Srinivasan, Ashvin; Singh, Ugrasen; Tirkkonen, Olav

Multi-Agent Reinforcement Learning Approach Scheduling for In-X Subnetworks

Published in:
Proceedings of the IEEE Vehicular Technology Conference

Accepted/In press: 01/01/2024

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Published under the following license:
Unspecified

Please cite the original version:
Srinivasan, A., Singh, U., & Tirkkonen, O. (in press). Multi-Agent Reinforcement Learning Approach Scheduling for In-X Subnetworks. In *Proceedings of the IEEE Vehicular Technology Conference* IEEE.

Multi-Agent Reinforcement Learning Approach Scheduling for In-X Subnetworks

Ashvin Srinivasan, Ugrasen Singh, and Olav Tirkkonen
Department of Information and Communications Engineering, Aalto University, Finland
{ashvin.1.srinivasan, ugrasen.singh, olav.tirkkonen}@aalto.fi

***Abstract*—We consider radio resource scheduling in a network of multiple non-coordinated in-X subnetworks which move with respect to each other. Each subnetwork is controlled by an independent agent, scheduling resources to devices within the subnetwork. The only information about decisions of other agents is through interference measurements which are non-stationary due to subnetwork mobility and fast fading effects. The agents aim is to serve the devices in their subnetwork with a fixed data rate and a high reliability. The problem is cast as a multi-agent non-stationary Markov Decision Process (MDP), with unknown transition functions. We approach the problem via Multi-Agent Deep Reinforcement Learning (DRL), leveraging Long Short Term Memory (LSTM) networks to handle the non-stationarity and Deep Deterministic Policy Gradient (DDPG) to manage high-dimensional continuous action spaces. Candidate actions given by DRL are quantized to discrete actions by a novel binary tree search method subject to reliability constraints. Simulation results indicate that the proposed LSTM-based DRL scheduling strategy outperforms strategies based on Feed Forward Neural Networks, Centralized Training with Decentralized Execution approaches found in the literature, and conventional heuristic approaches.**

***Index Terms*—6G, In-X subnetworks, URLLC, scheduling, Dynamic Spectrum Allocation, Machine Learning, Non-Stationary MDP, Deep Deterministic Policy Gradient, Multi-agent Reinforcement Learning, Long Short Term Memory.**

I. Introduction

Sixth generation (6G) wireless technology will unleash the Industry 4.0 vision of wireless factory [1]. Replacing cable networks with wireless connectivity is attractive due to its advantages in terms of deployment and maintenance cost, as well as support of mobility. Industry 4.0 imagines a seamless network between every device in a factory, from compact sensors and actuators to industrial robots [2]. Deploying a wireless network within an entity such as a machine, a robot, or a car, with spectrum access guaranteed by an umbrella 6G network, is investigated under the concept of in-X-subnetworks [3]. Control and automation of in-X-subnetworks requires support of cable-like reliability and ultra-low latency. Since subnetworks share the wireless spectrum with other subnetworks and applications, the shared and dynamic nature of the wireless channel poses significant challenges to support such Ultra-Reliable Low-Latency Communications (URLLC).

Recently, URLLC has been widely investigated in the context of industrial wireless services [4], [5]. The reliability of wireless communication is affected by dynamically changing channel states and random interference coming from other users of the channel. Remote control and automation in Industry 4.0 scenarios demand dense deployments of subnetworks supporting URLLC services. These use cases allow random mobility of subnetworks in factory environments, which results in a fading channel, and dynamically changing interference conditions. Lack of coordination among subnetworks can lead to high probability of packet failure. To cope with this, [6] considers a deep learning (DL) algorithm based on an offline-learning principle to establish coordination among the subnetworks.

Due to time varying conditions in a factory environment, performing DL based on observed data over a finite time window may not be a viable solution for providing URLLC services. In dynamic environments, sequentially evolving decision making processes like Reinforcement Learning (RL) stands out as a promising approach for URLLC [7]–[10]. In [7], an actor-critic algorithm was employed to derive an optimal stochastic policy for energy-efficient user scheduling and resource allocation in a heterogeneous network with static base stations. Mobile in-X networks are addressed in [8], [9], while [7], [10] consider static network deployments. In [8], an intelligent radio resource management method using Multi-Agent Deep Reinforcement Learning (MADRL) is proposed, relying only on received signal strength indicators, while [9] considered the problem of allocating one channel to each agent using a Double Deep Q-Network (DDQN). In these works the challenges posed by non-stationarity are solved by assuming centralized training, with the DRL agents learning the transition functions of the whole factory, at the cost of significant communication overhead. In [8], [10], the actions of an individual agent are found, while in [9] a joint action space of all agents is considered, with a small set of actions for each of the agents. In [7], [8], learning happens in a continuous valued action space, with discrete actions found in [7] by direct quantization, and stochastic actions considered in [8]. In [9], DDQN is formulated directly on a discrete action space, becoming computationally infeasible as the number of agents and the action dimensionality grows.

In this paper we address the challenges caused by relative mobility of in-X subnetworks with fully autonomous agents. The agents choose the channels they use, and schedule their devices on these channels, aiming at URLLC service. The main contributions of our research are summarized as follows:

- We consider a decentralized MADRL framework enabling agents to learn independently without direct communica-

tion between agents. This enhances scalability and efficiency for real-time multi-agent scenarios. Additionally, we introduce a State Processing Network (SPN) that integrates an LSTM layer within the Deep Reinforcement Learning (DRL) framework, effectively managing temporal fluctuations in channel conditions. The proposed strategy outperforms the state of the art- Centralized Training with Decentralized execution strategy(CTDE) discussed in the literature.

- We propose a novel binary tree search technique tailored to meet URLLC conditions, assuring a minimum of one resource allocation per device in the sub-network. This method transforms a single continuous action output from the DRL algorithm into multiple discrete candidate actions. From these candidates, optimal discrete action is selected. This is presented as Algorithm 1.

The remainder of the paper is as follows. Section II describes the system model, and the objective of the paper. Section III gives the channel resource allocation policy formulation. The Deep Reinforcement Learning (DRL) framework is presented in Section IV. Simulations are presented in Section V and Section VI concludes the paper.

II. System Model and Problem Formulation

We consider an indoor factory environment where M independent and mobile subnetworks are deployed with entities such as robots or moving machines. Each subnetwork is equipped with a controlling Access Point (AP) and J devices. The total available channel bandwidth is divided into N equal sub-channels. We assume orthogonal transmissions within the subnetwork to avoid intra-subnetwork interference. We assume a fully scheduled system, where the AP has full knowledge of the signal-to-interference plus noise ratio (SINR) of its devices on all sub-channels. The AP controlling a subnetwork sends control signals to devices to indicate scheduling decisions. We consider downlink transmission; principles governing uplink scheduling within a subnetwork would be similar.

The subnetworks share the same radio resources in an uncoordinated manner and can transmit at the same time. A sub-channel can thus be occupied by multiple subnetworks, which leads to inter-subnetwork interference. Collisions of channels occupied by sub-networks can be avoided by implementing an online learning algorithm at the sub-networks. We assume slotted channel access at all subnetworks, such that all subnetworks have the same understanding of slotting. Slot timing is controlled by an umbrella wide-area network. We consider a service model where an AP sends new data packets to devices in each time-slot. Instead of retransmitting failed packets, fresh packets are transmitted [11].

Since sub-networks are mobile over time, the fading channels associated with them can vary with time. The channel gain coefficient from AP $m \in \{1, 2, ..., M\}$ to device $j \in \{1, 2, ..., J\}$ on sub-channel $n \in \{1, 2, ..., N\}$ at time t is denoted as $h^n_{j,m}(t)$. The received signal at device j served by subnetwork b is

$$y^n_{j,b}(t) = h^n_{j,b}\ x_b + \sum_{m \neq b} h^n_{j,m}(t)\ x_m\ g_{n,m} + z_j, \qquad (1)$$

where $\mathbb{E}\left[|x_m|^2\right] = \mathbb{E}\left[|x_b|^2\right] = P_t$, x_b is the symbol transmitted by from the serving AP b to device j, x_m is a symbol transmitted by interfering AP m, and $z_j \sim \mathcal{CN}(0, \sigma^2)$ is Additive White Gaussian Noise (AWGN) at device j. The SINR at device j using channel n is thus

$$\gamma_{n,j}(t) = \frac{|h^n_{j,b}|^2}{\sum_{m \neq b} |h^n_{j,m}(t)|^2 g_{n,m} + \frac{1}{\gamma_0}}, \qquad (2)$$

where $\gamma_0 = P_t/\sigma^2$ is the ratio of the transmit power and the noise power. A fixed number of information bits is transmitted to each device in each slot. A variable number of subchannels may be used for transmission. In each slot, the AP agent has to solve the scheduling problem, selecting a set of channels for the transmission towards each device. With $\mathbf{a}^{(b)}_j$, the $N \times 1$ vector with entries $a^{(b)}_{n,j} \in \{0, 1\}$ indicating the channels allocated to device j, the solution of the scheduling problem of AP b is the $N \times J$ matrix $\mathbf{A}^{(b)}$ where $\mathbf{a}^{(b)}_j$ are columns. Then, the interference indicators are $g_{n,m} = \sum_j a^{(m)}_{n,j}$.

We model error performance by assuming a fixed Forward Error Correction (FEC) and modulation scheme within a subchannel, and repetition coding across subchannels. This provides an upper bound of error probability. The receiver Chase-combines the channels that are allocated to it before FEC decoding, resulting in the combined SINR

$$\gamma_j = \sum_{n=1}^{N} a^{(b)}_{n,j}\, \gamma_{n,j}\,. \qquad (3)$$

The corresponding probability of erroneously decoding the codeword for device j served by AP b can then be modeled as [12]

$$p_{j,b} = Q\left(\frac{2q\ C(\gamma_j) - 2k + \log_2(q)}{2\sqrt{q}\sqrt{V(\gamma_j)}}\right), \qquad (4)$$

where $Q(.)$ is the Gaussian q-function, $C(\gamma) = \log_2(1+\gamma)$ is the Shannon's channel capacity, $V(\gamma) = \gamma\frac{2+\gamma}{(1+\gamma)^2}(\log_2 e)^2$ is channel dispersion, and $R = k/q$ the code rate.

The objective is to learn to avoid inter-subnetwork interference to enhance wireless link reliability of subnetworks in a non-stationary environment, with an emphasis on maintaining consistent performance across time. Accordingly, we formulate the resource scheduling optimization problem as

$$P_1 : \min \sum_{j,b} p_{j,b}(t) \quad \text{s.t.} \quad p_{j,b}(t) \leq p_o, \qquad (5)$$

where p_0 is a predefined threshold, which for URLLC could typically be 10^{-5}.

In the fully distributed solution approach that we pursue, the scheduling decisions of an AP affects the decisions of other APs, which in turn affect the state of the environment as seen

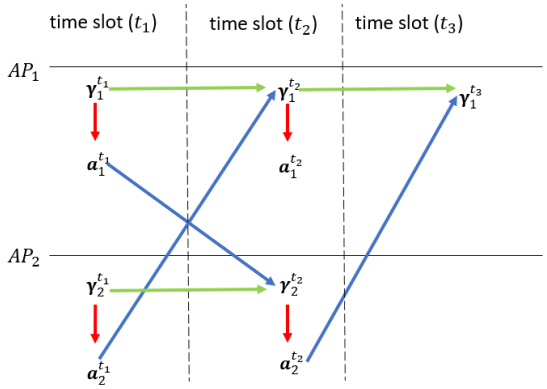

Fig. 1: Timing interaction between AP_1 and AP_2 depicting the SINR-dependent channel allocation and its subsequent impact on adjacent APs across discrete time frames, the effect of the actions of an AP on SINRs is one-slot delayed.

by the AP, i.e. the SINRs. Scheduling at an AP is based on interference measurements performed by the devices which is fed back to the AP. As we assume subnetwork-synchronous slotted access, the interference measurements governing the transmissions in a slot come from the previous downlink slot. The indirect effect, mediated by the environment, of the actions of an AP on the SINRs of its own devices will thus not be immediately seen in next-slot SINRs. The effect of the actions on SINRs will be *one-slot delayed*, complicating the interpretation of the problem as a Markov decision process (MDP). The timing diagram in Fig. 1 elucidates the interaction dynamics between two subnetworks. At time slot t_1, $\boldsymbol{\gamma}_1^{t_1}$ are measured in subnetwork 1. The AP determines the optimal channels $\boldsymbol{a}_1^{t_1}$, with the scheduling decision indicated by a red line. This channel allocation influences the $\boldsymbol{\gamma}_2^{t_2}$ measured in subnetwork 2 at time slot t_2, as depicted with a blue line. Consequently, it is crucial to recognize that in the timing mechanism, $\boldsymbol{\gamma}_m^{t+1}$ at any subsequent time slot $t+1$ for a given AP m is influenced by $\boldsymbol{\gamma}_m^t$ at the current time slot t as well as the channel allocation, $\boldsymbol{a}_m^{t-1}$ from the preceding time slot $t-1$. This dependency is vital for the development of the reinforcement learning problem that is addressed later.

III. Policy Formulation

We first formulate a non-stationary MDP characterized as $(\mathcal{S}, \mathcal{A}, P_T(:,t), r(\cdot), \eta)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, respectively. The transition probability $P_T(:,t)$ varies with time and describes the dynamics of the system environment. The function $r(\cdot)$ is used to calculate the immediate reward, and η represents the discount factor, which influences the importance of future rewards. At time t, the system is described by its current state $\mathbf{S}(t) \in \mathcal{S}$. The action taken at time t is $\mathbf{A}(t) \in \mathcal{A}$. The transition probability $P_T(\mathbf{S}(t+1)|\mathbf{S}(t), \mathbf{A}(t); t)$ is crucial, as it indicates the probability of moving to a new state $\mathbf{S}(t+1)$ from the current state $\mathbf{S}(t)$ when action $\mathbf{A}(t)$ is taken, highlighting the time-dependent nature of the system state transitions.

Each subnetwork is equipped with a control mechanism responsible for the allocation of channel resources to various devices, represented by the action matrix $\mathbf{A}(t)$.

The state of the environment, as seen by an AP agent, is primarily characterized by the SINRs (2) of its devices, which is an $N \times J$ matrix with real non-negatives values, $\boldsymbol{\Gamma}(t)$.

In an MDP, the transition probabilities at time t depend on the state and the action taken at time t. To cope with the *one-slot delayed* effect of the actions, depicted in Fig. 1, we expand the state space by incorporating the preceding action $\mathbf{A}(t-1)$ in the state. Accordingly, the state vector for a subnetwork is formally defined as:

$$\mathbf{S}(t) = [\boldsymbol{\Gamma}(t), \mathbf{A}(t-1)]. \tag{6}$$

This leads to the formulation of the state transition probability at time t as follows:

$$\begin{aligned} &P\big(\mathbf{S}(t+1) \mid \mathbf{S}(t), \mathbf{A}(t)\big) \\ &= P\big(\boldsymbol{\Gamma}(t+1), \mathbf{A}(t) \mid \boldsymbol{\Gamma}(t), \mathbf{A}(t-1), \mathbf{A}(t)\big) \\ &= P\big(\boldsymbol{\Gamma}(t+1)|\boldsymbol{\Gamma}(t), \mathbf{A}(t-1)\big) P\big(\mathbf{A}(t)|\boldsymbol{\Gamma}(t), \mathbf{A}(t-1), \mathbf{A}(t)\big). \end{aligned} \tag{7}$$

Thus by incorporating the previous action into the state definition, we create an MDP framework that also encapsulates the dynamics of SINR as discussed in Fig. 1.

The objective of this paper is to reduce the packet loss probability, a critical metric for ensuring high reliability in communication networks. For this, we formulate the reward function

$$r(t) = -\sum_{j=1}^{J} \exp\left(\frac{p_{j,b}(t)}{p_0} - 1\right). \tag{8}$$

This reward function imposes an exponential penalty on the network when the packet loss probability exceeds the specified threshold p_0, while if the packet loss probability is below this threshold, the penalty decreases rapidly but continuously towards zero, making it amenable to gradient search.

IV. Reinforcement Learning Framework

We treat the AP controlling each subnetwork as an independent RL agent. Each agent's objective is to increase its cumulative reward with the primary goal of minimizing its packet loss probability below a predefined threshold. The discounted cumulative reward starting from time-slot t till a time horizon T is $R_{\text{ac}}(t) = \sum_{k=t}^{T} \eta^{k-t} \mathbb{E}\Big[r(k)\Big]$. The optimization problem for each RL agent is formulated as:

$$P_2: \quad \max_{\mathbf{A}(t)} R_{\text{ac}}(t), \ 0 \le t \le T, \tag{9}$$

where $\mathbf{A}(t)$ is the action. In our scenario, the discrete actions per subnetwork are exponentially numerous (2^{NJ}), the Deep Q-Network (DQN) approach [13] becomes impractical due to its suitability for low-dimensional action spaces. Given our need for deterministic policies over stochastic policies produced by Proximal Policy Optimization (PPO), and Advantage Actor Critic (A2C), we pivot to the Deep Deterministic Policy Gradient (DDPG) algorithm [13] for optimal policy learning. However, to accommodate our discrete action requirement in a

framework originally designed for continuous action spaces, we employ a modified DDPG algorithm adapted for high-dimensional discrete actions.

In our MADRL framework, each subnetwork AP operates autonomously as an independent agent. They aim to identify the best scheduling policy by maximizing (9), enhancing network reliability through decentralized decision-making without coordination.

A. Actor, Critic and State Processing Networks

In the *State Processing Network* (SPN), we incorporate an LSTM layer coupled to a fully connected layer. The adoption of LSTM, known for its proficiency with sequential data, is crucial for learning patterns from historical data in non-stationary environments [14]. This integration facilitates the identification of optimal actions using fewer samples.

The *proto actor network* generates a continuous-valued action estimate $\hat{\mathbf{A}} = f_\phi(\mathbf{S}) \in \mathcal{R}^{N\times J}$, while the *critic network* provides an estimate of the Q-value function, formally defined as $Q_\theta(\mathbf{S},\mathbf{A}) = \mathbb{E}\big[\sum_{k=t}^{T} \eta^{k-t} r(k) | (\mathbf{S},\mathbf{A});\theta\big]$ [13].

To determine an instantaneous schedule, the continuous action given by the proto actor has to be quantized to a discrete action. This is a closest lattice point problem, which is NP hard. We find an approximate solution by list decoding, performing the process in two steps. In the first step, we identify a set $\mathcal{Z}$ of K candidate starting points $\mathbf{A}_k$ with entries $a_{k,n,j} \in \{0,1\}$, where precisely one resource is assigned to each user. The motivation for this is that as we strive for URLLC, it is not an option to have a device without a scheduled resource. We begin by sorting the entries in $\hat{\mathbf{A}}$ and selecting the J largest entries. If this set does not form a legal schedule, where each device is assigned a unique resource, we incrementally expand the set. We continue this process until we have K candidate starting points.

After the set of starting points is created, we complete each schedule by greedily assigning the remaining resources to users based on their proximity to $\hat{\mathbf{A}}$, or by leaving them unassigned. To do this, we describe possible scheduling decisions for a resource by matrix $\mathbf{V}$ consisting of a zero vector and an identity matrix of size J: $\mathbf{V} = \begin{bmatrix}\mathbf{0}_J & \mathbf{I}_J\end{bmatrix}^{\mathrm{T}}$. With $\mathcal{U}_k$ the set of non-allocated resources in starting point $\mathbf{A}_k$, and $(\mathbf{B})_n$ the nth row vector of matrix $\mathbf{B}$, we complete $\mathbf{A}_k$ using

$$(\mathbf{A}_k)_n = \underset{\mathbf{v}\in\mathbf{V}}{\arg\min} \left\| (\hat{\mathbf{A}})_n - \mathbf{v} \right\|_2 , \; n \in \mathcal{U}_k \,, \tag{10}$$

where we treat $\mathbf{V}$ as a set of row-vectors.

As a result, we get a set $\mathcal{Z}$ with up to K distinct matrices, each representing a possible resource allocation. The algorithm for creating K candidate quantizations is summarized in Algorithm 1. We refer to the collective setup of the proto actor network and the K-candidate quantization algorithm as the actor network.

For each of the K candidate discrete actions, a Q-value is computed by the critic network. The action is then chosen as the one giving the highest Q value:

$$\mathbf{A} = \underset{\mathbf{A}_k\in\mathcal{Z}}{\arg\max}\; Q_\theta(\mathbf{S},\mathbf{A}_k). \tag{11}$$

Algorithm 1 K-candidate Quantization

Input: $\hat{\mathbf{A}} \in \mathcal{R}^{N\times J}$, $0 \le \hat{a}_{n,j} \le 1$
Output: Set $\mathcal{Z}$ of K discrete actions.

```
Sort entries in Â in descending order, creating a list A of pairs (n, j) of a resource and a user. Set m = J.
while |Z| < K do
    Take set P of m first entries from A.
    Construct all legal schedules A_k from P, add them to Z
    m = m + 1
end while
Replace Z with the set of the schedules in Z with the largest weights Σ_{n,j} a_{k,n,j} â_{n,j}
for k=1, ... , K do
    Complete A_k ∈ Z by assigning unallocated resources as in (10).
end for
```

B. DDPG Algorithm

The actor and critic networks collectively generate the discrete action $\mathbf{A}_t$ for a given state input $\mathbf{S}_t$. An immediate reward r_t along with the next state $\mathbf{S}_{t+1}$ is obtained based on the agent interaction with the environment. These transitions are stored in buffer $\mathcal{B}$ and are sampled to update both networks.

While updating, the agent maintains a target actor $f_{\phi'}$ and critic $Q_{\theta'}$ which are updated more smoothly than the actor and critic. When updating, a set of samples $\mathcal{S} \subset \mathcal{B}$ of the buffer is chosen. For each sample, the discounted estimated Q-value is computed from the target networks as

$$y(i) = r(i) + \eta\, Q_{\theta'}(\mathbf{S}_{i+1}, f_{\phi'}(\mathbf{S}_{i+1})). \tag{12}$$

The parameters θ of the critic are then updated to minimize the loss

$$L(\theta) = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|}[y(i) - Q_\theta(\mathbf{S}_i,\mathbf{A}_i)]^2, \tag{13}$$

where $|\mathcal{S}|$ is the cardinality of the subset $\mathcal{S}$, and the actor parameters are updated by the gradient [13]:

$$\nabla_\phi f_\phi = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} \nabla_{\mathbf{A}} Q_\theta(\mathbf{S}_i,\mathbf{A})|_{\mathbf{A}=f_\phi(\mathbf{S}_i)}\; \nabla_\phi f_\phi(\mathbf{S}_i) \tag{14}$$

The target networks are softly updated:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta', \quad \phi' \leftarrow \tau\phi + (1-\tau)\phi'. \tag{15}$$

In the DDPG algorithm, actor and critic networks learn from a shared experience buffer, which improves decision-making by aligning estimated values with target outcomes. The pseudo code for this approach is detailed in Algorithm 2, iterating over $E_{\max}$ epochs.

V. Simulation Results

To evaluate the proposed DRL policy, we consider an indoor factory setting (20 × 20 meters). We have 5 subnetworks in the factory acting as RL agents, and each sub-network consists of one AP and 5 devices. The radius of a sub-network is 1m, and AP is in the center of the subnetwork. The subnetworks

Algorithm 2 Multi Agent DDPG algorithm

For each agent m, randomly initialize: critic network Q_{θ^m} and actor network f_{ϕ^m} with weights θ^m and ϕ^m.
Initialize target critic network $Q_{\theta^{m'}}$ and actor network $f_{\phi^{m'}}$ with weights $\theta^{m'}$ and $\phi^{m'}$. $\theta^{m'} \leftarrow \theta^m$, $\phi^{m'} \leftarrow \phi^m$.
Initialize replay buffer $\mathcal{B}^m$
Set the state $\mathbf{S}_0^m$ at random.
for epoch = 1 to $E_{\max}$ **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  if epoch $\neq$ 1 **then**
    Set state as $\mathbf{S}_T^m$.
  end if
  for t = 1 to T **do**
    Select candidate proto action $\hat{\mathbf{A}}^m = f_{\phi^m}(\mathbf{S}_T^m) + \mathcal{N}_t$
    Select discrete action $\mathbf{A}_t^m$ as per Algorithm 1 and (11).
    Execute action $\mathbf{A}_t^m$ and observe reward r_t^m and new state $\mathbf{S}_{T+1}^m$ given by the environment.
    Store transition $(\mathbf{S}_T^m, \mathbf{A}_t^m, r_t^m, \mathbf{S}_{T+1}^m)$ in buffer $\mathcal{B}^m$
    Sample a random mini batch $\mathcal{S} \subset \mathcal{B}^m$
    Set target $y^m(i)$ as per (12).
    Update the critic, Q_{θ^m} and actor network, f_{ϕ^m} as per the updating rules in (13), and (14) respectively.
    Update the target networks (15).
  end for
end for

move in random directions in the factory, and the distances between subnetworks change accordingly. The simulated scenario is summarized in Table I. Python scripting language is used to create the simulation environment, train agents, and produce plots. DRL framework is implemented by PyTorch.

For simplicity, we model intra-subnetwork channels as static over time. Further, due to moving nature of subnetworks in the factory environment, inter-subnetwork channels are time varying. The dynamics of their small-scale fading components follow the Jakes model [15], $\xi(t) = \frac{1}{\sqrt{L}} \sum_{l=1}^{L} c_l \, e^{i(2\pi f_l \, t + \psi_l)}$. There are L multipath components, c_l is the path gain, f_l the Doppler frequency and $\psi_l \in (0, 2\pi)$ the static phase shift of component l.

We use the Indoor-factory path loss model InF:SL of [16],

TABLE I: Simulation parameters

Parameter	Value
Deployment area [m^2]	20 × 20
Number of subnetworks, M	5
Number of devices per subnetwork, J	5
Radius of subnetworks [m]	1
Minimum distance between subnetwork centers [m]	2
Minimum distance between devices in sub-network [m]	0.2
Subnetwork velocity [km/h]	40
Carrier frequency f_c [GHz]	1.3
Transmit power P_t	20 dBm
Rx noise figure [dB]	10
Bandwidth	18 MHz
Number of subchannels, N	30
Data symbols per subchannel	500
Information packet size [bits]	400
Modulation	QPSK
Code rate, R	0.4
Packet loss probability target p_0	10^{-5}

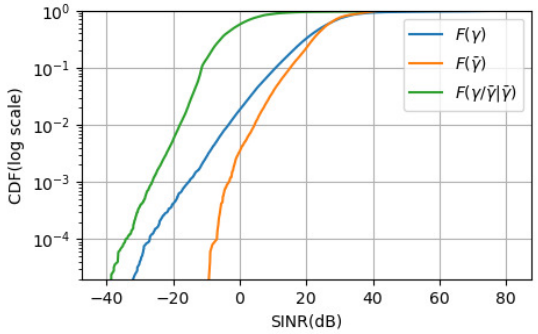

Fig. 2: SINR distributions with reuse 1 transmission.

with the path loss

$$\rho(d) = \max(\mathrm{PL}_{\mathrm{NLoS}}(d), \mathrm{PL}_{\mathrm{LoS}}(d))$$

being the larger of the LoS and non-LoS (NLoS) path losses

$$\begin{aligned} \mathrm{PL}_{\mathrm{LoS}}(d) &= 31.84 + 21.50 \log_{10}(d) + 19 \log_{10}(f_c), \\ \mathrm{PL}_{\mathrm{NLoS}}(d) &= 33 + 25.5 \log_{10}(d) + 20 \log_{10}(f_c)\,. \end{aligned}$$

Here, f_c is the carrier frequency and d is the propagation distance in meters. The channel coefficient on subchannel n from AP m to device j with distance $d_{j,m}$ is thus modeled as $h_{j,m}^n(t) = \sqrt{\beta_{j,m}^n}\xi_{j,m}^n(t)$, where $\beta_{j,m}^n = 10^{-\rho(d_{j,m})/10}$.

For the proposed DRL architecture, states $\mathbf{S}(t-1)$ at time slot $t-1$ and $\mathbf{S}(t)$ at time slot t are provided as inputs to the LSTM layer, whose output is projected onto a single fully connected layer of 256 neurons. The actor and critic networks consists of fully connected hidden layers of dimensions 512, 256 neurons. For our work, we choose the hyper parameter $K = 10$ for step 2 in Algorithm 1. The activation functions used are *ReLu*$(\cdot)$, only with the exception of the final activation function at the actor as *Sigmoid*$(\cdot)$. *Adam* optimizer is used with learning rates as $\alpha_\theta = 10^{-3}$ and $\alpha_\phi = 10^{-3}$ for actor and critic network respectively. The batch size is set to 64. The value of $\tau = 0.005$ for the updation of target networks, discount factor $\eta = 0.98$, and number of epochs $E_{\max}$ set to 4000.

Figure 2 depicts distributions of SINRs resulting from this model, if universal frequency reuse 1 were used in the factory, such that all APs would transmit on all subchannels. $F(\gamma)$ is the cumulative distribution function (CDF) of the SINRs experienced by devices on subchannels over the lifetime of the network. The average SINR is 22.9 dB. The distribution $F(\bar{\gamma})$ represents the average SINR of a subnetwork in a given slot obtained by averaging over devices and subchannels, while $F(\gamma/\bar{\gamma}\,|\,\bar{\gamma})$ shows the variation of the subchannel SINRs across devices and channels, as compared to $\bar{\gamma}$.

$F(\gamma/\bar{\gamma}\,|\,\bar{\gamma})$ captures the effect SINR variability caused by fast fading, and different distances of devices from the subnetwork center. In contrast, $F(\bar{\gamma})$ shows the effect of the mobility of the subnetworks on the average SINRs. In rare occasions when multiple subnetworks are close to each other, the average SINR in a slot may be less than 0 dB.

Figure 3 showcases the training performance of five LSTM-based agents within a reinforcement learning framework over

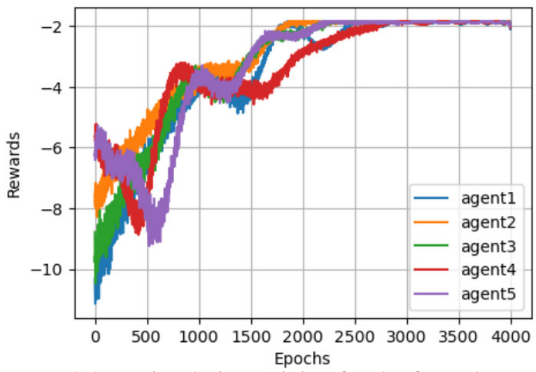

Fig. 3: Reward dynamics during training for the five sub-networks in an instance.

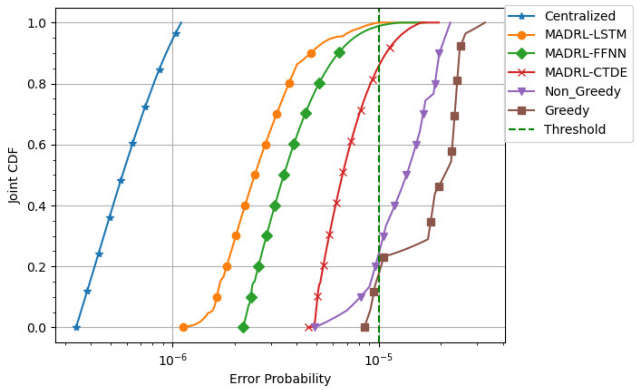

Fig. 4: CDF of Error Probabilities for Reinforcement Learning Agents with LSTM and FFNN Networks, compared against centralized, myopic strategies, and CTDE strategies found in literature.

4000 epochs. Initially, all agents exhibit poor performance, receiving higher negative rewards. However, as training progresses, there is a significant improvement in performance, with reward values increasing and converging for all agents, indicating that the LSTM networks effectively learn and optimize the policy over time. The plateauing of reward curves suggests that a performance ceiling is reached.

Figure 4 displays the CDF of error probability resulting from Algorithm 2 using LSTM networks against alternative methods. A strategy that centralizes optimization by considering the joint state and action space achieves the best results which serves as the lower bound. However, this method becomes difficult to manage as more sub-networks are added. The FFNN based DRL architecture has only fully connected layers and no LSTM layer. The centralized training with decentralized execution (CTDE) approach is taken from [8], [10], with the difference that we only consider channel allocation, assuming a fixed power allocation. One policy is learned from the collective experiences of all sub-networks, and the agents use this policy to derive their actions in their current state. In addition, myopic greedy and non-greedy one-shot strategies are considered, where the agents schedule only based on the current SINR situation. LSTM-based DRL outperforms all benchmark strategies except the centralized one, demonstrating

TABLE II: Error Probability across different architectures

Architecture	Agent	Mean(10^{-6})	SD(10^{-6})
LSTM	1	2.91	1.47
	2	2.89	1.46
	3	2.88	1.48
	4	2.92	1.53
	5	2.93	1.51
FFNN	1	3.99	1.73
	2	4.01	1.71
	3	3.98	1.68
	4	3.97	1.67
	5	4.1	1.74

the viability of LSTM to capture the non-stationarity, and that distributed agents excel over a single agent learning an average policy.

A myopic greedy channel allocation is found by an AP by optimizing $\max_{\mathbf{A}} \min_{j} \boldsymbol{\gamma}_j^{\mathrm{T}} \boldsymbol{a}_j$, where $\boldsymbol{\gamma}_j$ is the vector of SINRs of device j, and $\boldsymbol{a}_j$ indicates its allocation vector. The non-greedy strategy is formulated through solving the linear program $\min \sum_{j,n} a_{j,n}$ subject to $\boldsymbol{\gamma}_j^T \boldsymbol{a}_j > \gamma_{th}$, where the objective is to minimize the resource consumption while ensuring that the sum of SINRs on the allocated channels exceeds the threshold γ_{th} for each device.

In a multiagent scenario, it is important to compare the learning performance of different agents. Table II presents a quantitative analysis of the performance of five agents trained with LSTM and FFNN networks. The performance is evaluated in terms of the mean error probability and the standard deviation (SD) of the error.

Our service model has an AP send new packets to devices at regular intervals, without retransmission of failed ones, focusing on ensuring successful delivery of fresh data [11]. By evaluating error probabilities from SINR data, we track the delay between successful delivery of data packets to devices. Normal latency between transmissions corresponds to one slot. A failed transmission followed by a successful one causes excess latency of one slot, and two consecutive failures causes excess latency of two slots. Table III shows the probabilities of excess latency events for LSTM and FFNN architectures.

In Figure 5, we analyze the number of resources used by an agent, and the corresponding effect on error probability. The top figure shows the CDF of the resource consumption for different strategies. For the MADRL-LSTM strategy, in 99% of instances, 22 or fewer resources are utilized. In 0.1% of the cases, an AP uses all available resources. In the bottom figure, the average error probability as a function of the number of resources utilized is shown. Events with an average error probability greater than or equal to 10^{-5} typically occur during full resource utilization. These patterns

TABLE III: Data update latencies for different architectures

Architecture	Excess Latency	Value
LSTM	0	0.999988
	1	1.10796×10^{-5}
	2	1.23085×10^{-10}
FFNN	0	0.999988
	1	1.145048×10^{-5}
	2	$1.2769060 \times 10^{-10}$

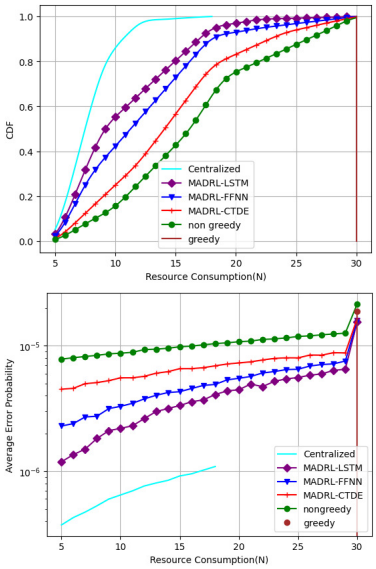

Fig. 5: Top: CDF of resource consumption for the different strategies. Bottom: Expected error probability vs the number of resources consumed in an instance.

indicate that the learning agents are well capable of handling the uncertainties related to changes in the channel environment between the time of making the scheduling decision, and the time of transmission. These events typically appear when many resources are needed. This indicates that while most of the time, the amount of resources is abundant, during the rare occasions when multiple subnetworks are close to each other, the resources are not sufficient to guarantee reliable service.

VI. Conclusion

In this paper, we considered channel allocation for an indoor factory setting consisting of mobile in-X subnetworks. The primary objective was to enhance the overall network reliability through decentralized training of all agents based on MADRL framework. We formulated a non-stationary MDP, where the state of a subnetwork consists of the SINRs of its devices on different channels, and the previous scheduling decision, while the actions are scheduling decisions. Non-stationarity arises both due to changes in the fast fading, and due to large scale mobility of subnetworks with respect to each other. The reward function was formulated to incentivize URLLC.

To solve the non-stationary MDP, We applied Reinforcement Learning with an LSTM-based DDPG architecture incorporating a novel mechanism to generate discrete action. The proposed scheme demonstrated superior performance compared to FFNN-based DDPG and myopic optimization strategies. Furthermore, the proposed scheme outperformed the centralized training with decentralized execution scheme as used in the literature. For future work, we plan to incorporate multiple time slots to better leverage the efficiency of LSTMs and simplify the overall architecture.

Acknowledgement

This work was funded in part by the Academy of Finland (grant 345109) and by Business Finland under the project "Extreme Machine Type Communications for 6G".

References

[1] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G networks: Use cases and technologies," *IEEE Communications Magazine*, vol. 58, no. 3, pp. 55–61, 2020.

[2] A. M. Ramly, N. F. Abdullah, and R. Nordin, "Cross-layer design and performance analysis for ultra-reliable factory of the future based on 5g mobile networks," *IEEE Access*, vol. 9, pp. 68 161–68 175, 2021.

[3] G. Berardinelli & al., "Extreme communication in 6G: Vision and challenges for 'in-x' subnetworks," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 2516–2535, 2021.

[4] F. Hamidi-Sepehr, M. Sajadieh, S. Panteleev, T. Islam, I. Karls, D. Chatterjee, and J. Ansari, "5G URLLC: Evolution of high-performance wireless networking for industrial automation," *IEEE Communications Standards Magazine*, vol. 5, no. 2, pp. 132–140, 2021.

[5] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF detnet standards and related 5G ULL research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2019.

[6] R. Adeogun, G. Berardinelli, and P. Mogensen, "Learning to dynamically allocate radio resources in mobile 6G in-X subnetworks," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2021, pp. 1–5.

[7] Y. Wei, F. R. Yu, M. Song, and Z. Han, "User scheduling and resource allocation in HetNets with hybrid energy supply: An actor-critic reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 17, no. 1, pp. 680–692, 2018.

[8] X. Du, T. Wang, Q. Feng, C. Ye, T. Tao, L. Wang, Y. Shi, and M. Chen, "Multi-agent reinforcement learning for dynamic resource management in 6G in-X subnetworks," *IEEE Transactions on Wireless Communications*, vol. 22, no. 3, pp. 1900–1914, 2023.

[9] R. Adeogun and G. Berardinelli, "Distributed channel allocation for mobile 6G subnetworks via multi-agent deep Q-learning," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2023, pp. 1–6.

[10] X. Tan, L. Zhou, H. Wang, Y. Sun, H. Zhao, B.-C. Seet, J. Wei, and V. C. M. Leung, "Cooperative multi-agent reinforcement-learning-based distributed dynamic spectrum access in cognitive radio networks," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 19 477–19 488, 2022.

[11] S. R. Khosravirad, O. Tirkkonen, U. Parts, L. Zhou, D. Korpi, P. Baracca, and M. A. Uusitalo, "Communications survival strategies for industrial wireless control," *IEEE Network*, vol. 36, no. 2, pp. 66–72, 2022.

[12] X. Zhang, Q. Zhu, and H. V. Poor, "Non-asymptotic performance for finite blocklength coding over Nakagami-m channels," in *IEEE International Conference on Communications (ICC)*, pp. 1–6.

[13] T.P. Lillicrap & al., "Continuous control with deep reinforcement learning." in *ICLR*, 2016.

[14] T. Ni, B. Eysenbach, and R. Salakhutdinov, "Recurrent model-free RL can be a strong baseline for many POMDPs," in *International Conference on Machine Learning*, 2022, pp. 16 691–16 723.

[15] William C. Jakes, Ed., *Microwave Mobile Communications*. John Wiley & Sons, 1975.

[16] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz," Tech. Rep. TR 38.901, V16.1.0, Dec. 2019.