A Sample Efficient Multi-Agent Approach to Continuous Reinforcement Learning

Diarmuid Corcoran
Ericsson AB and Software and Computer Systems
KTH Royal Institute of Technology
Stockholm, Sweden
diarmuid.corcoran@ericsson.com

Per Kreuger
RISE AI
Research Institutes of Sweden
Kista, Sweden
per.kreuger@ri.se

Magnus Boman
Software and Computer Systems
KTH Royal Institute of Technology
Stockholm, Sweden
mab@kth.se

***Abstract*—As design, deployment and operation complexity increase in mobile systems, adaptive self-learning techniques have become essential enablers in mitigation and control of the complexity problem. Artificial intelligence and, in particular, reinforcement learning has shown great potential in learning complex tasks through observations. The majority of ongoing reinforcement learning research activities focus on single-agent problem settings with an assumption of accessibility to a globally observable state and action space. In many real-world settings, such as LTE or 5G, decision making is distributed and there is often only local accessibility to the state space. In such settings, multi-agent learning may be preferable, with the added challenge of ensuring that all agents collaboratively work towards achieving a common goal. We present a novel cooperative and distributed actor-critic multi-agent reinforcement learning algorithm. We claim the approach is sample efficient, both in terms of selecting observation samples and in terms of assignment of credit between subsets of collaborating agents.**

***Index Terms*—Machine learning, Radio resource scheduling**

I. Introduction

The LTE and 5G Radio Access Networks (RAN) [1] are complex distributed ecosystems of HW, SW and radio spectrum resources. Configuring, operating and coordinating these systems is a costly, error-prone and labor-intensive task. One very promising approach is the augmentation of existing automation approaches like SON (Self Organizing Networks) [2] with state-of-the-art self-learning machine learning techniques. Reinforcement learning (RL) [3] is a branch of machine learning that has shown considerable potential in learning sophisticated control and decision-making strategies, often called policies, in highly stochastic environments. In previous work [4], we demonstrated the potential of RL to learn complex radio resource management scheduling patterns. One assumption of that method is a centralized learning approach, using a combination of simulator (or digital twin [5]) and real-world system deployment, where the learning agent has access to a rich observation space. While that approach is certainly viable for coordinated deployments using a common scheduler, many real-world resource management mechanisms are naturally modeled using a decentralized and distributed control approach [6]. We refer to the approach taken in [4] as a single agent RL (SARL) solution, in which a single policy is trained, using a set of distributed resources, by observing a trajectory of rewards selected from a distribution of actions conditioned on state. In contrast, in a multi-agent RL (MARL) system, a set of cooperating agents train independent policies while simultaneously ensuring that these agents act towards maximizing some chosen common goal or intent.

In this work, we extend a concept of continuous SARL, as described in [4], to learn effective and scalable MARL solutions to the SON problem of Inter-Cell Interference Coordination (ICIC) [7]. The MARL approach is particularly suited to this problem domain for a number of reasons. First, radio networks are in their nature highly distributed compute systems and while centralized decisions is very desirable on a local cluster of resources, it is often not feasible on a large scale [8]. Second, centralizing data observations for timely decision-making on a sub-second timescale is problematic in terms of added latency and transport cost. Finally, using a distributed MARL approach reduces compute requirements for continuous training as each agent works on its local, reduced, state space. This has the advantage of reducing the deep-learning complexity requirements per agent (cf. [9]) so that standard processors, instead of expensive and power hungry GPUs, may be sufficient. Technical solutions for goal coordination across multiple distributed agents (sometimes called the credit assignment problem [10]) in an RL setting is a particularly challenging area, with several possible approaches (see section III). In this paper, we focus on an adaptive partial reward sharing between groups of agents to guide a common intent. The key contributions to the MARL field as proposed in this work are as follows: 1) A mechanism to, adaptively, identify which agents are most likely to have dependent actions. 2) A sample efficient partial reward sharing strategy across dependent agents. 3) A novel distributed RL algorithm, with extension to the actor-critic policy-based approach. 4) Experimental comparisons of SARL and MARL algorithms based on previous baselines.

II. Distributed Reinforcement Learning Systems

1) Background: Reinforcement learning [3] is a way to learn a policy to select actions that optimize a selected goal for a range of system states. Agents interact with an environment with observable state s, through actions a. The environment

978-3-903176-51-5 © 2022 IFIP

disburse rewards r to the agent, based on the value of actions a in state s. The environment also transitions into a new observable state s'. The policy $\pi(a|s)$ is the probability of choosing the action a, given state s. The objective of an RL agent is to learn a policy that selects the best a in any given state s and a specific optimization objective. Formally, an RL problem setting can be described as maximizing an objective J, a discounted (γ) reward expectation over a sequence of time-steps (t), in following a trajectory (τ) as selected by policy π:

$$\max \left(\left| J(\pi) = \mathbb{E}_{\tau \sim \pi}[R(\tau)] = \mathbb{E}_{\tau} \left[\sum_{t=0}^{T} \gamma^t r_t \right] \right| \right) \quad (1)$$

In many modern RL settings, the policy π is parameterized by a set of parameters θ, implemented as a neural network [11], where θ represents the neural network weights. In a SARL setting, the objective then becomes the tuning of θ in the objective $\max_\theta J(\pi_\theta)$, with parameter update following the objective gradient: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\pi_\theta)$. There are many methods to achieve this objective [3], but for most large scale real-world problems there are currently two main approaches: 1) Value-based methods based on temporal difference learning (TD, SARSA, Q-learning) and Bellman optimality [12]. 2) Policy-based methods that directly learn an optimal (or approximate) policy, using the policy gradient theorem [13], [14]. In policy-based approaches, the objective gradient [14] can be derived using equation 2:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} A_t^\pi \nabla_\theta \log \pi_\theta (a_t \mid s_t) \right] \quad (2)$$

where A_t^π is an encoding of reward called the advantage function, which in the simplest case can be the Monte-Carlo reward return from trajectory episodes. Another possible encoding of A_t^π is an estimate of the current state-action value for each time-step, t, measured against the action-value: $A^\pi(s_t a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$. This approach leads to a rich family of actor-critic algorithms. In this work, we adapt actor-critic architectures for use in a distributed and multi-agent setting.

2) Multi-Agent RL: In a cooperative MARL setting, a set of agents $g \in \{1, \ldots, N\}$ collaborate towards a common objective J. In the most general case, each agent g follows an independent policy π^g and the objective $\max_{\bar{\theta}} J(\pi_{\bar{\theta}})$, where $\bar{\theta} = \{\theta_1, \ldots, \theta_N\}$ is the set of policies followed by each agent. The gradient of the multi-agent objective can then be described by adapting equation 2 to incorporate a set of collaborating agents. Equation 3

$$\nabla_{\bar{\theta}} J(\pi_{\bar{\theta}}) = \mathbb{E}_{\tau \sim \pi_{\bar{\theta}}} \left[\sum_{t=0}^{T} \sum_{r_g}^{g \in I_g} A_t^{\pi^g} \nabla_{\bar{\theta}} \log \pi_{\bar{\theta}} (a_t^g \mid s_t^g) \right] \quad (3)$$

formalizes our approach, where each agent first aggregates partial rewards from a collaborating set of agents $\mathcal{I}_g$, then performs a standard sum across time-steps. Since each agent g can be present in several collaboration sets $\mathcal{I}_g$, the rewards dependency impact on $\bar{\theta}$ propagates through the system as described in section V.

III. Related Work

Currently, there are two main approaches to tackling RL multi-agent systems. In the first, independent learning (IL), each agent acts independently, using Q-learning or policy-based algorithms, and only perceive each other as part of the environment. A second approach is to use centralized training, with a richer observation of the environment, and then decentralized execution (CTDE). Within the CTDE category, MADDPG [9], a multi-agent adaption of the standard DDPG policy algorithm, trains a decentralized actor but centralized critic by observing a joint state-action space. Another approach, COMA [15], trains a counterfactual multi-agent policy using an adaptation to the actor advantage estimation. Again, COMA needs to observe the joint agent state-action space during training. Value decomposition networks [16] (VDN) and QMIX [17], an extension to VDN, are value-based approaches where each agent Q-value network shares centralized gradient information during training. Compared against the relative complexity of CTDE, IL approaches are simple and in some cases can be effective and competitive against CTDE [18]. From [18] it seems clear that in those cases where IL underperforms, the cause is failure to infer an appropriate credit assignment between collaborating agents in a non-stationary environment. In this work, we propose an approach that combines the simplicity of IL with a partial reward sharing mechanism between a select subset of collaborating agents. In contrast to CTDE it does not assume, but can benefit from, central simulator-based training. We make no assumptions of access to a complete action or state space during training or execution. The method is especially suited to continuous learning in networking scenarios where access to both state and action space is limited but communication channels between independent agents are readily available.

IV. System Overview

A. Problem Description

The LTE and the 5G radio system support Orthogonal Frequency Division Multiple Access (OFDMA) [19] where all sectors share the same set of frequency resources to allow higher spectral efficiency. This flexibility can lead to high levels of interference between allocated radio resources at cell boundaries and thus degraded quality of service for users in such high-interference zones. The problem of interfering cells in a RAN system is particularly challenging with the deployment of 5G and increasingly dense small cell deployments, heterogeneous networks (HetNet) and *ad hoc* configurations [20]. Fig. 1 illustrates two such scenarios with significant user distributions at cell edges, and a dense hot-spot (Fig. 1, inset) with users clustered at intersecting cell boundaries. With this challenge in mind, detailed manual cell planning becomes difficult and the need for self-organizing approaches ever more important. In LTE and 5G, one of the most basic units of end-user resource allocation is a physical resource block (PRB). A PRB is a contiguous unit of spectrum (Hz) and time (ms or sub-ms), depending on the particular

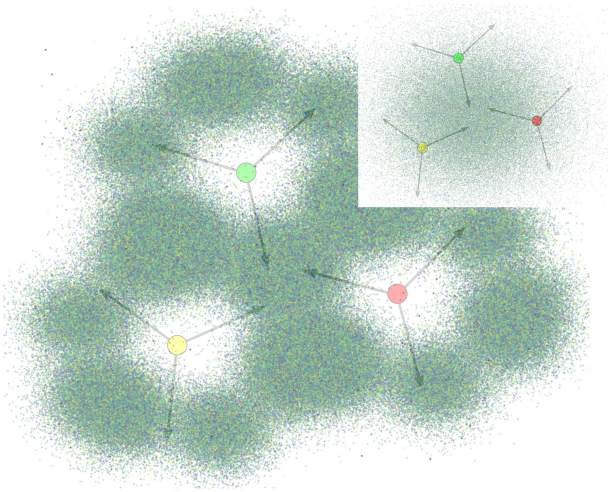

Fig. 1. Cell Edge Distribution and centered hot-spot distribution (inset)

radio standard and generation [19]. In dense configurations, at cell edges, there can be high levels of interference between transmitters when PRBs are allocated in an uncoordinated fashion. In self-learning approaches, the advantage of a MARL solution over SARL lies in the fact that it more closely aligns to both the hardware and software implementation architecture of a RAN with distributed computation and data. As such, we can utilize existing RAN computation nodes, without the need for expensive, centralized, data processing infrastructure. In section V-A2, we describe a distributed and sample efficient actor-critic RL algorithm which we call DAC (distributed actor-critic).

B. System Model

The system model has been described in detail in a previous paper [4] that focused on a single-agent solution, but we provide a condensed description here for the sake of completeness. It consists of two principal parts. First, a static part which is configured and created at the start of each training episode, described by a configuration tuple $\mathcal{S}$. Second, a dynamic part which evolves by applying system scheduling decisions a to an environment configured by $\mathcal{S}$ at each time-step within a training episode. The model consists of a set of cells $\mathcal{C}$, beams $\mathcal{B}$, and available PRBs $\mathcal{R}$ per beam. A beam is associated with a cell and all PRBs in $\mathcal{R}$ are available in each beam. Each beam has transmit power $\mathcal{W}$; gain pattern $\mathcal{G}$ used for fading calculations by a propagation model $\mathcal{P}$. A set of users $\mathcal{U}$ is applied to represent service demand and these can be fixed or assigned according to a mobility model, as described in section VI.

From $\mathcal{S}$, we obtain a received signal power (RSP) for every user by applying $\mathcal{P}$. PRBs are disabled within beams by decreasing the transmit powers by a fixed ratio, according to a PRB schedule a, and subsequently applying a stochastic fast fading model $\mathcal{D}$ separately over individual PRB in $\mathcal{R}$, yielding a set of unique RSP samples per PRB. For each PRB, approximate signal-to-noise-and-interference-ratios (SINR) are calculated by summing over $\mathcal{R} \times \mathcal{B} \times \mathcal{U}$. In our RL setting, we designate the resulting SINRs the *observable* states of the modeled system. As a final step, a radio resource management scheduling [21] algorithm $\mathcal{M}$ is applied to all active PRB, as determined by a. Different targets (i.e. performance or fairness) can be applied in $\mathcal{M}$ and the specifics of how to estimate RL rewards for these are described in section V-B. This chain of operations, is applied for each schedule TTI (Time Transmission Interval) and results in a *realized* spectral efficiency e, and updated system state of which the observable part is s. In our RL setting, we use three specific metrics of utilization: 1) System spectral efficiency (SSE) [22] in $b/s/Hz$, which in our current model has a maximum theoretical value of $\sim 4.3\, b/s/Hz$. 2) Fairness, a fractional measure of how fair the scheduling allocation is across all available users or devices $\mathcal{U}$, where 1 means completely fair. 3) PRB utilization (Ruse), a fractional measure of the PRBs needed to achieve SSE or fairness utilization, where 1 means all system PRBs are in use. Given the problem setting described in IV-A, the goal of our RL method is to create PRB activation schedules, a, which maximize SSE, or SSE and fairness, while at the same time minimizing system resources (Ruse) needed to achieve this. In a SARL setting, as described in [4], a single schedule a is generated and applied to the system model. In a MARL setting, as described herein, an agent per beam $\mathcal{B}$, generates independent schedules a, with a distributed reward assignment sharing scheme (section V) used to ensure multi-agent convergence towards a common objective.

V. Method

In our MARL setting, there are two key contributing components: 1) A method, as specified in algorithm 1, to select the user devices most likely to provide useful measurement samples for training and, subsequently, a mechanism to select cooperating agents. 2) A distributed MARL algorithm, as outlined in algorithm 2, that efficiently selects and distributes reward assignment among cooperating agents. The approach is sample efficient as it selects the training observation space with most potential from a much larger set, actively selects groups of cooperating agents per agent and allows for the possibility of down-sampling credit assignment information between cooperating agents. In this MARL setting, each agent follows an independent policy π, and interacts with the local environment through an action a, upon which the environment will update its internal local state and produce a local reward r and a new (local observable) state s'. Consistency with eqn. 3 would necessitate that these parameters be tagged g to indicate agent local scope, but to simplify notation, we shall assume this is implicit in our method description. Each independent agent policy $\pi(a|s)$, applied to s, is the probability of choosing the action a given s, and the objective of an MARL agent is to learn the best a given local state s for a specific optimization objective. As independently operating agents will not necessarily converge towards a common multi-agent operating goal, contributing reward or credit assignment needs to shared between agents.

Algorithm 1: Collaborating agent selection

Data:
·Set of devices U that can measure and report RSRP p_u, for device u, relative to one or more beams b where $u \in U$
·Set of measurement reports P_u of p_u^b for b observed by u
·Set of collaborating agents $g \in \mathcal{G}_N$ of size N controlling beam b
Result:
· $\mathcal{O}_g$, set of best devices u observed by g
· $\mathcal{I}_g$ set of agents which g, periodically, shares reward with, where, $\mathcal{I} \subseteq \mathcal{G}$

1 **async** *each agent* $g \in \mathcal{G}$
2 Using all P_u reported to g, build $\mathcal{O}_g$, a set of D best p_u^b observed
3 $\mathcal{O}_g \leftarrow \text{argmax}_D\,(P_u)$
4 **send-to** $\mathcal{O}_g$ *to all* $g \in \mathcal{G}$
5 **async** *each agent* $g \in \mathcal{G}$
6 Set of report $\mathcal{O}_i$ where $i \leftarrow 1..N-1$ and $g \neq i$ from collaborating agents
7 $L_g[N]$: an array of size N
8 **for** $i \leftarrow 1..N-1$ **do**
9 $L_g[i] \leftarrow |\mathcal{O}_g \cap \mathcal{O}_i|$
10 **end**
11 $\mathcal{I}_g \leftarrow \text{argmax}_B\,(L_g)$

A. Sample and Cooperating Agent Selection

1) Sample Selection: In many MARL settings, agents present in the system may not necessarily be equally dependent on chosen actions by all other agents. In these scenarios, distributing reward or credit assignment information across all agents may be resource and time-consuming. Algorithm 1 is a method to determine which agents are most likely to benefit from reward/credit assignment exchange transactions. A set of devices U can observe and report received signal reference power (RSRP), p_u^b, relative to a set of beams B_u. Each beam has a controlling agent g, with a total of N agents in the system. Using P_u, a set of reports of p_u^b, reported to each agent g, $\mathcal{O}_g$ the set of best D devices observed by g is created and distributed to all other agents in $\mathcal{G}_N$ (Lines 3 - 4). Each agent g, collects the sets of $\mathcal{O}_{i \neq g}$ from other available agents. A set of agents, L_g, that have maximum potential for reward conflict through independent action selection, is created through calculating the observation space intersection between a given agent observation space $\mathcal{O}_g$ and other, available, agent observation spaces $\mathcal{O}_i$ (Line 9), communicated through an implementation specific agent network. From L_g, a set of cooperating agents $\mathcal{I}_g$ of size B, is determined, and will subsequently share reward assignment information (Line 11).
2) Distributed MARL Algorithm: Algorithm 2 is composed of two asynchronously running threads of computation. The first (Line 5) uses a periodically updated observation space $\mathcal{O}_g$ and set of credit assignment cooperating agents, $\mathcal{I}_g$, for each agent instance g, to guide policy action selection and trajectory recording. Each agent, $g \in \mathcal{G}$, acts independently in action selection from its policy $\pi_\theta(s)$. All agents g then apply the sampled action to the system model in a time synchronous step (specific for the radio domain scheduling algorithm) to the system model (Line 10). The local agent actions a are Bernoulli vectors of size $|\mathcal{R}|$, each variable specifying if a resource block in $\mathcal{R}$ should be active or not. In the single agent case, the total policy action space is $2^{|\mathcal{R}|\times|\mathcal{B}|}$, while in the multi-agent case, it is reduced to $2^{|\mathcal{R}|}$. Each system model iteration produces an updated (local observable) state s and a realised system efficiency e_g per agent. The updated state space has shape $|\mathcal{R}|\times|\mathcal{O}_g|\times\left|\mathcal{I}_g^*\right|$, where $\mathcal{I}_g^*$ indicates inclusion of agent g, in addition to its collaborators. Realized system efficiency has shape $|\mathcal{R}| \times |\mathcal{O}_g|$ and is considered a partial reward p vector until aggregated in a later step. The agent trajectory τ_g is recorded for each time-step t. At coordinated system policy update events each agent g exchanges its partial reward vectors, τ_{g_p}, from τ_g with collaborating agents in $\mathcal{I}_g$. An optional down-sampling step Φ can be applied on the reward vector and is a way to trade transfer reward samples against learning accuracy (section V-D).

The second thread of computation for each agent (Line 18) aggregates partial reward assignment from all collaborating agents and updates the individual agent policy. Each agent policy π is parameterized by θ, implemented as a neural network [11]. A new aggregated reward vector p_g (Line 24) is constructed for each agent g by combining the trajectory partial reward components τ_{g_p} with partial rewards from collaborators. Optionally, down-sampling can be applied to τ_{g_p} and the action space log probability distribution (Line 26), if incoming partial rewards have been down-sampled before distribution. An effective agent reward from action a, including the impact on cooperating agents in $\mathcal{I}_g$ can constructed by applying a chosen target function (Line 25, as described in section V-B. Each agent objective function $\mathcal{J}(\pi_\theta)$ maximizes the expected reward r for collaborating agents, and the policy gradient theorem [14] ensures that the gradient of $\mathcal{J}(\pi_\theta)$ can be can be incrementally calculated across the (down-sampled) trajectory τ as done in Lines 28-31. Agent advantage function $A_g^{\pi_\theta}(s,a)$ encodes the estimation of target reward derived in Line 25, and we employ an actor-critic [23] architecture as described section V-C. The parameterized policy $\pi(a \mid s;\theta)$ is updated using the objective function gradient and a chosen learning rate through back-propagation (Line 33) [11], [24].

B. Reward Target Functions

Many possible reward functions can be applied to our system setting but in this work we consider two: 1) maximum throughput (MT), and 2) proportional fair (PF). MT attempts to maximize SSE across all resource blocks with no consideration to user fairness. Equation 4 shows how reward can be estimated for an MT goal. The aggregated reward vector p_g for agent g is summed and then averaged across the state space size observed by an agent. In the PF case we attempt to balance throughput against fairness. To achieve this we need to maintain running throughput averages, $T_\mathcal{O}$ over the state space observed by devices in $\mathcal{O}_g$. This vector is then exponentiated with a weight w, to guide fairness, and a larger w will improve fairness at the cost of throughput. The aggregated reward vector p_g is

Algorithm 2: Distributed RL Algorithm - DAC

Initialize for $g \in \mathcal{G}$:
  Weights of parameterized policy network θ
  Learning rate α; Initial state s_0; Trajectory: τ
  $(\mathcal{I}_g, \mathcal{O}_g) \leftarrow$ periodically from **algorithm 1**
async *each agent* $g \in \mathcal{G}$
  foreach *TTI:* $t \in T$ **do**
    $s \sim \mathcal{O}_g$
    Sample action $a \leftarrow \pi_\theta(s)$
    Apply system model time synchronously
    time-sync $\{s, p \leftarrow e_g\} \leftarrow \{\mathcal{P}, \mathcal{D}, \mathcal{S}(a), \mathcal{M}\}$
    Record trajectory: $\tau_g \leftarrow \{s, a, p\}_t$
    if *update policy event* **then**
      send-to *collaborating agents*
        ▷ optional down-sample across t
        $p \leftarrow \Phi\left(\tau_{g_p}\right)$
    end
  end
async *update policy event*
  $\nabla_\theta \mathcal{J}(\pi_\theta) \leftarrow 0$
  $R = \{\}$: List to collect partial rewards
  receive *partial rewards* p *from all* $g \in \mathcal{I}_g$
    R.append(p)
  ▷ merge partial rewards
  $p_g \leftarrow array(\Phi\left(\tau_{g_p}\right) \cdot R)$
  Derive reward $r \leftarrow$ Targ(p_g)
  $\mathcal{L} \leftarrow \Phi\left(\log \pi_\theta(a|s)\right)$
  foreach *step* i *in* τ **do**
    $\Delta\theta_i$ = -$\nabla_\theta \mathcal{L} A_g^{\pi_\theta}(s_i, a_i)$
    $\nabla_\theta \mathcal{J}(\pi_\theta) \leftarrow \nabla_\theta \mathcal{J}(\pi_\theta)$ + $\Delta\theta_i$
  end
  $\nabla_\theta \mathcal{J}(\pi_\theta) \leftarrow mean(\nabla_\theta \mathcal{J}(\pi_\theta))$
  ▷ Update policy parameters through back-propagation and reset τ
  $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{J}(\pi_\theta)$
  Reset τ

then normalized with $T_{\mathcal{O}}^w$ and the result averaged across the observed state space as shown in equation 5.

$$r_{mt} = \frac{\sum_g p_g}{|\mathcal{R}| \times |\mathcal{O}_g| \times \left|\mathcal{I}_g^*\right|} \quad (4) \qquad r_{pf} = \frac{\sum_g \frac{P_g}{T_{\mathcal{O}}^w}}{|\mathcal{R}| \times |\mathcal{O}_g| \times \left|\mathcal{I}_g^*\right|} \quad (5)$$

C. Advantage Function

We use a standard actor-critic architecture where the advantage function is estimated for each agent, g, as in eqn. 6.

$$A_g^{\pi_\theta}(s_t, a_t) = Q_g^{\pi_\theta}(s_t, a_t) - V_g^{\pi_\theta}(s_t) \quad (6)$$

Where $Q_g^{\pi_\theta}$, is estimated as $r_t + V_g^{\pi_\theta}(s_{t+1})$ and $V_g^{\pi_\theta}$ is estimated from each actor's separate critic neural network.

D. Partial Reward Down-Sampling

Down-sampling, denoted by Φ in algorithm 2, is a method to reduce the number of rewards samples shared between cooperating agents at the expense of learned policy effectiveness. An agent partial reward trajectory, τ_{g_p}, is down-sampled by averaging rewards across a discrete number of consecutive time-steps (N), before distribution to cooperating agents in $\mathcal{I}_g$. A complimentary Φ operation, using the same N, must be performed on $\log \pi_\theta(a|s)$ in each agent receiving a partial reward. The effective result of down-sampling is to group the likelihood of Bernoulli action combinations across aggregates of discrete time-steps.

E. Sample Efficiency

As in [4], we assume that the SARL implementation has access to the complete state and action space. The MARL has only partial access to the state space and this is determined by the number of reporting devices $\mathcal{O}_g$ and collaborating agents $\mathcal{I}_g^*$. For the purpose of this work we define the observation sample efficiency, $o = \frac{|\mathcal{R}| \times |\mathcal{B}| \times |\mathcal{U}|}{|\mathcal{R}| \times |\mathcal{O}_g| \times \left|\mathcal{I}_g^*\right|}$, as the ratio of SARL to MARL state space, with a higher o being preferable.

VI. Experimental Results

To evaluate our distributed multi-agent actor-critic (DAC) algorithm we compare against the SSE, Ruse and fairness metrics generated against a single agent actor-critic implementation (SAC, in this context) operating in the same system model with equivalent configurations. Both the SA(RL) and MA(RL) results are compared against two baselines: 1) RUSE-1, using a Ruse of one, with all $\mathcal{R}$ in all beams $\mathcal{B}$ utilized. 2) RUSE-R, a random binomial schedule using a Ruse factor, R, equal to that produced by the SA or MA algorithms. As shown in previous work [4], the SA algorithm will always learn a working policy that gives significantly higher SSE at lower Ruse, and thus better energy efficiency. We use distinct experimental set-ups with increasingly more challenging learning complexity. As our method is designed to minimize cell edge interference, we tune our traffic models to reflect this. During system model execution, algorithm 2 executes for a specified number of TTI iterations that constitute an episode. A number of episode trajectories are collected into a batch, at which point the policy is updated, as described in algorithm 2. The algorithm learning phase terminates when the policy $\pi(s)$ entropy average across all agents converges towards zero and stabilizes, or when the episode number exceeds 10000. In DAC implementation, we set $\left|\mathcal{I}_g^*\right| = 4$ (i.e three cooperating agents and self) and $|\mathcal{O}_g| = 3$ for all experiments unless otherwise noted. In these cases, the sample efficiency, o, as defined in section V-E is ≈ 20, meaning that the DAC implementation sees, on average, 5% of the SA. In general, training DAC takes in the order of 3-5 times the number of episodes required by SAC (≈ 1500 vs $\approx 4500 - 7500$). We use two system model orientation configurations. The first sets up a cell beam orientation, with center and opposing beams where cell edge interference will be high, as illustrated in Fig. 1. The second generates a random, seed-based, network configuration with *ad hoc* cell locations and beam orientations. For both configurations, we apply the following static system model settings: $|\mathcal{C}| = 3$, $|\mathcal{B}| = 9$, $|\mathcal{U}| = 27$, $\mathcal{W} = 30\,\mathrm{dBm}$ and system dimensions $z = 1\,\mathrm{km}^2$. Learning rate was set to $\alpha = 1.5 \times 10^{-5}$ and is a critical success parameter, while γ is not. We use one of four user distribution models for the duration of each experiment: D-1) stationary user (UE) placement around cell edges for experiment duration; D-2) cell edge placement according to a normal distribution, and new user placement generated each

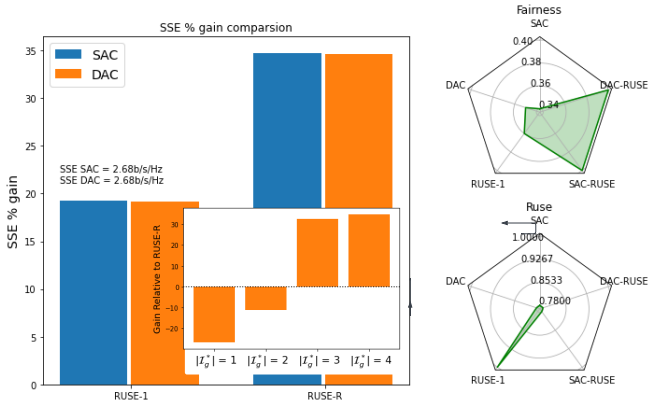

Fig. 2. Experiment set 1: MT, stationary UE pattern, RB = 6

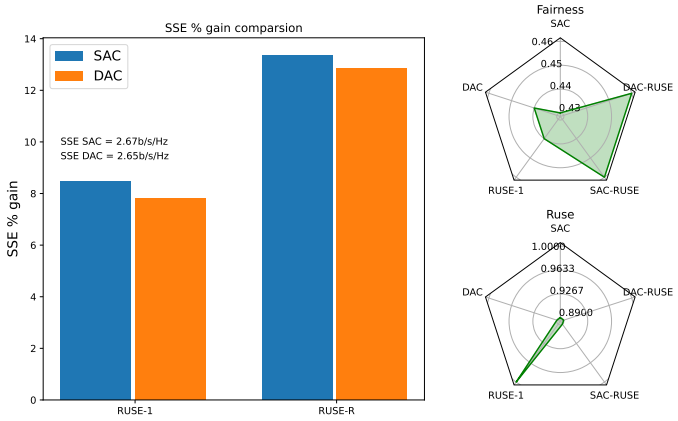

Fig. 3. Experiment set 2: MT, cell edge mobility, RB=15

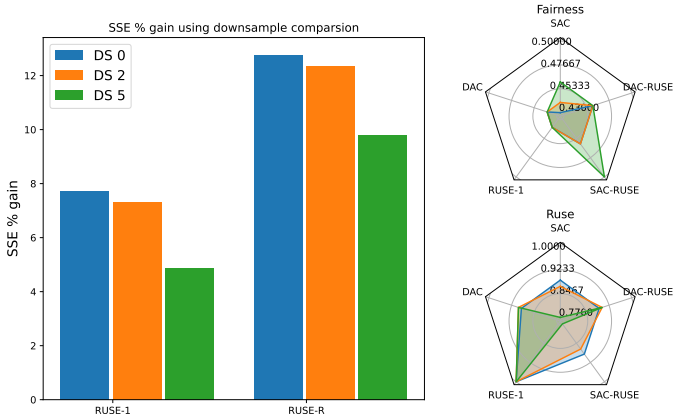

Fig. 4. Experiment set 2: MT with DS, cell edge mobility, RB=15

episode (see Fig. 1 for visualization); D-3) hot-spot user mobility at cell and beam intersections, and new user placement generated each episode (see Fig. 1-inset); D-4) uniformly random user placement each episode.

A. *Experiment Set* 1*: Stationary Users*

Experiment set 1 uses a stationary cell edge UE distribution (D-1). Fig. 2 shows the MT target result with similar SAC and DAC SSE gains, at 19% and 35% respectively, against RUSE-1 and RUSE-R baselines. Fairness is similar for SAC and DAC at ≈ 0.34, while fairness for both SAC and DAC binomial random schedules (SAC-RUSE and DAC-RUSE, respectively) is slightly higher as a random schedule will, on average, expose more users to the scheduler. Of note is that both RL algorithms achieve significantly better SSE at much lower Ruse (≈ 0.78) compared to RUSE-1. Fig. 2-inset shows the importance of selecting a sufficient number of cooperating agents $|\mathcal{I}_g^*|$, with four being optimal in this case. Below three DAC performs worse than all baselines, with $|\mathcal{I}_g^*| = 1$ equivalent to an independent AC algorithm (IL with no reward sharing). This clearly demonstrates the importance of our partial reward sharing approach.

B. *Experiment Set* 2*: MT Cell Edge Mobility*

Experiment 2 uses MT target and an aggressive mobility model according to D-2 and $\mathcal{R} = 15$, for each $\mathcal{B}$. Fig. 3 shows SAC and DAC results to be comparable, with a very modest decrease in DAC SSE for both baselines. Fig. 4 shows the result of down-sampling experiment 2, according to alg. 2 and section V-D, with rates 2 and 5 (DS 2 and DS 5, respectively). This demonstrates the potential to reduce the partial reward sample volumes between agents with a factor of 2 and 5, while trading SSE efficiency and training time (episodes), which for this experiment was: DS 0 $\approx$ 5200; DS 2 $\approx$ 8500; DS 5 terminated at 10000.

C. *Experiment Set* 3*: PF Cell Edge Mobility*

Experiment 3 uses a similar configuration to experiment 2 but with a PF target. Fig. 5 shows SSE comparable (slightly lower for DAC) gains for SAC and DAC using a fairness weight $w = 0.5$, which tends towards moderate overall fairness. Ruse for DAC is slightly higher (≈ 0.89 vs ≈ 0.84) but fairness is comparable at (≈ 0.68).

D. *Experiment Set* 4*: MT Random ad hoc Configurations*

Experiment 4 uses a random, seed-based, network configuration with *ad hoc* cell locations and beam orientations. New users are randomly placed each episode according to model D-4. Fig. 6 shows average utilization metrics across 10 randomly generated *ad hoc* configurations. DAC-1 sets $\left|\mathcal{I}_g^*\right| = 4$ and $|\mathcal{O}_g| = 3$ and underperforms SAC by an unacceptable margin. DAC-2 increases cooperating agents and number of sample reporting devices to $\left|\mathcal{I}_g^*\right| = 5$ and $|\mathcal{O}_g| = 5$, respectively. As a result sample efficiency, o, is reduced (≈ 20 for DAC-1 and ≈ 12 for DAC-2) but SSE % gain is considerably increased. DAC-2 Ruse is slightly higher compared to SAC (≈ 0.69 vs ≈ 0.64), but we consider this an acceptable compromise given the very favourable implementation characteristics of DAC.

VII. Conclusion

Self-learning methods, such as reinforcement learning, show vast potential as core enabling techniques in automating and optimizing the operation of complex systems such as LTE and 5G. To be widely applicable in these systems, however, it must be possible to deploy the methods in resource-limited and highly distributed settings. Other characteristics such as the ability to continuously learn while in operation, and sample efficiency while observing and distributing data are

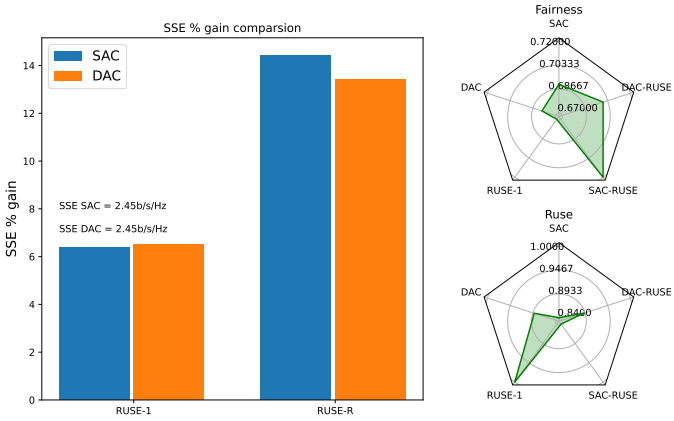

Fig. 5. Experiment set 3: PF, cell edge mobility, RB=15, $w = 0.5$

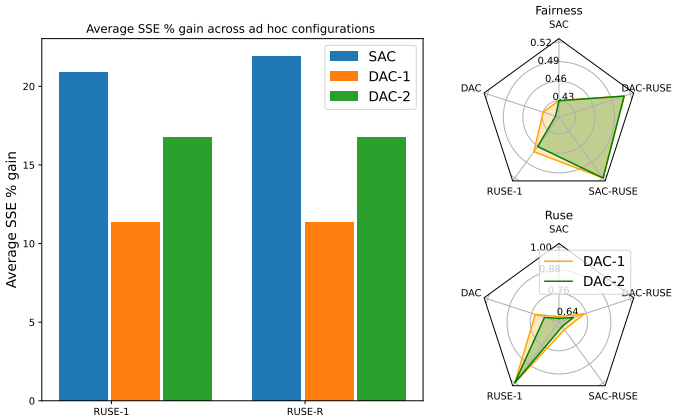

Fig. 6. Experiment set 5: MT, *ad hoc* beams, random episodic users, RB=6

also essential. The distributed multi-agent actor-critic (DAC) using partial-reward sharing demonstrated in this work meets these criteria. We show that our multi-agent approach performs on par with a single-agent equivalent in many cases and to within a few percentage points (2-5%), on average, in many challenging *ad hoc* cases. While we strongly believe DAC is suitable for use in many real-world multi-agent environments but acknowledge there are currently some design limitations that need to be addressed in future work. One important enhancement is the incorporation of agents using diverse reward functions. Currently DAC can only successfully operate in environments where the same reward function is used by all cooperating agents.

Acknowledgment

Kreuger is partially funded by Ericsson. Corcoran and Boman are partially funded by the WASP (Wallenberg Autonomous Systems and Software Program) research program.

References

[1] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network NR, Stage 2 (Release 15)," Tech. Rep. TS 36.212 V15.1.0, 2018.

[2] S. Hämäläinen, H. Sanneck, and C. Sartori, *LTE self-organising networks:Network management automation for operational efficiency*. Wiley, 2012.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. MIT press, 2018.

[4] D. Corcoran, P. Kreuger, and M. Boman, "Reinforcement learning for automated energy efficient mobile network performance tuning," in *2021 17th International Conference on Network and Service Management (CNSM)*, 2021, pp. 216–224.

[5] M. Liu *et al.*, "Review of digital twin about concepts, technologies, and industrial applications," *Journal of Manufacturing Systems*, vol. 58, pp. 346–361, 2021.

[6] U. Ozguner, "Decentralized and distributed control approaches and algorithms," in *Proceedings of the 28th IEEE Conference on Decision and Control,*. IEEE, 1989, pp. 1289–1294.

[7] D. Lopez-Perez *et al.*, "Enhanced intercell interference coordination challenges in heterogeneous networks," *IEEE Wireless Communications*, vol. 18, no. 3, pp. 22–30, 2011.

[8] V. Marik and D. McFarlane, "Industrial adoption of agent-based technologies," *IEEE intelligent systems*, vol. 20, no. 1, pp. 27–35, 2005.

[9] R. Lowe *et al.*, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc NIPS 30*, I. Guyon *et al.*, Eds., 2017.

[10] M. J. Mataric, "Using communication to reduce locality in distributed multiagent learning," *Journal of experimental & theoretical artificial intelligence*, vol. 10, no. 3, pp. 357–369, 1998.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[12] M. Sniedovich, "A new look at Bellman's principle of optimality," *Journal of optimization theory and applications*, vol. 49, no. 1, pp. 161–176, 1986.

[13] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[14] R. S. Sutton *et al.*, "Policy gradient methods for reinforcement learning with function approximation," in *Proc NIPS 12*, S. Solla, T. Leen, and K. Müller, Eds. MIT Press, 2000, pp. 1057–1063.

[15] J. Foerster *et al.*, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[16] P. Sunehag *et al.*, "Value-decomposition networks for cooperative multi-agent learning," 2017.

[17] T. Rashid *et al.*, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018.

[18] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Comparative evaluation of cooperative multi-agent deep reinforcement learning algorithms."

[19] E. Dahlman, S. Parkvall, and J. Skold, *4G, LTE-Advanced Pro and The Road to 5G*, 3rd ed. USA: Academic Press, Inc., 2016.

[20] L. Liu *et al.*, "Time-domain ICIC and optimized designs for 5G and beyond: a survey," *Science China Information Sciences*, vol. 62, no. 2, p. 21302, 2018.

[21] F. Capozzi *et al.*, "Downlink packet scheduling in LTE cellular networks: Key design issues and a survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 678–700, 2013.

[22] M. Toril *et al.*, "Estimating Spectral Efficiency Curves from Connection Traces in a Live LTE Network," *Mobile Information Systems*, vol. 2017, June 2017.

[23] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 5, pp. 834–846, 1983.

[24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.