (Nov 2021), http://arxiv.org/abs/2006.07869, arXiv:2006.07869 [cs, stat]
35. Peng, B., Rashid, T., de Witt, C.A.S., Kamienny, P.A., Torr, P.H.S., Böhmer, W., Whiteson, S.: FACMAC: Factored Multi-Agent Centralised Policy Gradients (May 2021), http://arxiv.org/abs/2003.06709, arXiv:2003.06709 [cs, stat]
36. Rashid, T., Samvelyan, M., de Witt, C.S., Farquhar, G., Foerster, J., Whiteson, S.: QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning (Jun 2018), http://arxiv.org/abs/1803.11485, arXiv:1803.11485 [cs, stat]
37. Shani, G., Pineau, J., Kaplow, R.: A survey of point-based POMDP solvers. Auton Agent Multi-Agent Syst **27**, 1–51 (2013). https://doi.org/10.1007/s10458-012-9200-2
38. Spaan, M.T.J., Vlassis, N.: Perseus: Randomized Point-based Value Iteration for POMDPs. Journal of Artificial Intelligence Research **24**, 195–220 (2005)
39. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W.M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J.Z., Tuyls, K., Graepel, T.: Value-Decomposition Networks For Cooperative Multi-Agent Learning (Jun 2017), http://arxiv.org/abs/1706.05296
40. Terry, J.K., Grammel, N., Son, S., Black, B.: Parameter Sharing For Heterogeneous Agents in Multi-Agent Reinforcement Learning (Jan 2022), http://arxiv.org/abs/2005.13625
41. Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., de Freitas, N.: Sample Efficient Actor-Critic with Experience Replay (Jul 2017), http://arxiv.org/abs/1611.01224
42. Wong, A., Bäck, T., Kononova, A.V., Plaat, A.: Deep Multiagent Reinforcement Learning: Challenges and Directions (Oct 2022), http://arxiv.org/abs/2106.15691, arXiv:2106.15691 [cs]
43. Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., Wu, Y.: The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games (Nov 2022), http://arxiv.org/abs/2103.01955, arXiv:2103.01955 [cs]
44. Zhu, B., Frangopol, D.M.: Risk-Based Approach for Optimum Maintenance of Bridges under Traffic and Earthquake Loads. Journal of Structural Engineering **139**(3), 422–434 (Mar 2013). https://doi.org/10.1061/(ASCE)ST.1943-541X.0000671, publisher: American Society of Civil Engineers
45. Åström, K.J.: Optimal control of Markov processes with incomplete state information. Journal of Mathematical Analysis and Applications **10**(1), 174–205 (1965). https://doi.org/10.1016/0022-247X(65)90154-X

7 Appendix

As described in the main text, we use the 4-out-of-5 setting for hyperparameter optimization and report the tuned hyperparameters in Table 4. We note that decentralized agents often exhibit instabilities when using large replay buffers, thus, their replay buffers are much smaller than their centralized counterparts. This is because random samples from a large replay buffer can correspond to

Table 3. Mean performance of the algorithms aggregated over ten training instances (random seeds) with ± indicating the 95% confidence interval. Bold indicates best in the respective k-out-of-n setting.

		1-out-of-5	**2-out-of-5**	**3-out-of-5**	**4-out-of-5**	**5-out-of-5**
Baseline	TPI-CBM	1485.74	1498.8	**1654.19**	3998.41	14780.43
CTCE	JAC	**372.46 (±2.51)**	**1070.24 (±178.81)**	**2487.57 (±285.79)**	**3488.80 (±11.30)**	**12698.22 (±50.20)**
	DCMAC	**377.01 (±5.83)**	**1146.16 (±162.51)**	**2774.12 (±256.45)**	**3451.70 (±9.68)**	**12642.19 (±56.38)**
	DDMAC	**379.71 (±21.31)**	**1194.30 (±188.91)**	2965.18 (±152.77)	**3517.82 (±12.38)**	**13097.86 (±160.00)**
CTDE	IACC	13467.32 (±940.42)	13844.57 (±988.96)	14329.28 (±850.25)	9952.01 (±723.77)	20830.69 (±277.56)
	IACC-PS	13311.32 (±1326.17)	13320.42 (±1316.56)	11802.21 (±2468.70)	7449.54 (±1025.22)	17457.71 (±2312.53)
DTDE	IAC	4797.06 (±3042.81)	4892.51 (±2266.34)	**3917.98 (±1273.71)**	4319.42 (±49.72)	13528.65 (±278.93)
	IAC-PS	5442.69 (±3246.56)	7326.18 (±2951.19)	**4051.89 (±2519.96)**	**3632.35 (±86.58)**	13505.44 (±131.70)

Table 4. Tuned hyperparameters of the algorithms used to train agents on various k-out-of-n settings

Paradigm		**CTCE**			**CTDE**		**DTDE**	
Algorithm		**JAC**	**DCMAC**	**DDMAC**	**IACC**	**IACC-PS**	**IAC**	**IAC-PS**
Episodes (E)		100,000						
Timesteps		$E \times t_H = 100,000 \times 50 = 5M$						
Architecture	Actor	[21, 64, 64, 243]	[21, 32, 32, 15]	[21, 16, 16, 3] x 5	[5, 16, 16, 3] x 5	[10, 32, 32, 3]	[5, 16, 16, 3] x 5	[10, 32, 32, 3]
	# parameters	21,363	2,255	3,375	2,095	1,507	2,095	1,507
	Critic	[21, 64, 64, 1]	[21, 64, 64, 1]	[21, 64, 64, 1]	[21, 64, 64, 1]	[21, 64, 64, 1]	[5, 16, 16, 1] x 5	[10, 64, 64, 1]
	# parameters	5,633	5,633	5,633	5,633	5,633	1,925	4,929
	Total # parameters	26,996	7,888	9,008	7,728	7,140	4,020	6,436
learning	batch size	64 (timesteps)						
Actor	optimizer	Adam						
	initial lr	0.0001	0.0001	0.0001	0.0005	0.0005	0.0001	0.0005
	decay factor	0.1						
	decay episodes	20,000	20,000	20,000	10,000	10,000	25,000	25,000
	decay type	linear						
Critic	initial_lr	0.005	0.005	0.001	0.005	0.001	0.001	0.001
	decay factor	0.1						
	decay episodes	20,000	20,000	20,000	10,000	25,000	25,000	25,000
	decay type	linear						
ϵ-greedy strategy	ϵ-start	1						
	ϵ-end	0.005	0.005	0.005	0.001	0.005	0.001	0.001
	decay episodes	20,000	20,000	20,000	10,000	10,000	25,000	25,000
	decay type	linear						
Replay Buffer	timesteps	500,000	500,000	500,000	10,000	10,000	10,000	10,000

policies significantly different from the current policy, forcing agents to modify their recently learned policy drastically.

For training, we employ a variant of the actor-critic with experience replay (ACER) algorithm for learning [41] as introduced for I&M in [4,5] and outlined above. To minimize the variance caused by the importance sampling weights, we clip the values w_i by setting $\bar{w} = 2$.

Algorithm 1: Joint Actor Critic (JAC)

```
   Initialize: Actor $\boldsymbol{\pi}(\mathbf{a}|\mathbf{b};\theta)$, Critic $V^{\boldsymbol{\pi}}(\mathbf{b};\phi)$, Experience Replay buffer $\mathcal{D}$,
               exploration scheduler $\varepsilon$, learning rates $\alpha_a^0, \alpha_c^0$, learning rate
               schedulers $\zeta_a, \zeta_c$, optimizers $\eta_a, \eta_c$, batch size $\mathtt{B}$, Episodes $E$, global
               time step $t' = 0$, exploration probability $\epsilon_0$.
1  for $e = 1$ to $E$ do
2      $\mathbf{b}_0 = \mathrm{env}()$                                          // get initial belief
3      for $t = 0$ to $t_H - 1$ do
4          if explorationstrategy($\epsilon_t'$) $=$ random then
5              $\mathbf{a}_t \sim \mathcal{U}(\mathbf{A})$, $\mathbf{A}$ is the joint action space
6          else
7              Sample action: $(\mathbf{a}_t, \mathbf{p}_t) = \boldsymbol{\pi}(\mathbf{b}_t;\theta)$
8          Environment step: $(\mathbf{b}_{t+1}, c_t, \mathtt{d}_t) = \mathrm{env}(\mathbf{a}_t)$          // $\mathtt{d}_t := \mathbb{1}_{t_H-1}(t)$
9          Store experience: $(\mathbf{b}_t, \mathbf{a}_t, p_t, \mathbf{b}_{t+1}, r_t, \mathtt{d}_t) \cup \mathcal{D}$
10         if $t' > 10 \times B$ then
11             Sample batch from buffer: $\mathcal{B} = \{(\mathbf{b}_t, \mathbf{a}_t, p_t, \mathbf{b}_{t+1}, c_t, \mathtt{d}_t)\}_{i=1}^{\mathtt{B}} \sim \mathcal{D}$
12             $\boldsymbol{\pi}(\mathbf{a}|\mathbf{b};\theta^{\mathrm{t+1}})$, $V^{\boldsymbol{\pi}}(\mathbf{b};\phi^{\mathrm{t+1}}) =$ Train()
13         $\mathbf{b}_t \longleftarrow \mathbf{b}_{t+1}$
14     $\epsilon_{t'+1} \longleftarrow \varepsilon(\epsilon_t')$                           // update exploration parameter
15     $\alpha_a^{t'+1} \longleftarrow \zeta_a(\alpha_a^{t'})$                             // update actor learning rate
16     $\alpha_c^{t'+1} \longleftarrow \zeta_c(\alpha_c^{t'})$                             // update critic learning rate
17     $t' \longleftarrow t' + 1$
18 Train():
19     For each sample $i$ in batch,
20     future $= \mathtt{sg}(V^{\boldsymbol{\pi}}(\mathbf{b}_i';\phi)) \cdot \mathtt{d}_i$                  // stop gradients: sg($\cdot$)
21     Advantage: $\mathtt{A}_i = c_i + \gamma \cdot$ future $- V^{\boldsymbol{\pi}}(\mathbf{b}_i;\phi)$                 // TD error
22     $\rho_i = \log \boldsymbol{\pi}(\mathbf{a}_i|\mathbf{b}_i;\theta)$
23     Importance sampling weights: $w_i = \frac{\mathtt{sg}(\boldsymbol{\pi}(\mathbf{a}_i|\mathbf{b}_i;\theta))}{p_i}$
24     Clipping weights: $w_i \longleftarrow \min(w_i, \bar{w})$
25     Actor gradients: $\mathrm{g}_\theta = \frac{1}{\mathtt{B}} \sum_{i=1}^{\mathtt{B}} w_i \cdot (\nabla_\theta \rho_i \cdot \mathtt{A}_i)$
26     Critic gradients: $\mathrm{g}_\phi = \frac{1}{\mathtt{B}} \sum_{i=1}^{\mathtt{B}} w_i \cdot \nabla_\phi \mathtt{A}_i$
27     $\theta' \longleftarrow \eta_a(\mathrm{g}_\theta, \theta, \alpha_a)$
28     $\phi' \longleftarrow \eta_c(\mathrm{g}_\phi, \phi, \alpha_c)$
29     return $\boldsymbol{\pi}(\mathbf{a}|\mathbf{b};\theta')$, $V^{\boldsymbol{\pi}}(\mathbf{b};\phi')$
```