Network Maintenance Planning Via Multi-Agent Reinforcement Learning*

Jonathan Thomas[1], Marco Pérez Hernández[2], Ajith Kumar Parlikad[2] and Robert Piechocki[1]

***Abstract*— Within this work, the challenge of developing maintenance planning solutions for networked assets is considered. This is challenging due to the very nature of these systems which are often heterogeneous, distributed and have complex co-dependencies between the constituent components for effective operation. We develop a Multi-Agent Reinforcement Learning (MARL) solution for this domain and apply it to a simulated Radio Access Network (RAN) comprising of nine Base Stations (BS). Through empirical evaluation we show that our model outperforms fixed corrective and preventive maintenance policies in terms of network availability whilst generally utilizing less than or equal amounts of maintenance resource.**

I. Introduction

The maintenance planning of networked assets is a complex problem. Not only because of the variety and quantity of components each asset might have, but also because of the inter-dependencies between the assets within the network. The components of the assets have their own expected service life and degradation profiles that in turn influence asset's service life and the overall network operation. Likewise, the maintenance or repair jobs, at component level, have different effects at the asset and the wider network levels.

These factors become relevant when designing a network-wide maintenance policy that considers not only the maintenance operations costs (e.g. labour and parts) but also the costs of unplanned downtime within the network.

This is a complex problem with growing interest among the research community. As reported in Section III, some of the approaches proposed to tackle this challenge, focus more on the component or system-level complexities. Moreover, modelling frameworks such as Markov Decision Processes (MDP) have proven effective to capture some of the problem dynamics and relationships, however, when larger numbers of elements are considered the problem becomes even harder to address. The resulting state and action spaces of the MDP grow exponentially and the associated transition probabilities are difficult to determine, for even modest numbers of elements numerical methods become infeasible.

Alternatively, Reinforcement Learning (RL) approaches have been widely used in different contexts where problems can be defined as MDPs. RL provides an effective framework to determine the sequence of actions that maximises a given reward function. In addition, when the system is composed of multiple entities, each one with their own state spaces and the ability to perform individual actions that influence the overall system, then the problem can be further broken down into multiple RL agents. The Multi-Agent RL (MARL) framework becomes an option to solve the most suitable network-wide maintenance policy by learning from the distributed actions taken by the agents of the system.

This paper addresses the problem of finding a cost-effective maintenance policy for a network of multi-component assets using MARL. Particularly, the contributions of the paper are threefold: 1) Definition of the networked asset maintenance problem as a Multi-Agent problem 2) An approach to solve this problem based on the Multi-Actor Critic (MAAC) framework and 3) Demonstration of the proposed approach in a simulated environment within the context of Radio Access Networks (RAN).

The remaining of this paper is organised as follows. First, Sections II and III, present the relevant works in MARL and group asset maintenance planning, respectively. Next, Section IV introduces the formulation of the group maintenance problem as a stochastic game. The Section V describes the architectural aspects of the proposed approach. The results and discussion of the evaluation, based on a simulated case study, are presented in Section VI. Conclusions and future work are presented in Section VII.

II. Single and Multi Agent Reinforcement Learning

Deep RL has seen considerable success in a variety of domains where most of the high profile successes have been within game-play domains [1], [2]. Most of the major successes of RL have been within the single-agent or two-player games. Some problems, don't naturally fit into this paradigm and are more appropriately modelled as larger MARL problems where agents behave cooperatively, competitively or both in a shared environment [3]. Due to the increase in the number of agents, the complexity of the interactions with the environment are exacerbated and in recent years considerable research has been undertaken with the intention of addressing the issues this introduces. Some of the major challenges include environment non-stationarity [4] and credit reward assignment [5]. Some examples of work undertaken include Multi-Agent Deep Deterministic Policy Gradients (MADDPG) [3], where a centralised critic is utilised to reduce non-stationarity, Counterfactual Multi-Agent Policy Gradients (COMA) for the multi-agent credit assignment problem and DIAL [6] introduces a method to

*This work was supported by the Next Generation Convergent Digital Infrastructure Project funded by Engineering and Physical Sciences Research Council (EPSRC) and British Telecom (BT) under grant EP/R004935/1.

[1]Communications, Systems and Network Group, University of Bristol, [jt17591,r.j.piechocki]@bristol.ac.uk

[2]Institute for Manufacturing, University of Cambridge, [mep53, aknp2]@cam.ac.uk

facilitate communication. A comprehensive overview of RL and MARL can be found in [5] and [7] respectively.

III. Asset Maintenance Planning

This sections reviews relevant efforts addressing the multi-asset maintenance planning and also applications of RL for asset management.

A. Maintenance of Network of Assets

Markov decision processes have been widely used to model asset deterioration and to develop maintenance plans. The problem of finding the optimum maintenance policy for a component is formulated as an MDP in [8]. The policy for determining when the component should be maintained, is obtained via unichain policy iteration algorithms. Authors of [9] use partially-observable MDPs as basis of a methodology for assessing the benefits of condition monitoring systems.

While these works address mainly single asset scenarios, MDPs have also supported models of multi-asset systems. A model to optimize the predictive maintenance policy for a multi-system multi-component network of assets is presented in [10]. The policy is obtained using a genetic algorithm that enable exploitation of the benefits of grouping system interventions and overlapping downtime in a two-bridge network. A MDP is also used to formulate a joint Condition-Based Component Replacement and Inventory Control Policy (CBRICP) in [11]. Dynamic programming is used to solve the CBRICP, determining the optimal maintenance and inventory actions for *k-out-n:F* systems.

B. Reinforcement Learning for AM

Several works have addressed the use of single agent RL for asset maintenance planning. A RL model for determining the optimal timing for maintenance while maintaining reliability of the system is presented in [12]. The model was shown to effectively optimise maintenance scheduling considering expected reward and mean time between failures (MTBF). In [13], a Monte Carlo RL (MCRL) approach is used to find the optimal preventive maintenance strategy for multiple components of a group of assets, with application to a fleet of military trucks. The authors showed that MCRL enabled learning of the transition probabilities of the underlying MDP while optimising at asset system-level rather than locally. A method combining Q-learning algorithm and an ensemble of Artificial Neural Networks (ANN) enables learning of the optimal operation and maintenance policy for a power grid [14]. Despite the reported high computational time required, a distinct feature of this method is the ability to potentially consider large systems with high dimensional state-action spaces. The benefits of neural networks with RL are also exploited in [15], where the Double Deep-Q Network (DDQN) framework is applied to determine the preventive maintenance policies that reduce the cost in a serial production line system.

The MARL framework has been less exploited for addressing asset maintenance planning problems. A small system of two machines and one buffer is considered in [16], where authors propose a cost-sharing-RL algorithm that outperforms single-agent RL when comparing the system average costs. Likewise, authors of [17] show the benefits of a multi-agent configuration where the RL framework enables reduction of downtime and maintenance costs comparing to other available strategies in a parallel production line system. Deep Centralised Multi-Agent Actor Critic (DCMAC) is proposed for obtaining the maintenance policy of multi-component systems, with application to a truss bridge [18]. Authors of [19] combine Genetic Algorithms (GA) and MARL to maximise average revenue rate of a manufacturing system with intermediate buffers. This revenue considers the cost of corrective and preventive maintenance of the entire system as part of the agent's reward function. The GA assumes global visibility of the system and periodically signals agents to adjust their actions. While there has been efforts to address cases of larger multi-component systems [18] and small scale multi-asset systems [16], [17] separately, the use of RL frameworks to address the complexity of maintenance policy computation in networks of multi-assets multi-component systems is still to be explored.

IV. Problem Formulation

The system of interest is a network of m assets (A) each one with n components (c) that follow cycles of deterioration and maintenance. The key elements of the problem are further described below.

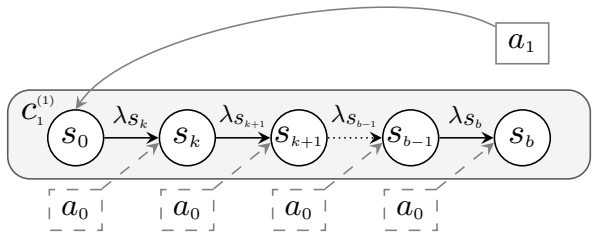
A. Component Level

Each component is modelled as a MDP as illustrated in Fig. 1. The continuous state space $S_c = \{s_0, s_1, ..., s_k, s_{k+1}, ..., s_{b-1}, s_b\}$ represents the states of the component until breakdown (s_b). Where, each state is collated from observation of the component's sensors which are inherently noisy. The deterioration rate λ_k determines the change in condition from state s_k to state s_{k+1}, where the probability of the component transitioning directly to s_b increases as the component matures.

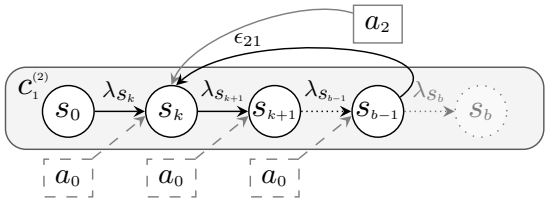
At every state, three distinct actions are considered: *Do nothing* (a_0), *Component Replacement* (a_1) and *Component Repair* (a_2). a_1 results in the state of the component being returned to s_0. a_2 generally restores state of the component, however as the component is restored multiple times, the effect is reduced. This is shown in Fig. 1a.

B. System Level

Given each component's MDP, the natural way to model the decision problem for the entire system (network) is a stochastic game [20]. This can be defined by the tuple, $\Gamma = (\mathcal{S}, \mathcal{U}, r, p, Z, O, n, \gamma)$. Where $\mathcal{S}$, denotes the state space from which an agent observes Z_N according to an observation function O. For the entire system the state S depends on the states of the assets A, whose state is the aggregation of their components states. $\mathcal{U}$, denotes the joint action to which each agent contributes an action u^n. Similarly to the states, the joint action for the system can



(a) Replacement action a_1 taken at s_b



(b) Component Repair action a_2 with effectiveness ϵ

Fig. 1: MDP showing two actions taken for components $c_1^{(1)}$ and $c_1^{(2)}$ of assets 1 and 2 respectively. State transitions are paired with the actions a that lead to the next state. λ represents the deterioration rate. s_b represents a component breakdown. *"Do nothing"* actions are dashed.

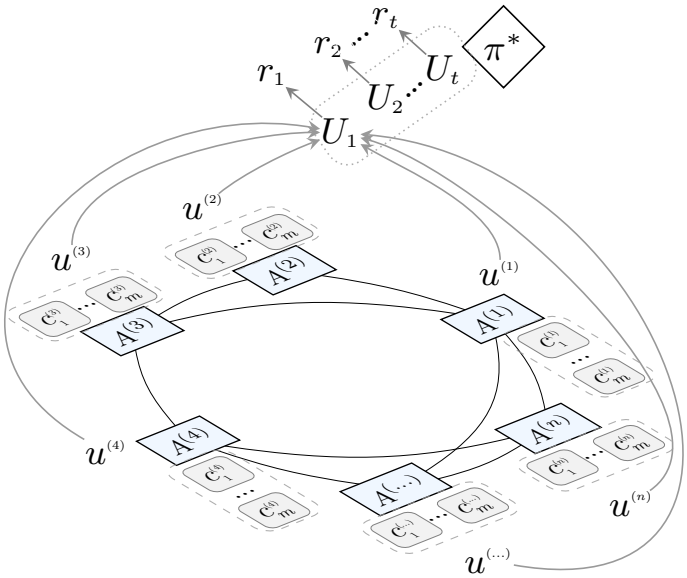


Fig. 2: Maintenance actions u are taken in a network of n assets A, each one with m components c. For a given time t, a joint action $u^{(k)}$ is taken for each asset, contributing to a network action U_t. The goal is to find the policy π that maximises the cumulative joint reward r.

be regarded as the set of actions taken for each asset at a given time. An asset possible actions are defined by the cartesian product of the set of for their components e.g. "Do nothing" for a_0 and "Component repair" for a_2. The reward function, r, is defined as $r : \mathcal{S} \times \mathcal{U} \rightarrow \mathrm{R}$. The transition probability, p, defines the transitions between states and can be represented as $p : \mathcal{S} \times \mathcal{U} \rightarrow \Omega(\mathcal{S})$, where the output is a probability over the states. Finally, to enable convergence of the cumulative rewards, a discount factor is introduced such as $\gamma \in [0, 1)$. Multiple definitions of r are possible capturing different aspects of interest of the network of assets, such as the topology or the dynamic workloads for each asset. As illustrated in Fig. 2, the goal is to find asset level maintenance policy (π^*) to determine the actions to take such that the expected cumulative discounted reward (r) is maximised.

V. MARL FOR ASSET MAINTENANCE

This section presents the approach for solving the problem defined in Section IV based on the MARL framework. The solution approach comprises the definition of the joint reward function r and the strategies that agents use to find policies that maximise the cumulative discounted reward which they experience.

A. Reward Definition

$$r = -\alpha \sum_{i \in n} R_i - \beta \sum_{i \in n} T_i \tag{1}$$

The intention is to find a network maintenance strategy that maximises network profitability. The reward function is formulated as shown in Equation 1, where this was derived from conversations with domain experts. R_i represents the cost of maintenance operation scheduled by asset i and T_i is a binary variable referring to the assets operational status (1 if the asset is unavailable and 0 otherwise). This formulation captures the trade-off between maintenance and network availability which is used as a proxy for profitability. The importance of these diametrically opposing objectives can be controlled through appropriate selection of α and β.

B. Learning Strategies

The target task could be approached in a centralized manner, where a single agent is responsible for learning and executing a maintenance policy for all assets. As discussed by [18], single agent RL can struggle with large discrete action spaces. In this case, the action space is likely to be very large due to significant numbers of assets and the action space scaling exponentially. A more practical approach and one that is considered in this work is a decentralized solution where techniques from MARL are applied with the intention of improving training efficiency and learning cooperative policies. Agents are assigned to assets and are responsible for their associated maintenance decisions. By decentralizing, there are also other benefits including a reduction in communication overheads, system robustness (as a centralized controller presents a single point of failure), and the framework supports the introduction of new assets [21].

The stochastic game (Section IV-B) defines the agents observations as being comprised of raw sensor values from which agents must learn a policy $\pi : Z \rightarrow u$ with the intention of maximizing the cumulative reward which the collective of agents experience. Due to their versatility, agents policies are parametrized by expressive Deep Neural

Networks (DNN), where LSTMs are utilized due to their capacity to extract temporal correlations.

Deep MARL approaches can take considerable time to train and hence training within simulation is considered to be the most viable path to deployment. A recent example of this strategy is the work undertaken by [22] where they train a policy within a simulator to control a stratospheric balloon for telecommunications applications and then successfully deploy it in the real-world. The work introduced in this paper utilises a custom simulator, but more generally a Digital Twin [23] is a viable option for algorithm training where asset models could be refined through integration of real-world data. The use of a simulated environment also provides opportunity to investigate system response in rare circumstances, which is beneficial from a system resilience perspective.

The utilisation of a simulation to train MARL also brings algorithmic benefits like Centralised Training Decentralised Execution (CDTE) [6]. This common MARL paradigm facilitates the introduction of extra information within training to improve training efficiency as long as this information is not required when the model is deployed [3]. A number of approaches exist including Multi-Agent Deep Deterministic Policy Gradients (MADDPG) [3] and Q-MIX [24], for example.

MADDPG considers a modification to Actor-Critic (AC) [5] for the MARL domain. Within single-agent RL AC (or Independent AC in [3]) the Actor is responsible for selection of actions and is conditioned on the observation of an individual agent and the critic provides feedback to the agent about the return associated with its selected action and is conditioned on the action and state of the agent. When naively applied to a MARL problem, this can encounter issues with non-stationarity as any estimate of the value which the critic produces is implicitly dependent on the state and policies of other agents. In MADDPG, the critic is centralized and is conditioned on the observations of all agents. By doing so, the problems aforementioned can be reduced where Lowe et al suggest that it improves the consistency of gradient signals [3]. Notably, when it comes to deployment the centralized critic is not required as it is not necessary for inference.

C. MAAC

The MADDPG approach is adapted for the preventive maintenance task and implemented into Advantage Actor-Critic (A2C) due to DDPG not supporting continuous action spaces. A2C was selected mostly because of its ease of implementation. Algorithm 1 presents the Multi-Agent Actor Critic (MAAC) base workflow used in this paper. The algorithm calculates the Advantage A, Policy gradient $\nabla_\theta J(\theta)$ and Value function gradient $\nabla_\phi J(\phi)$ with the functions detailed in Equations 3, 4 and 5 respectively.

D. convMAAC

$$\mathbf{h}_n^{(k)} = \sigma\left(\mathbf{W}^k \sum_{v \in \mathcal{N}(n) \cup \{n\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(n)||\mathcal{N}(v)|}}\right) \quad (2)$$

In situations where there is a high number of agents, as there will be in any realistic network maintenance operation the utilisation of a centralized critic may invoke significant computational expense due to it being parameterised by a fully connected network. To aid in model scalability, the problem is cast as a Graph Regression task. Thus Graph Convolutional Networks (GCN) [25] are used to learn useful node embeddings which capture topological information and then a mean operation on the set of node embeddings is used to reduce the graph to a fixed length vector. Where the associated graph $G = \{V, E\}$ is defined by the connections between elements in the network. Equation 2 defines the operation of the GCN, where $\mathbf{h}_u^k$ refers to the encoding of agent n after k convolutions and is initialised to Z_u at $k = 0$, $\mathbf{W}^k$ is a trainable weight matrix, and $\mathcal{N}\{n\}$ represents the set of neighbours of agent n. This results in a model with significantly less parameters than the fully connected equivalent, MAAC. This modification is referred to as convolutional MAAC (convMAAC) hence forth.

$$A(Z, U) = r + \gamma V(Z') - V(Z) \quad (3)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(Z, u) A(Z, U)] \quad (4)$$

$$\nabla_\phi J(\phi) = \mathbb{E}_{V_\phi}[\nabla_\theta r + \gamma V(Z') - V(Z)] \quad (5)$$

Algorithm 1: Multi-Agent Actor-Critic algorithm

Input: Γ
Output: π_θ, V_ϕ
initialise π_θ and V_ϕ ;
for *episodes* **do**
  t = 0;
  Initialise episode buffer $\mathcal{D}$;
  while *not terminal* **do**
    Get states Z_t from env;
    Sample $u_t \sim \pi(\cdot|Z_{i,t})\ \forall i \in N$;
    Receive r, and new observations Z_{t+1};
    Store $(Z_t, Z_{t+1}, \pi(u_i|Z_{i,t}), r)$ in D;
    t = t + 1;
  end
  For all samples in D;
  Compute Advantage using *Equation 3*;
  Compute Policy Gradient using *Equation 4*;
  Compute Value Gradient using *Equation 5*;
  $\theta \leftarrow \theta - \alpha_\theta \nabla_\theta J_\pi(\theta)$;
  $\phi \leftarrow \theta - \alpha_\phi \nabla_\phi J\phi(\phi)$;
end

TABLE I: Experiment parameters, where the numbers associated with actions u_x are their costs and the components RUL are initialised randomly according to a uniform distribution. Note that the the actions u_x which apply to both components are proportionally cheaper, this assumption is based on an operations engineer travel time being reduced relative to the equivalent actions concurrently.

Parameter	Value
Number of assets	9
Max maintenance operations per timestep	2
Episodes	3000
Episode length	100
u^0, Do nothing	0
u^1, Replace c_1	0.3
u^2, Repair c_2	0.1
u^3, Replace c_2	0.3
u^4, Replace c_1 and c_2	0.55
u^5, Repair c_1	0.1
u^6, Repair c_1 and c_2	0.175
Component RUL	$\mathcal{U}(95, 105)$
α	1
β	1

VI. Simulated Case Study: Radio Access Networks

The proposed approach is applied to the maintenance of the telecommunications infrastructure deployed within a region of interest. Particularly, the assets of interest are RAN base stations (BS) and the components are the network equipment units used in each BS. This definition is specific to this use case, however the granularity to which assets and components are defined might change from case to case.

As network elements can be connected according to different topologies, this case study considers two scenarios: complete network and star topologies. The first scenario involves a number N of BS directly connected with each other. This version of the problem results in the unavailability of a BS only impacting the users within its coverage range. In the second scenario, a central router interconnects the $N-1$ BS. Hence, the failure of the central router will be significantly detrimental to the function of the network.

A. Implementation

The algorithms discussed in Section V are implemented in PyTorch. The agents utilise parameter sharing [26] and comprise of a separate network for the actor and the critic. Both algorithms utilise the same actor architecture consisting of a 2-layer Multilayer Perceptron (MLP) followed by a 2-layer LSTM. The critic follows broadly the same structure but have slight differences to allow for architectural differences. MAAC's critic comprises of approximately 1.2 million parameters and utilises a 2-layer MLP followed by a 2-layer LSTM which takes as input the concatenation of all agents observations and outputs the state value. convMAAC's critic comprises of approximately 610 thousand parameters and utilises a 2-layer MLP which encodes the observation of all agents separately, and then all encodings are passed into a 2-layer GCN, the output is then averaged node-wise and passed into a 2-layer LSTM which then outputs the state value. To optimise the model we utilise a grid-search across parameters, where the hyperparameters utilised within the experiments are depicted in Table III. Training is performed on a Nvidia RTX 2080Ti and takes 30 minutes for a full run. WandB [27] is used for experimental logging and the GCN is implemented using DGL [28].

B. Experiments

The parameters used in the experiments are presented in Table I. These are set based on discussions with partners of the NG-CDI project[1] and try to reflect dynamics of real scenarios. It is assumed that components of each asset are independent of each other and maintenance actions take 1 time step to be completed. The actions that involve joint repair of an asset's associated components are the most cost effective solution reflecting the reduction in travel time for engineers. The episodes consists of 100 time-steps which is equivalent to approximately 5 full life-cycles of a component and the agents are trained for 3000 episodes. At each time-step the agents observe five concurrent sensor readings and can request maintenance for the next maintenance window. This resource is constrained to a maximum of two maintenance jobs at a time step for the entire network. Reflecting the assumption that there will be some finite number of available operations engineers at any one time step. If the number of maintenance requests exceed the number of resources, the assignment of resource to the requested jobs is decided randomly among the requested jobs. All network elements are assumed to be deployed simultaneously in factory condition where there is some variance in longevity.

The derived policies are compared to two maintenance baselines including a corrective maintenance policy and a preventive maintenance policy. The corrective policy requests for component replacement upon failure. The preventive policy, takes the form of a distributed oracle which can observe the Remaining Useful life (RUL) of the asset's component and requests maintenance resource when the RUL drops below 20. To demonstrate the impact of critic centralisation Independent Actor-Critic (IAC) [29] agent is implemented too, where the architecture follows a similar structure to convMAAC and MAAC.

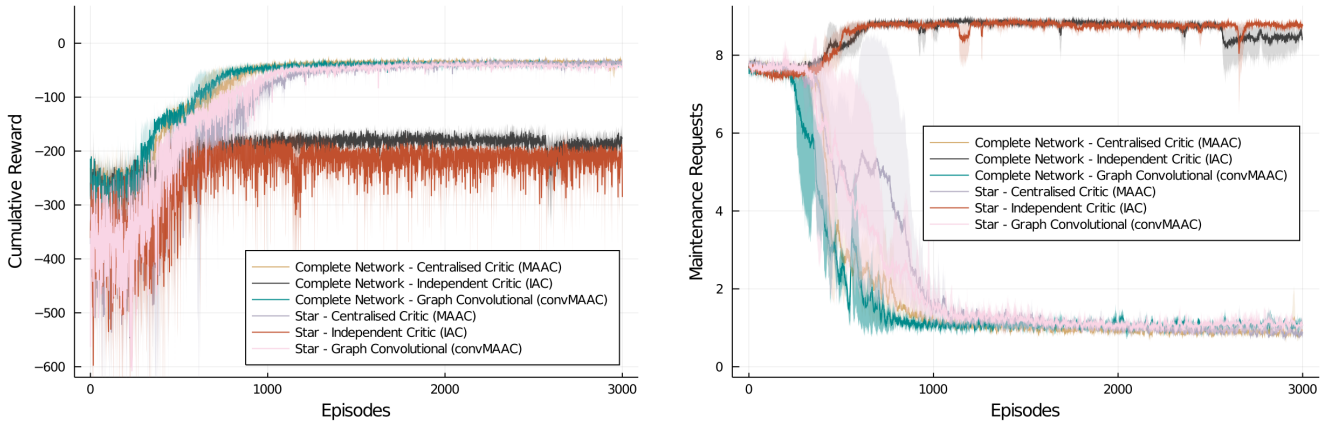
C. Results and Discussion

Comparisons of the algorithms performances for the two topologies analysed can be found in Fig. 3. Fig. 3a shows the cumulative reward obtained as the agents learn their policies. We find that the convMAAC and MAAC are both effective at converging towards a high cumulative reward, with convMAAC learning slightly quicker. IAC appears to encounter difficulties in learning an effective policy, where this policy was found to be approximately uniformly random regardless of the topology. The differences in MARL learning algorithms performance highlight the importance of

[1] https://www.ng-cdi.org/

TABLE II: Final Algorithm performances, where the first number is the mean and the second number the standard deviation.

Model	**Complete Network**		**Star Topology**	
	Network Availability	Maintenance frequency	Network Availability	Maintenance frequency
MAAC	0.9972, 0.0035	0.8981,0.1356	0.998, 0.0019	0.8864, 0.07219
convMAAC	0.9997, 0.0005	1.1001, 0.1525	0.9997, 0.0012	1.092, 0.1678
IAC	0.9803, 0.0088	7.4490, 1.4560	0.97, 0.008	8.4631, 0.0699
Corrective	0.8750, 0.0084	1.1118, 0.0744	0.7911, 0.0181	1.111, 0.0769
Preventive	0.9638, 0.0100	1.0863, 0.1169	0.9384, 0.0331	1.0738, 0.1039



(a) (b)

Fig. 3: Key MARL parameters throughout training. (a) Cumulative reward and (b) Maintenance request per time step.

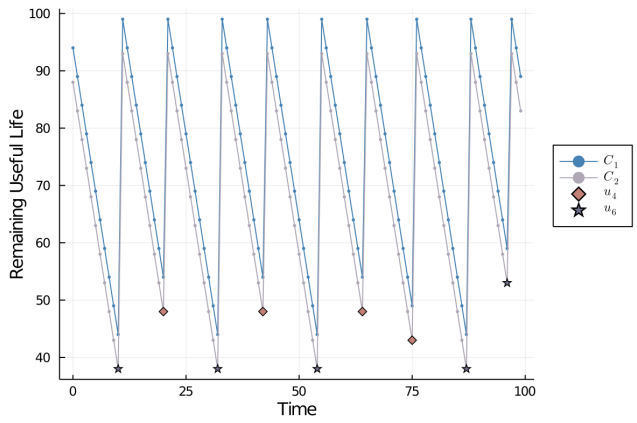


Fig. 4: Example degradation of an asset maintained by MARL agent. C1 and C2 represent components and u_4 and u_6 represent actions 4 and 6 respectively. Where u_4 is a joint replacement and u_6 is a joint repair of both components simultaneously.

the centralised critic in training, as without it, it is difficult to estimate the true value within the environment as it is partially observable. There are no significant performance differences among the two analysed topologies, although the learning convergence is slightly faster in the case of the complete topology. Where, this is likely due to the marginal increase in complexity introduced by the more complex star topology. The behaviour is similar in terms of the number of maintenance requests per time-step (Fig. 3b). Additional

TABLE III: Model Hyperparameters

Model	Actor LR	Critic LR	Gradient Clipping
MAAC	0.0001	0.0005	1.0
convMAAC	0.0001	0.0005	1.0
IAC	0.00005	0.0005	0.5

results for network availability and maintenance frequency are presented in Table II. This shows that MAAC and convMAAC outperform corrective and preventive baselines.

The policies derived by the convMAAC and MAAC in general learn to alternate between full system repair and replacement operations and infer this directly from noisy sensor data. This suggests that, given the experiment parameters, the agents understand that performing repair is a preferable operation to replacement and that coordinating jobs at a single site is cost-effective. An annotated example trajectories for individual assets and the actions taken by the corresponding agent can be found in Fig. 4. It shows that the agents are risk adverse and tend to perform preventive maintenance of the asset when the components have a RUL of approximately 40. Where this is the point in an assets degradation where the probability of failure for an asset starts to increase more rapidly. This is seemingly a pragmatic and reasonable decision if the possibility of contention among agents for the limited maintenance resource available is considered.

Perhaps this could be driven further down by facilitating communication between agents which may enable them to

have a better understanding of the full system state but this comes with associated overheads and privacy concerns. Through this algorithmic implementation no potentially sensitive information has to traverse the network and it achieves remarkable performance considering.

VII. Conclusions and Future Work

This paper provided an analysis of how to apply MARL to a network maintenance problem considering the practicalities of training and deployment. The proposed MARL model was applied to a representative Radio Access Network maintenance task where the top performing algorithm, convMAAC, achieved a network availability increase of 3.49% and 6.13% in the complete and star topologies over a preventive maintenance baseline whilst maintaining similar levels of network maintenance.

In future work we plan to expand the scope and realism of our experimental setting whilst exploring methods to improve algorithmic performance. From an application perspective avenues include evaluation in real-world deployments, considering variation in numbers and type of assets over the network lifetime, and integration of a supervisory agent to select between maintenance jobs in place of the random selection when demand exceeds supply and variation in both the amount of available maintenance resource and cost of the supply. From an algorithmic prospective there is opportunity to consider the multi-agent credit assignment problem and the potential of communications to aid in more effective and conscientious repair policies.

ACKNOWLEDGMENT

The authors would like to thank colleagues from BT for the fruitful discussions.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, oct 2017.

[3] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6382–6393.

[4] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning," Tech. Rep., 2019.

[5] R. Sutton and A. Barto, *Reinforcement Learning - An Introduction*, 2nd ed. Cambridge: MIT Press, 2018.

[6] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 2145–2153.

[7] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, p. 750–797, Oct 2019.

[8] G. K. Chan and S. Asgarpoor, "Optimum maintenance policy with Markov processes," *Electric Power Systems Research*, vol. 76, no. 6-7, pp. 452–456, 2006.

[9] R. Srinivasan and A. K. Parlikad, "Value of condition monitoring in infrastructure maintenance," *Computers & Industrial Engineering*, vol. 66, no. 2, pp. 233–241, 2013.

[10] Z. Liang and A. K. Parlikad, "Predictive group maintenance for multi-system multi-component networks," *Reliability Engineering and System Safety*, vol. 195, no. October 2019, p. 106704, 2020.

[11] J. Wang and X. Zhu, "Joint optimization of condition-based maintenance and inventory control for a k-out-of-n:F system of multi-state degrading components," *European Journal of Operational Research*, vol. 290, no. 2, pp. 514–529, 2021.

[12] M. Knowles, D. Baglee, and S. Wermter, "Reinforcement learning for scheduling of maintenance," *Res. and Dev. in Intelligent Syst. XXVII: Incorporating Applications and Innovations in Intel. Sys. XVIII - AI 2010, 30th SGAI Int. Conf. on Innovative Techniques and Applications of Artificial Intel.*, pp. 409–422, 2011.

[13] S. R. Barde, S. Yacout, and H. Shin, "Optimal preventive maintenance policy based on reinforcement learning of a fleet of military trucks," *Journal of Intelligent Manufacturing*, vol. 30, no. 1, pp. 147–161, 2019.

[14] R. Rocchetta, L. Bellani, M. Compare, E. Zio, and E. Patelli, "A reinforcement learning framework for optimal operation and maintenance of power grids," *Applied energy*, vol. 241, pp. 291–301, 2019.

[15] J. Huang, Q. Chang, and J. Arinez, "Deep reinforcement learning based preventive maintenance policy for serial production lines," *Expert Systems with Applications*, vol. 160, p. 113701, 2020.

[16] X. Wang, H. Wang, and C. Qi, "Multi-agent reinforcement learning based maintenance policy for a resource constrained flow line system," *Journal of Intelligent Manufacturing*, vol. 27, no. 2, pp. 325–333, 2016.

[17] A. Kuhnle, J. Jakubik, and G. Lanza, "Reinforcement learning for opportunistic maintenance optimization," *Production Engineering*, vol. 13, no. 1, pp. 33–41, 2019.

[18] C. P. Andriotis and K. G. Papakonstantinou, "Managing engineering systems with large state and action spaces through deep reinforcement learning," *Reliability Engineering and System Safety*, vol. 191, no. April, p. 106483, 2019.

[19] B. Li and Y. Zhou, "Multi-component Maintenance Optimization: An Approach combining Genetic Algorithm and Multiagent Reinforcement Learning," *2020 Global Reliability and Prognostics and Health Management, PHM-Shanghai 2020*, 2020.

[20] L. S. Shapley, "Stochastic Games," in *Proceedings of the national academy of sciences 39.10*, vol. 39, 1953, pp. 1–7.

[21] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in multi-agent systems and applications-1*, pp. 183–221, 2010.

[22] M. G. Bellemare, S. Candido, P. S. Castro, J. Gong, M. C. Machado, S. Moitra, S. S. Ponda, and Z. Wang, "Autonomous navigation of stratospheric balloons using reinforcement learning," *Nature*, vol. 588, no. 7836, pp. 77–82, 2020.

[23] M. Grieves, *Virtually Perfect: Driving Innovative and Lean Products through Product Lifecycle Management*, 11 2011.

[24] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4295–4304.

[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[26] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *In Proceedings of the Tenth International Conference on Machine Learning*. Morgan Kaufmann, 1993, pp. 330–337.

[27] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[28] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," *arXiv preprint arXiv:1909.01315*, 2019.

[29] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual Multi-Agent Policy Gradients," *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 2974–2982, may 2017.