Multi-Agent Actor-Critic with Harmonic Annealing Pruning for Dynamic Spectrum Access Systems

George Stamatelis*, Angelos-Nikolaos Kanatas†, and George C. Alexandropoulos*
*Department of Informatics and Telecommunications, National and Kapodistrian University of Athens,
Panepistimiopolis Ilissia, 16122 Athens, Greece
†School of Electrical and Computer Engineering, National Technical University of Athens,
Zografou Campus, 15772 Athens, Greece
e-mails: {georgestamat, alexandg}@di.uoa.gr, el19169@mail.ntua.gr

***Abstract*—Multi-Agent Deep Reinforcement Learning (MADRL) has emerged as a powerful tool for optimizing decentralized decision-making systems in complex settings, such as Dynamic Spectrum Access (DSA). However, deploying deep learning models on resource-constrained edge devices remains challenging due to their high computational cost. To address this challenge, in this paper, we present a novel sparse recurrent MARL framework integrating gradual neural network pruning into the independent actor global critic paradigm. Additionally, we introduce a harmonic annealing sparsity scheduler, which achieves comparable, and in certain cases superior, performance to standard linear and polynomial pruning schedulers at large sparsities. Our experimental investigation demonstrates that the proposed DSA framework can discover superior policies, under diverse training conditions, outperforming conventional DSA, MADRL baselines, and state-of-the-art pruning techniques.**

***Index Terms*—Dynamic spectrum access, multi-agent systems, reinforcement learning, neural network pruning.**

I. Introduction

Modern wireless communications are witnessing a rapid surge in the number of devices contending for limited spectrum resources. Since the available spectrum is inherently sparse, this increased demand underscores the need for effective spectrum management schemes. In this paper, we focus on the Dynamic Spectrum Access (DSA) problem, specifically, its distributed variant, where multiple users broadcast messages over a shared set of orthogonal channels to maximize network utility, all without direct coordination. Upon transmitting on a given channel, a user receives a binary observation indicating whether the transmission succeeded or failed due to interference from others, as well as information about the transmit Signal-to-Noise Ratio (SNR). Consequently, each user must not only determine the most suitable channels for their location and equipment, but also infer the access strategies of their peers to minimize collisions and optimize channel utilization.

Deep Reinforcement Learning (DRL) algorithms have emerged as a powerful candidate for finding channel allocation policies, but their inherent issues regarding neural network over-parameterization, storage, and computation requirements, especially when being deployed in lightweight Internet of Things (IoT) devices, remain largely unaddressed.

This work has been supported by the SNS JU project 6G-DISAC under the EU's Horizon Europe research and innovation program under Grant Agreement No 101139130. George Stamatelis' contribution is funded by the Hellenic Foundation for Research and Innovation (HFRI) under the 5th Call for HFRI PhD Fellowships (Fellowship Number: 21080).

A. Background

a) DRL for DSA: Dynamic spectrum management problems, including DSA, have been thoroughly studied in [1], and optimal solutions have been developed for certain special cases (e.g., [2]). Due to the difficulty of realistic DSA problems, various methods for finding approximately optimal policies have been proposed, including greedy algorithms [3], conventional optimization (i.e., without deep neural networks), reinforcement learning [4], and bandits [5]. Recent research on DSA problems (e.g., [6]) applies existing DRL algorithms for Partially Observable Markov Decision Processes (POMDPs), with elaborate modifications to handle the unique DSA challenges. Distributed Multi-Agent DRL (MADRL) algorithms, using the Centralized Training with Decentralized Execution (CTDE) principle [7], have been also studied in [8]–[10]. The existing approaches for MARL-based DSA fall into three main categories. The first involves *Deep Q-Networks (DQN) with parameter sharing*, where all agents share a single Q-network during training and deploy identical clones [8], [11]. The second category includes *Actor-Critic (AC) methods with full parameter sharing*, applied to dynamic channel access and power control [10]. Finally, the *Independent Actor with Global Critic (IAGC) paradigm* allows each agent to maintain an independent actor while sharing a centralized critic, [9], [12]. The latter structure has been shown to offer greater flexibility in modeling heterogeneous agents while leveraging a shared critic for more robust value estimation.

b) Neural Network Pruning: With the growing adoption of IoT and edge computing, there is an increasing demand for running complex neural networks on memory-constrained embedded devices. To address this challenge, techniques such as neural network pruning [13] have gained significant attention, as they reduce model size without substantial performance degradation. In the single-agent DRL domain, it has been well established that state-of-the-art algorithms often lead to agents underutilizing their parameters [14]. Studies have shown that carefully designed dynamic pruning strategies can remove a

significant portion of redundant weights/connections, while maintaining [15], or even improving [16], [17], the agent's performance. In [16], *gradual magnitude pruning* was presented as a general technique for improving performance in value-based DRL. However, these works primarily focus on single-agent environments where full or near-complete observability is assumed. Hence, pruning in multi-agent problems remains underexplored, especially for recurrent policies needed in partially observable environments. A Pruning at Initialization (PaI) framework [18] was introduced for MADRL, but it only prunes feed-forward agent networks, which are typically not appropriate for modeling temporal dependencies, as usually appearing in DSA problems [8], [9].

c) This Paper's Contribution: Motivated by the recent findings of [16], [17] we present, in this paper, a novel multi-agent actor pruning framework for distributed DSA. Each actor network is iteratively pruned using magnitude-based sparsification, guided by a pruning scheduler. Our numerical results showcase that our method not only preserves performance, but can even surpass dense IAGC-trained agents, reinforcing the role of pruning as a *general performance-enhancing technique* in MADRL. In addition, we introduce a novel *harmonic annealing* scheduler that enables periodic network weight regrowth to counteract excessive sparsification. This scheduler achieves comparable or superior performance at high sparsities compared to conventional linear and polynomial pruning schedules. Notably, it also has the potential to discover significantly better policies under favorable training conditions, highlighting the impact of dynamic sparsity in MARL training and policy optimization.

II. System model

Consider N users sharing K orthogonal channels (i.e., subbands). Each user (agent) n, at time t, can access one of the channels or even remain inactive (action a_t^n). The channel indices are $1, 2, \ldots, K$ and an inactive action is represented with $a_t^n = 0$. If a collision occurs (which means that $a_t^n = a_t^m$ for some n, m), neither of the users accesses that channel [8]. Note that actions are taken in a fully decentralized/independent manner, and each user cannot know which channels the other users will select. Similarly to [8], we assume that all users are constantly backlogged and can transmit on discrete time slots (i.e., Aloha-type narrowband transmissions).

The received signal after attempting channel access will be $o_t^n \in \{\text{ACK}, \text{NACK}\}$, depending on whether a collision occurs. We will use 1 to denote ACK, and -1 to denote NACK. If a user n successfully accesses the k-th channel, transmission happens with an SNR of β_k^n. The SNR values are assumed to be time-invariant, but unknown [9]. The cumulative SNR of user n for a known and finite horizon T is defined as:

$$\text{SNR}^n(T) \triangleq \sum_{t=1}^{T} \beta^n(t), \tag{1}$$

where $\beta^n(t) \triangleq \beta_{a_t^n}^n \mathbb{1}\{o_t^n = 1\}$ with $\mathbb{1}\{\cdot\}$ being the indicator function of a set/action. Note that the channels are not necessarily identical, since certain users may get higher SNR values on some channels depending on factors, such as the locations of the communication nodes and their equipment. In addition, the users are unaware of the access attempts of their peers, hence, they must learn their access patterns, while simultaneously identifying the most favorable channels based on their past observations and rewards.

Let g^n represent the action selection policy of user n, and let $\mathrm{g} \triangleq (g^1, \ldots, g^N)$. In this paper, we focus on solving the following *throughput* maximization problem:

$$\mathcal{OP}: \quad \max_{\mathrm{g}} E\left\{\sum_{t=1}^{T}\sum_{n=1}^{N} \log_2\left(1+\beta^n(t)\right)\right\},$$

where the expectation operation treats the possible stochasticity of the collection of policies g. Users in this system may include lightweight devices such as IoT sensors, wearables, or smart home appliances, which often face limitations in memory, computation, and energy. These constraints make it impractical for each device to run large or resource-intensive models to optimize action selection. To address this, we apply neural network pruning, removing redundant weights from a dense model to create a sparse version with significantly reduced memory requirements.

The model presented above can be easily extended to account for environments with Primary Users (PUs), leading naturally to the licensed shared access paradigm [19], [20]. Each PU can be actually modeled as an external process (e.g., [3], [5], [8]), and when a secondary user (agent) n attempts to access an occupied channel k, it receives an observation indicating PU activity. We denote channel occupancy by a PU with an observation value equal to -2. A channel k is occupied by a PU with probability π_k, in which case all access attempts will result in observations of -2. Otherwise, the previously defined observation and SNR rules apply.

III. MARL with Harmonic Annealing Pruning

In this section, we present our multi-agent algorithm comprising multiple training iterations; let their number be I_{T}. Each i-th iteration ($i = 1, \ldots, I_{\mathrm{T}}$) includes three steps. The first step involves sampling of trajectory data, whereas, in the second step, we use these trajectories to update the neural networks via Proximal Policy Optimization (PPO) [21]. Once the updates are completed, we prune the resulting actor networks during the final third step.

a) Neural Network Structures: We consider N individual recurrent actor networks $\theta^1, \theta^2, \ldots, \theta^N$. The input to both the actor and critic networks is the latest action-observation pair, which we represent as:

$$\boldsymbol{y}_t^n \triangleq [a_{t-1}^n, o_{t-1}^n]. \tag{2}$$

All actors as well as the global critic ϕ are Long Short-Term Memory (LSTM) [22] networks, whose final output is passed through a linear layer. For the actors, the linear layer's output passes through a softmax activation function to generate the action distributions. Each of these distributions is denoted as:

$$g_t^{\theta^n} = g(a_t^n | \boldsymbol{y}_t^n, \boldsymbol{h}_{t-1}^n; \theta^n), \tag{3}$$

with $g(\cdot)$ indicating each action's distribution dependence on the input parameters, and $\boldsymbol{h}_t^n$ being the recurrent network's hidden state vector. The global critic estimates the value for a given state. Therefore, its output $V(\boldsymbol{y}_t^n|\boldsymbol{h}_{t-1}^{n,c};\phi)$ is a scalar, with $\boldsymbol{h}_t^{n,c}$ denoting its hidden state.

b) Step 1: Trajectory Data Generation: We have designed a simulator generating episodes that describe the DSA system presented in Section II. At the beginning of each episode, all agents receive an initial dummy observation: $o_0^n = 0\ \forall n$. The hidden states of both the actor and critic networks are initialized to zero. At each time step $t = 1, 2, \ldots, T$, all agents select actions a_t^n, and the simulator computes the resulting collisions among users, as well as the individual SNRs $\beta^n(t)$. The instantaneous reward provided to each n-th agent was the system throughput[1], given as follows: $r_t^n = \sum_{n=1}^{N} \log_2\left(1 + \beta^n(t)\right)$. To increase training stability, we have performed reward normalization.

c) Step 2: Neural Networks Updating: By using multiple trajectories from the first step, we optimize the actors and the global critic with the clipped version of the PPO algorithm. We adopt the *bootstrapped sequential updates* paradigm, introduced in [23], where updates are performed by initializing the hidden states at the beginning of each sampled trajectory, and sequentially updating the network with the current losses at each t, until the episode concludes. For a given trajectory and time instant t, the actor and critic gradient update rules (performed through the Adam optimizer [24]) are given by the following expressions:

$$\theta^n \leftarrow \theta^n + \alpha_n \nabla_\theta L^{\text{clip}}(\boldsymbol{y}_t^n, a_t^n, \theta_{i-1}^n, \theta^n), \tag{4}$$

$$\phi \leftarrow \phi - \alpha_c \nabla_\phi L_{\text{critic}}^n(\boldsymbol{y}_t^n|\boldsymbol{h}_{t-1}^{n,c};\phi), \tag{5}$$

where α_n and α_c represent learning rates, and the loses $L^{\text{clip}}(\cdot)$ and $L_{\text{critic}}^n(\cdot)$ were designed as follows.

Actor Loss: Let us first define the clipping function as follows:

$$\text{clip}(\epsilon, A) = \begin{cases} (1+\epsilon)A, & A \geq 0 \\ (1-\epsilon)A, & A \leq 0 \end{cases}, \tag{6}$$

with ϵ being a manually selected parameter. The clip objective is then defined as

$$L^{\text{clip}}(\boldsymbol{y}_t^n, a_t^n, \theta_{i-1}^n, \theta^n) \triangleq \min\left(\frac{g(a_t^n|\boldsymbol{y}_t^n, \boldsymbol{h}_{t-1}^n;\theta^n)}{g(a_t^n|\boldsymbol{y}_t^n, \boldsymbol{h}_{t-1}^n;\theta_{i-1}^n)} A^{g^{\theta_{i-1}^n}}(\boldsymbol{y}_t^n, a_t^n), \text{clip}(\epsilon, A^{g^{\theta_{i-1}^n}}(\boldsymbol{y}_t^n, a_t^n))\right). \tag{7}$$

The advantage function $A^{g^{\theta_{i-1}^n}}(\boldsymbol{y}_t^n, a_t^n)$ under policy $g^{\theta_{i-1}^n}$ in (7) provides a way to compare the relative benefit of actions in a given state. Let $Q^{g^{\theta_{i-1}^n}}(\boldsymbol{y}_t^n, a_t^n)$ denote the action-value (often referred to as Q-value) function, which represents the expected cumulative reward after taking action a_t^n at state $\boldsymbol{y}_t^n$, and then following $g^{\theta_{i-1}^n}$. The latter function is defined as:

$$A^{g^{\theta_{i-1}^n}}(\boldsymbol{y}_t^n, a_t^n) \triangleq Q^{g^{\theta_{i-1}^n}}(\boldsymbol{y}_t^n, a_t^n) - V^{g^{\theta_{i-1}^n}}(\boldsymbol{y}_t^n), \tag{8}$$

which can be computed using [25]'s generalized estimator.

Critic Loss: For each n-th agent, the global critic's loss at each time t is a typical least squares error function:

$$L_{\text{critic}}^n(\mathbf{y}_t^n|\mathbf{h}_{t-1}^{n,c};\phi) \triangleq \left(V(\boldsymbol{y}_t^n|\mathbf{h}_{t-1}^{n,c};\phi) - \hat{R}_t^n\right)^2, \tag{9}$$

where $\hat{R}_t^n$ denotes the reward-to-go at time t.

d) Step 3: Neural Networks Pruning: In this step, we apply *magnitude-based unstructured pruning* to the actor networks of step 2, by setting some of their weights to zero, achieving a sparsity level of p_i^n. This sparsity level can be determined by the gradual pruning rule [13]:

$$p_i^n = \begin{cases} 0 & \text{if} \quad i < i_{\text{start}} \\ p_{\text{final}}^n \sigma^n(i) & \text{if} \quad i_{\text{start}} \leq i < i_P, \end{cases}, \tag{10}$$

where i_P denotes the total number of pruning iterations, i_{start} is the training iteration when pruning begins, $\sigma^n(\cdot)$ represents the pruning schedule, and p_{final}^n is the target sparsity level for n-th agent; clearly, we allow different sparsity requirements for each agent. Some agents may operate on lightweight, resource-constrained devices, necessitating smaller networks, whereas others with access to greater computational resources can accommodate larger models.

Two typical choices for scheduling sparsity levels are the linear and the third-degree polynomial schedulers [13], [17]:

$$\sigma_{\text{linear}}^n(i) \triangleq \frac{i - i_{\text{start}}}{i_P - i_{\text{start}}},\ \sigma_{\text{poly}}^n(i) \triangleq 1 - \left(1 - \frac{i - i_{\text{start}}}{i_P - i_{\text{start}}}\right)^3. \tag{11}$$

In these schedulers, sparsity monotonically increases at each i-th iteration. This implies that, once critical weights are pruned (i.e., set to zero), they cannot be recovered. This fact can potentially break crucial network pathways, thus, limiting the network's capacity for exploration, restricting its ability to adapt and refine policies over time. To deal with this limitation, in this paper, we propose the following novel, *harmonic annealing* pruning scheduler, which selects sparsity levels as:

$$p_i^n = \min\Big\{\max\Big(b_i^n + c_i^n,\, p_0^n\Big),\, p_{\text{final}}^n\Big\}, \tag{12}$$

where

$$c_i^n = 0.1\, \sin\Big(\frac{2\pi\, i}{200}\Big), \tag{13}$$

$$b_i^n = p_{\text{final}}^n + \frac{1}{2}\Big(p_0^n - p_{\text{final}}^n\Big)\left(1 + \cos\Big(\pi\, \frac{i - i_{\text{start}}}{i_P - i_{\text{start}}}\Big)\right). \tag{14}$$

In this definition, the parameter b_i^n follows a cosine annealing function that ensures smooth and gradual pruning, while c_i^n introduces a periodic oscillation to allow periodic weight regrowth during training. Finally, the parameter p_0^n represents the initial sparsity level of each n-th agent.

The average sparsity level across agents with the proposed harmonic annealing scheduler for $I_{\text{T}} = 1000$ and $p_{\text{final}} = 0.95$

[1]Our algorithm is purely model-free, and hence, alternative rewards suitable for optimizing other objectives can be incorporated.

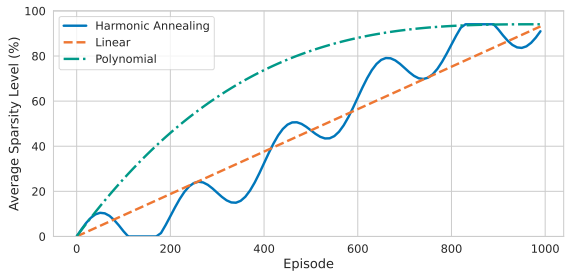


Fig. 1: The average sparsity level for the three different considered pruning schedulers for $I_{\mathrm{T}} = 1000$ and $p_{\mathrm{final}} = 0.95$.

is illustrated in Fig. 1, together with the schedulers in (11). It can be seen that our scheduler introduces periodic weight regrowth through its oscillatory nature, allowing previously pruned connections to recover. As it will be showcased in the following section, our dynamic prune-and-regrow mechanism, not only stabilizes training, acting as an implicit regularization mechanism, but also expands the search space, increasing the potential to uncover superior policies.

Remark 1 (Pruning Critic Networks): During deployment, only the actor networks are running at the agents, allowing the critic networks to be large, if this benefits the training process. Since the computational cost of large critics is incurred only during training, we restrict pruning to the actor networks. Moreover, in the single-agent setting, it has been observed that estimating value functions is more challenging than learning policies, and pruning critics often degrades performance [17].

Remark 2 (Pruning Interval): Rather than pruning after each i-th iteration, we have noticed that allowing networks to regrow for a sufficient period, and then pruning every i_{prune} iterations can lead to a noticeable performance increase.

IV. Numerical Results and Discussion

a) Neural Networks Implementation Details: We have used the PyTorch framework [26]. The actors were modeled as unidirectional LSTMs with two hidden layers of 128 units, using Rectified Linear Units (ReLU) activations. The critic shared the same architecture, except for its final output layer. We have set the learning rates of the actors to $\alpha_n = 10^{-4}$ and those of the critic to $\alpha_c = 5 \times 10^{-5}$. The PPO's clip parameter was set to $\epsilon = 0.2$ and the discount factor to 0.99. The target sparsity levels of the agents were set as $p^n_{\mathrm{final}} = 0.95 \ \forall n = 1, \ldots, N$. Each learning algorithm was run for $I_{\mathrm{T}} = 1000$ episodes with a horizon of $T = 100$ time steps, and the results were obtained via averaging over 10 different random seeds. Pruning was applied every $i_{\mathrm{prune}} = 5$ iterations. For both the linear and harmonic annealing schedulers, we set $i_{\mathrm{start}} = 0$, while for the polynomial scheduler, we performed experiments with $i_{\mathrm{start}} \in \{0, 200\}$. Further investigations on the pruning hyperparameters as well as sensitivity studies will be provided in the journal version of this work.

b) Benchmark Schemes: In terms of DSA benchmarks, we have considered two learning-based ones: a recurrent DQN with full parameter sharing between the agents [8], and a recurrent actor-critic with individual actors and a global critic (IAGC), as in [9], [12], as well as a randomized slotted aloha scheme. We have also considered an Opportunistic Channel-Aware (OCA) policy. For fair comparison, all learning-based methods used identical network architectures.

In addition, to evaluate the effectiveness of our proposed pruning framework, we compare it with the *PaI with Parameter Sharing* baseline [18]. In this approach, all actor networks were initialized with identical parameters, but each agent was assigned a unique pruning mask. The agents were then trained using PPO, with only the remaining parameters being shared. The target sparsity was set to 50%, which is lower than that of our proposed method, since we observed that, at higher sparsity levels, PaI's performance deteriorated significantly, approaching that of a random policy.

c) Performance Evaluation: We have simulated a DSA system comprising $N = 10$ secondary users (agents) and $K = N/2$ orthogonal channels. Studies with larger N values will be presented in a future journal version of this work. The SNR values were sampled as $\beta^n_k \sim \mathcal{U}[30, 40]$ at the start of each episode, with a communication horizon of $T = 100$. We have considered two different system setups: (A) without PUs; and (B) each channel k was occupied by a PU with probability $\pi_k = 0.2$, independently of the other channels.

At every 10 episodes, the networks training was paused for evaluation over 100 episodes. Figures 1 and 2 include respectively the sparsity levels and training curves of all considered schedulers, illustrating both the mean episodic reward ($\frac{1}{N}\sum_{t=1}^{T}\sum_{n=1}^{N} r^n_t$) and the reward's standard deviation across multiple seeds. Additionally, Table I reports the final rewards achieved by the best performing seed for each studied method. The key takeaways from our results are as follows:

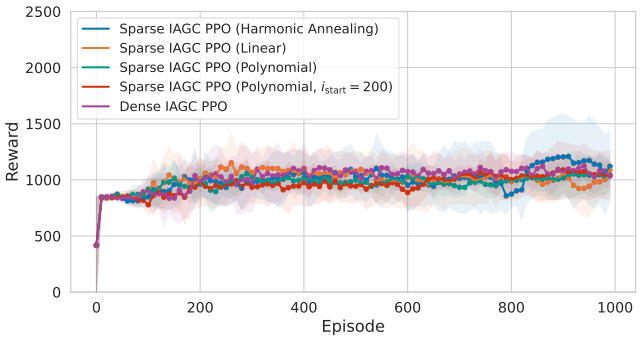
Comparison with DSA benchmarks: Both dense (i.e., unpruned) IAGC-PPO and all our sparse IAGC PPO (95%) versions with different pruning scheduling outperform the considered DSA benchmarks. In setup (A), all methods achieve an average reward exceeding 1000, while in Setup (B), all of surpass 800 after training convergence.

Comparison with PaI pruning baseline: It can be observed that all three variants of our algorithm outperform PaI in terms of reward, despite achieving higher sparsities.

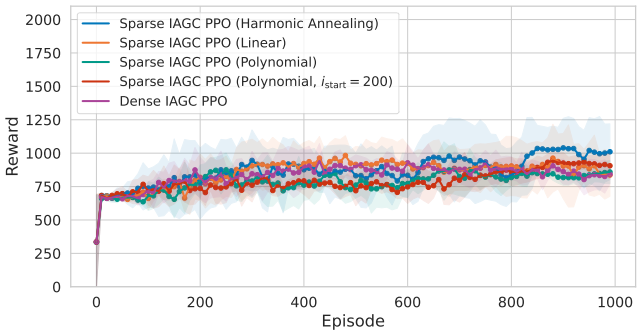
Comparison among pruning schedulers: As shown, all pruning schedules achieve *highly sparse networks* (i.e., $\geq 90\%$ sparsity) with minimal performance degradation. Notably, our harmonic scheduler exhibits superior average performance at high sparsities, and is particularly effective at *discovering the best policies under favorable seeds*.

V. Conclusion and Future Work

In this paper, we presented a novel multi-agent gradual pruning approach based on the IAGC framework and applied it to the distributed DSA problem. In addition to existing pruning schedulers, we proposed a harmonic annealing scheduler that



(a) Setup A.



(b) Setup B.

Fig. 2: Training curves for the proposed MADRL framework for all considered pruning schedulers, including the standard deviation across multiple random seeds (shaded region).

TABLE I: Reward evaluation for the best-case seed.

Algorithms	**Setup A**	**Setup B**
Slotted Aloha	993.77	764.10
OCA	1023.45	811.52
IAGC [9], [12]	996.11	802.24
DQN with parameter sharing [8]	742.54	555.41
PaI [18] (50%)	1111.88	825.34
Dense IAGC PPO	1556.83	1115.74
Proposed IAGC PPO (95%) with scheduler:		
Harmonic annealing	**1826.86**	**1223.41**
Linear	1194.82	1083.76
Polynomial	1412.89	1092.19
Polynomial ($i_{\text{start}} = 200$)	1496.46	1048.51

enables periodic weight regrowth. Our experiments demonstrated that our framework outperforms existing MADRL pruning schemes as well as DSA benchmarks. Notably, harmonic pruning not only achieves strong performance, but also unveils optimal policies missed by other pruning schedulers. Future work includes exploring alternative objective functions and extensions of our approach to other applications, such as games and transmit power control [10]. Additionally, we plan to investigate meta-learning techniques to bias training towards promising initialization conditions. Finally, the effectiveness of our harmonic annealing scheduler will be investigated on more traditional supervised and unsupervised tasks, such as image classification and autoencoders.

REFERENCES

[1] F. Hu *et al.*, "Full spectrum sharing in cognitive radio networks toward 5G: A survey," *IEEE Access*, vol. 6, pp. 15754–15776, 2018.

[2] S. H. A. Ahmad *et al.*, "Optimality of myopic sensing in multichannel opportunistic access," *IEEE Trans. Inf. Theory*, vol. 55, no. 9, pp. 4040–4050, 2009.

[3] Q. Zhao *et al.*, "Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 3, pp. 589–600, 2007.

[4] K.-L. A. Yau *et al.*, "Enhancing network performance in distributed cognitive radio networks using single-agent and multi-agent reinforcement learning," in *Proc. IEEE LCN*, (Denver, USA), 2010.

[5] K. Cohen *et al.*, "Restless multi-armed bandits under time-varying activation constraints for dynamic spectrum access," in *Proc. Asilomar*, (Pacific Grove, California, USA), 2014.

[6] S. Wang *et al.*, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cognitive Commun. Network.*, vol. 4, no. 2, pp. 257–265, 2018.

[7] L. Kraemer and B. Banerjee, "Multi-agent reinforcement learning as a rehearsal for decentralized planning," *Neurocomputing*, vol. 190, pp. 82–94, 2016.

[8] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for distributed dynamic spectrum access," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 310–323, 2019.

[9] L. Dong *et al.*, "Dynamic spectrum access and sharing through actor-critic deep reinforcement learning," *EURASIP J Wireless Com Network*, vol. 48, 22.

[10] Z. Lu *et al.*, "Dynamic channel access and power control in wireless interference networks via multi-agent deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 71, no. 2, pp. 1588–1601, 2022.

[11] V. Mnih *et al.*, "Human-level control through deep reinforcement learning.," *Nature*, vol. 518, p. 529–533, 2015.

[12] R. Lowe *et al.*, "Multi-agent actor-critic for mixed cooperative-competitive environments," *arXiv preprint: 1706.02275*, 2020.

[13] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint: 1710.01878*, 2017.

[14] A. Kumar *et al.*, "Implicit under-parameterization inhibits data-efficient deep reinforcement learning," *Proc. ICLR*, 2021, Vienna, Austria.

[15] D. Livne and K. Cohen, "PoPS: Policy pruning and shrinking for deep reinforcement learning," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 789–801, 2020.

[16] J. Obando-Ceron *et al.*, "In value-based deep reinforcement learning, a pruned network is a good network," *Proc. ICML*, 2024, Vienna, Austria.

[17] L. Graesser *et al.*, "The state of sparse training in deep reinforcement learning," *Proc. ICML*, 2022, Baltimore, USA.

[18] W. Kim and Y. Sung, "Parameter sharing with network pruning for scalable multi-agent deep reinforcement learning," *arXiv preprint: 2303.00912*, 2023.

[19] N. Taramas *et al.*, "Opportunistic beamforming for secondary users in licensed shared access networks," in *Proc. IEEE ISCCSP*, (Athens, Greece), 2014.

[20] S. Vassilaras and G. C. Alexandropoulos, "Optimizing access mechanisms for QoS provisioning in hardware constrained dynamic spectrum access," in *Proc. IEEE SPAWC*, (Edinburgh, UK), 2016.

[21] J. Schulman *et al.*, "Proximal policy optimization algorithms," *arXiv preprint: 1707.06347*, 2017.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997.

[23] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," *proc. AAAI Fall Symposium, (Austin, USA)*, 2015.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint: 1412.6980*, 2017.

[25] J. Schulman *et al.*, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint: 1506.02438*, 2018.

[26] A. Paszke *et al.*, "Automatic differentiation in PyTorch," *Proc. NeurIPS*, 2017, Long Beach, California, USA.