# Intent-based multi-agent reinforcement learning for service assurance in cellular networks

Satheesh K. Perepu*, Jean P. Martins†, Ricardo Souza S† and Kaushik Dey*
*Ericsson Research, India, †Ericsson Research, Brazil
Email: {perepu.satheesh.kumar, jean.martins, ricardo.s.souza, deykaushik}@ericsson.com

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001426

***Abstract*—Recently, intent-based management is receiving good attention in telecom networks owing to stringent performance requirements for many of the use cases. Several approaches on the literature employ traditional methods in the telecom domain to fulfill intents on the KPIs, which can be defined as a closed loop. However, these methods consider every closed-loop independent of each other which degrades the combined closed-loop performance. Also, when many closed loops are needed, these methods are not easily scalable. Multi-agent reinforcement learning (MARL) techniques have shown significant promise in many areas in which traditional closed-loop control falls short, typically for complex coordination and conflict management among loops. In this work, we propose a method based on MARL to achieve intent-based management without the requirement of the model of the underlying system. Moreover, when there are conflicting intents, the MARL agents can implicitly incentivize the loops to cooperate, without human interaction, by prioritizing the important KPIs. Experiments have been performed on a network emulator on optimizing KPIs for three services and we observe the proposed system performs well and is able to fulfill all existing intents when there are enough resources or prioritize the KPIs when there are scarce resources.**

***Index Terms*—multi-agent reinforcement learning, cognitive networks, intent-based networking**

## I. Introduction

The future networks like 6G will primarily be driven by Intents from network operators. Such intents may consist of one or more objectives. Intents are received and need to be fulfilled by an Intent Manager. Within each network slice, there could be multiple such objectives and each in turn may need one or more parameters to be optimized. In telecom domain, each such objective is represented by a Key Performance Indicator (KPI) goal can be achieved by one closed loop which optimizes the respective parameter(s). Quite often these closed loops are interacting i.e. any action performed by one closed loop may affect the KPIs assured by another closed loops. Optimization for each objective in isolation or in some cases sequentially [1] is possible through existing methods in literature, but the challenge manifests when the objectives conflict with each other and the model of the environment is not available. It is to be noted that the operator or higher domain functions, which are feeding such objectives, may not be aware of such conflicts and hence pre-planning on conflict avoidance may not be feasible in all scenarios. In such circumstances, the efficiency of future networks would depend on ability to autonomously manage multiple conflicting objectives and adapt [2] to the expectations based on priority of individual intents.

In our work, we simulated a situation where an Intent Manager has to fulfill 3 intents for three different services, Conversational Video (CV), Ultra Reliable Low Latency Communication (URLLC) and massive IoT (mIoT). Each of these may involve optimizing multiple parameters in the network. For tractability, we have chosen packet priority and Maximum Bit Rate (MBR) as the two controlling parameters. Now considering a resource constrained environment, increase of packet priority might improve the Quality of Experience (QoE) of the CV service but might degrade the packet loss of URLLC service. Similarly, increasing MBR for URLLC may improve packet loss and reduce latency for URLLC but may degrade the mIoT and CV service. Additionally the target for each of the services might change frequently and the model needs to respond towards the change without any additional training cycles. So in summary, the crux of the challenge is to optimize the realization of the intents, some of which may conflict with each other, in a resource constrained network setting.

A classical optimization technique is often not suitable as the model of the environment is not available and also the compute for optimization may not be available at run time with the targets for the goals changing frequently. Also any other model driven technique may not be scalable for the aforesaid reasons.

We have chosen a model-free technique using Multi-agent Reinforcement Learning (MARL) to solve this problem. Here each of the agents are responsible for tuning a parameter related to the objective defined by the intent. A CV may be optimized by at least two agents, one for packet priority another for MBR adjustments. The packet priority agent for CV interacts with the packet priority agents for URLLC and mIoT and all these three agents learn to "Plan to Coordinate" to achieve an optimal global trade-off during the training phase. Additionally in our method, during execution phase, none of the agents need to observe actions or rewards (+/- towards goal) from another agent. Each agent can see only the aggregated Global reward and the current KPI (parameter value) for its own service and hence this design prevents any communication bottleneck during execution.

Hence our method is able to address multiple intents simultaneously for circumstances where model of the environment may not be available.

## II. Background

### A. Intent-based closed loops paradigm

One trend in network management automation is the employment of intent-driven closed loops. This paradigm is expected to cause a gradual shift of concerns from human to machine operations, where goals and objectives are conveyed through high-level intents and closed control loops act to fulfill these intents.

Intents can be defined as *formal specification of all expectations including requirements, goals, and constraints given to a technical system* [3]. In summary, intents define states a system should reach, without explicit information on how to reach those states. Intent-based interfaces can be employed by several domains. A fully automated SLA contract system could be steered by business intents while at the same time employ service-level intents to interface with service operation [3]. In natural language, a service-level intent could take to the form of "*I want a conversational video service, where 80% of the users have a QoE of at least 4.0*", and based on that different closed loops would be instantiates to ensure intent fulfillment.

A closed loop can be defined as the management of an entity that have specific goals and that can be monitored and acted upon [4]. According to ETSI ZSM, a closed loop consists of four logical steps (i.e., monitoring, analysis, decision and execution). However, in the context of this work closed loops will be implemented by multiple goal-conditioned Reinforcement Learning (RL) agents. In this approach, intents will provide different goals for the multiple RL agents depending on their scope and the agents will perform all necessary monitoring, decision and actuation tasks on the managed entities.

### B. Related Work

The interaction and conflict handling among multiple intelligent agents or control loops managing different cellular network aspects have been investigated by several authors. Moysen et al. [5] provides a comprehensive survey of several different elements related to Machine Learning (ML) applied to the general area of Self-organized Network Management, for the scope of this work we focus on conflict and coordination between multiple control loops. As identified in [5], most approaches focus on the proposition of a complex coordination mechanisms to handle conflict. These approaches range from applying problem specific heuristics [6], a multi-step workflow to perform coordination while handling scalability issues [7], or defining the coordinator as a RL-agent [8]. One common thread with all the aforementioned approaches is the fact that in all cases individual control loops are independent greedy policies. That is, control loops or agents are either ML-based approaches trained independently or use case specific solutions and, therefore, requires more complex coordination.

For the investigations presented on this work, we considered a multi-agent setup where agents are trained in conjunction with each other to manage multiple services. In this setup, the conflicts are inherently handled by the multi-agent setup and simple coordination mechanism is proposed to simplify the execution and convergence.

### C. Multi-agent Reinforcement Learning

RL comprises a set of techniques that enable an agent to learn to interact with an environment by iteratively exploring and evaluating the outcomes of its actions. The overall goal in RL is to derive policies that map the observed situations perceived by the agent to actions that would maximize the cumulative rewards received by such an agent [9].

There are many cases where multiple agents could be deployed in the environment but we need them to learn, based on context, either collaboration, competition or both. To learn these, we use MARL techniques. However, learning the joint policy is not trivial, as agents usually have a local perception of the whole system (partially observable), and, in many cases, direct communication between them is absent. These and other characteristics define environments in which it is difficult for a learning agent to separate the effect of its actions from dynamics caused by other agents or other unobserved components.

The foremost technique to learn the collaboration between agents in independent Q-learning [10]. However, a problem with this technique is that since each agent will take independent action, it will result in non-stationary environment. To overcome this problem, many value function decomposition methods are proposed like QMIX [11] etc. The idea behind this method is, the agents are trained based on combined value function $Q_{tot}$ which is obtained by combining the individual agent Q-values $Q_i$ using a neural network. In this work, we used QMIX to learn the collaboration between these agents.

In QMIX [11], [12], the main goal is to guarantee that decentralized agents' decisions are consistent with those of a centralized counterpart. That is enforced during the training phase, allowing actuation to be performed in a decentralized fashion (centralized learning with decentralized execution). Each agent's policy $\pi_i$ $(i = 1, \ldots, n)$ is derived from a $Q_i$ function. QMIX add to that setup a *mixing* network $\mathbf{Q}(\mathbf{s}, \mathbf{a})$, with joint states $\mathbf{s} = (s^1, \ldots, s^n)$ and joint actions $\mathbf{a} = (a^1, \ldots, a^n)$, from which a joint policy $\boldsymbol{\pi}(\mathbf{s})$ is derived. During training, the loss function directs the learning of $\boldsymbol{\pi}$ towards producing optimal joint actions while directing $Q_i$ value-functions towards $\mathbf{Q}$. This strategy induces individual policies $\pi(s^i)$ such that $\boldsymbol{\pi}(\mathbf{s}) = (\pi(s^1), \ldots, \pi(s^n))$. That enables us to employ $\pi_i$ for decentralized execution, instead of the centralized $\boldsymbol{\pi}$.

### D. Intent-aware policies via Goal-conditioned RL

In scenarios such as intent-based service assurance, we require RL policies that can adapt to changes in the goals during the execution phase. As an example, suppose an agent is pursuing a certain level of quality of experience, and as over time the target level changes, the agent should seamlessly continue to pursue the new goal, i.e., the agent generalizes over the domain of goals. One way of achieving such results is via goal-conditioned reinforcement learning.

These aspects can be formalized by state-action value functions $Q(o, g, a)$, where $o$ and $g$ come from the KPI domain and reward functions $r(o, g) = d(o, g)$, where $d$ refers to

any similarity measure [13]. During training, $g$ values are randomly chosen from the domain at the beginning of every episode, so that the resulting policy is able to generalize over different goals during the execution phase.

The simpler way of implementing goal-conditioned RL is by adding the goals $g$ as an additional dimension in the observation space. That effectively changes the observation space, which we formalize later. Although there might be limitations to such an approach [13], it allows us to easily employ out-of-the-box RL algorithms in a goal-conditioned setting.

## III. Methodology

### A. Network Emulator

To evaluate our solution, a network emulator was employed with the basic requirements for the problem at hand. The emulator provides an end-to-end environment, from compute resources hosting services (e.g., edge and central sites) to the users of those services. The emulator models gNBs and User Plane Functions (UPFs) connecting the User Equipments (UEs) to the services. Figure 1 illustrates the network topology implemented on the emulator.

Three service types are supported on the current version of the emulator - i.e. CV, URLLC and mIoT services. Services can serve multiple UEs at the same time, while URLLC and mIoT services can have multiple instances. The emulator exposes a collection of Application Programming Interface (API) endpoints. The APIs give the agents access to several data points, including Throughput Uplink (UL)/Downlink (DL), Packet Loss UL/DL, Number of Bytes UL/DL, services' packet priority, UEs MBR, Service latency and Service QoE. It also allow the agents to act on the environment by modifying individual UEs MBR and Services' packet priority on the network. To handle interactions with the network emulator a gym-like interface was implemented and exposed to the agents.

### B. MARL Environment specification

In our scenarios, multiple closed loops are interacting while optimizing their particular KPIs by tuning particular control knobs. Those characteristics translate to MARL formalism as multiple heterogeneous agents for which the conflicting demands for resources require cooperative behavior.

In this work, we employ traditional QMiX [11] to train agents able to cooperate without communication in decentralized settings. The objective is to arrive at goal-conditioned agents able to adapt to dynamic changes in the intents while also maintaining high performance in an open MARL setup, where agents may join or leave at any time. Next, we specify the RL components defined on top of the network emulator infrastructure.

*1) Observation space:* Since we deal with multiple heterogeneous agents, we first specify the local observation spaces, and later the joint or global observation space.

Each service type is associated with a single KPI, e.g., while the quality of a conversational video service is measured in terms of QoE, the quality of an URLLC service might be measured in terms of packet losses. Overall, the first dimension of the local observation spaces is always the respective KPI measurement, while the second dimension is the KPI target defined by the intents. To facilitate the emergence of cooperation during training, we also include the global reward $G \in \mathbb{R}$ and the number of UEs $n \in \mathbb{N}$ using the service as additional dimensions.

Considering $T$ as a service type index, and $\mathcal{K}_T$ as the KPI ranges associated with $T$, the local observation spaces are defined as $\mathcal{S}_T \subseteq \mathcal{K}_T \times \mathcal{K}_T \times \mathbb{R} \times \mathbb{N}$. The specific KPI and their domains are specified in Table I. For easy notation, we consider $s = (o_j, g_j, G, n_j) \in \mathcal{S}_j$, where $j$ is an index referring to the service types, and $o_j, g_j \in \mathcal{K}_j$.

TABLE I
Service types, KPIs and domains.

| # | Service type $T$ | KPI | $\mathcal{K}_T$ |
|---|---|---|---|
| 1 | Conversational video (CV) | QoE | $[1, 5]$ |
| 2 | URLLC | Packet-loss ratio (PLR) | $[0, 1]$ |
| 3 | mIoT | Packet-loss ratio (PLR) | $[0, 1]$ |

Altogether, the local observation spaces compose a joint observation space that can be interpreted as the global state of the MARL environment. Such joint observation space is based on the concatenation of all local observation spaces, which is composed of elements $\mathbf{s} \in \boldsymbol{\mathcal{S}}$.

*2) Reward function:* Analogous to the observation spaces, the reward functions are also divided in local and global. The local reward functions concerns the individual KPIs domains, and are defined to comply with the goal-conditioning setup, i.e., $r_j(o, g) = |o - g|$, where $o, g \in \mathcal{K}_T$. Such reward functions would output higher values as the current KPI measurement $s$ is further away from the target $g$ in either direction.

Additionally, as we are dealing with heterogeneous agents and each reward function may be in a different range, we scale down the individual rewards into the range $[0, 1]$ by employing a normalization factor $\Delta_j$ that represents the maximum absolute difference between $o_j$ and $g_j$. Therefore, given a pair $(o_j, g_j)$, the reward function for a service $j \in T$ is define as

$$r_j(o_j, g_j) = 1 - (|o_j - g_j|/\Delta_j). \tag{1}$$

From local rewards functions, local observations and goals, a linear global reward function can be defined. Here, we also introduce the notion of penalties $\rho_j$ (or preferences). If all penalties are equal, all services should be equally served, otherwise, those services with higher penalties should be favored at the expense of the others.

$$G(\mathbf{s}) = \sum_{s_j \in \mathbf{s}} \rho_j \cdot r_j(o_j, g_j) \tag{2}$$

Penalties are particularly important whenever we don't have enough resources to achieve all intents. In such cases, equal penalties lead to equal degradation of the service KPIs. By adding the penalty terms to the global reward function, we

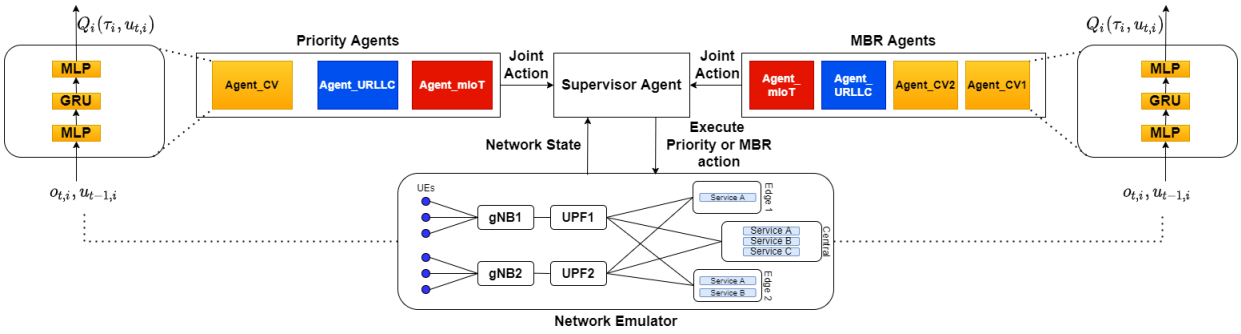


Fig. 1. MARL setup and environment.

can tune the desired behavior by properly choosing different penalty terms for each service.

All the above components are same for both the packet priority MARL agent and MBR MARL agent. However, their action space is different and is explained below.

*3) Action spaces:* Service assurance for each service type can be pursued via tuning of multiple configuration parameters (control knobs), i.e., for each service we can have multiple agents. In this paper, we consider two control knobs the MBR and the packet priority. Therefore, we have two types of agents which we refer to as MBR agents and packet priority agents, both learned via RL. Since QMIX was our algorithm of choice, both action spaces were enforced to be discrete.

In the case of MBR agents, the actions consist of decisions for increasing or decreasing the current MBR value, i.e., $\mathcal{A} = \{-1, 1\}$. The increment/decrements are fixed and defined from the discretization of the original action space $[1, \alpha]$ into bins of size $0.5$ Mbps (where $\alpha$ is the air-link bandwidth). In this setting an increase/decrease action would move to the next/previous bin. After the updated bin $b_t = b_{t-1} + a_t$ is identified, a random value is chosen within its bounds to specify the UE's MBR.

$$\text{MBR}_t = \text{rand}(b_t) \tag{3}$$

Similarly, in the case of priority agents, the actions consist of decisions for increasing or decreasing the current packet priority of a service, i.e., $\mathcal{A} = \{-1, 1\}$. The minimum/maximum priority values are defined to comply with the Network Emulator design, e.g., $[1, 100]$, where a small value denotes high priority.

$$\text{Priority}_t = \text{Priority}_{t-1} + a_t \tag{4}$$

### C. MARL training and evaluation

Each agent is modelled by 2-layer Gated Recurrent Unit (GRU) network with 2 nodes in each layer. During training, QMIX benefits from local and global observations and rewards, i.e., centralized training. The training phase consists of multiple episodes, each lasting for a maximum of $H = 30$ time steps or until all agents have reached their intents. During the evaluation phase, agents only rely on their local observations to reach their individual goals, i.e., decentralized execution.

Packet priority and MBR agents have different scopes, as specified by the network emulator used for the experiments. While packet priority is defined at the service level, MBR is defined at the UE level. Although it is scalable to set packet priority for each service, it may not be realistic to set MBR to individual UEs. Therefore, MBR agents take decisions for sets of UEs instead.

The rationale for grouping UEs depend on the format of the intents supported. Each intent follows a scheme *'$x\%$ of UE's to have KPI greater than $g$'*. Hence, we can divide the UEs from service into two groups of sizes $x\%$ and $(100-x)\%$. The MBR agents' actions then affect all the UEs within a group equally, all of them getting the same MBR value. If the intent is on $100\%$ of UEs then the MBR actuation reduces to the service level, analogously to packet priority agents.

Both priority and MBR agent groups were trained independently to reduce the computational overhead. In the current design, the agent groups take turns actuating on the environment. Such an approach requires an additional level of decision-making specifying which agents to activate and deactivate during execution time. In this paper, we evaluate a very simple supervisor agent, which chooses the specific agent to activate based on knowledge about how the network emulator works, following the rules described on Table II.

TABLE II
SUPERVISOR AGENT AND ITS RULES.

| # | Rule | Agent to use |
|---|---|---|
| A. | If all UEs throughput = MBR, | MBR agents |
| B. | Otherwise, | Priority agents |

The supervisor agent decisions take place every 5 time steps, after which the chosen agent (MBR or Priority) stay active for another round. The overall architecture of the proposed method is illustrated by Figure 1.

## IV. Results and Discussions

We evaluated the efficacy of the proposed solution in an end-to-end network emulator, considering two scenarios, (1) the first considers that plenty of resources are available, allowing the fulfilment of all intents, while (2) in the second

resources are scarce. To assess the benefits and limitations of the supervisor agent we also compare the results with those achieved by employing only priority or only MBR agents.

### A. Experimental setting

*1) Performance metric:* To compare and quantify the methods w.r.t goal realization and convergence speed we use a metric defined in (5). The metric computes the average distance between observed KPI values to the intended target.

$$M = \frac{1}{H}\sum_{t=1}^{H}|o_t - g_t| \tag{5}$$

where $H$ is number of time steps, $o_t$ is the observed KPI value and $g_t$ is the intended KPI value (goal) at instant $t$. An ideal agent would produce trajectories whose $M$ is close to zero, i.e., the agent quickly reaches its goals and stay close to it during its whole execution time.

*2) Intents specification:* We use the same set of intents for both scenarios (plenty vs scarce resources). After a period of time, we induce a change in CV service intent (from 1.1 to 1.2 in Table below) to evaluate agent's ability to adapt. This is where goal-conditioned training becomes beneficial. The whole set of intents is defined in Table III.

TABLE III
INTENT DEFINITIONS

| Intent | Service | % of UEs | KPI | KPI target |
|---|---|---|---|---|
| 1.1 | CV | 75 | QoE | $\geq 3.0$ |
| 1.2 | CV | 75 | QoE | $\geq 3.5$ |
| 2 | URLLC | 100 | Packet-loss ratio | $\leq 0.02$ |
| 3 | mIoT | 100 | Packet-loss ratio | $\leq 0.04$ |

*3) Network emulator settings:* We configured 4 UEs for each service and 2 gNBs connected to an equal number of UEs across all the services. At the beginning of an episode, all services are set to the same packet priority (7), while all UEs are set to the same MBR value (1Mbps).

Once the agents change the priority or MBR values, the effect is not immediately observable. The time needed for actions to affect the KPIs depend on the type of action and the current network state. In this work, we used a fixed time window of 40 seconds and 10 seconds to observe the effect of packet priority and MBR changes, respectively.

### B. Results

In this section we illustrate results for three agent group setups: (a) Only priority agents, (b) only MBR agents, and (c) both agents orchestrated via a Supervisor. For brevity, we summarize the overall performance results in tables and show plots only for the third setup.

*1) Scenario 1: Plenty of Resources:* In this scenario, the airlink bandwidth is 20 Mbps, which is enough for all intents being met (there is no bottleneck in the transport layer).

Figure 2 shows the results for this scenario. The figure at left shows how QoE and Packet-loss ratio (lines) reach the KPI targets (dots) defined by the intents. The figure at the right shows how the agents' actions affect priority and MBR values over time. Here, we can also observe the time windows in which each agent is active (e.g., packet priority loop, MBR loop). From the plots, it is evident that how the agents are able to readjust their trajectory even if the goals/intent changes midway. Hence given the agents are goal-conditioned they can reach any goal value from any point in state space.

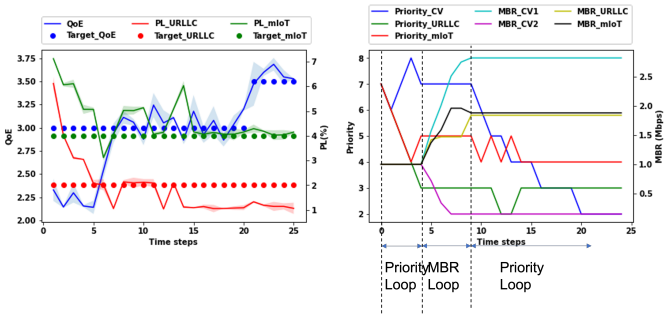


Fig. 2. Results with a supervisor agent and plenty of resources.

The performance results $M$ for all agent group setups are summarized in Table IV. The smaller values indicate that the orchestration of priority and MBR MARL loops, with the help of a supervisor agent, led to faster and better convergence towards the intended goals. Here, we note that the bad performance of MBR agents for mIoT services is expected, since such services require very low throughput and do not benefit much from MBR increases.

TABLE IV
$M$ VALUE COMPUTED FOR AVAILABLE RESOURCES

| Service | Only Priority | Only MBR | Priority & MBR |
|---|---|---|---|
| CV | 1.49 | 0.97 | 0.25 |
| URLLC | 1.27 | 1.68 | 0.85 |
| mIoT | 1.49 | 2.27 | 1.09 |

### C. Scenario 2: Scarce Resources

In this scenario, the airlink bandwidth is 4 Mbps, which is not enough for all intents to be met.

Figure 3 show the results for this scenario. The supervisor agent decides to activate the priority agents twice in first ten time steps, followed by the MBR agents for next five time steps, and then again the priority agents. Here we notice that, due to the scarcity of resources, all observed KPIs are below/above their targets and intents are not met. The performance results $M$ are summarized in Table V.

TABLE V
$M$ VALUE COMPUTED FOR SCARCE RESOURCES

| Service | Only Priority | Only MBR | Priority & MBR |
|---|---|---|---|
| CV | 1.58 | 1.24 | 0.64 |
| URLLC | 1.07 | 1.74 | 0.94 |
| mIoT | 1.84 | 2.45 | 1.94 |

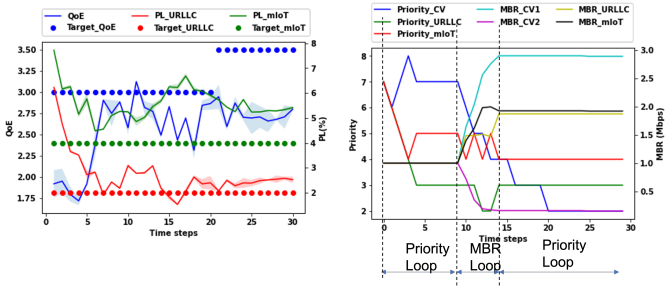


Fig. 3. Results with a supervisor agent, scarce resources and equal penalties.

However, the behavior of equally degrading all services performance may not be desired in all scenarios. From the business perspective, sometimes a specific service has to be prioritized over others even when there are not enough resources. Let us assume, URLLC service, for example, is assigned as high priority i.e. it has to reach its intent even at cost of degrading the others. To achieve such autonomous trade-offs for specific services in scarce resources scenarios, we introduce differentiated penalties for each service, to quantify how the violation of a high priority service intent compares against others. Therefore, to prioritize URLLC over CV and mIoT we associate a higher penalty to it, e.g., $\rho = 10$ for URLLC and $\rho = 1$ to CV. By considering such penalties in the global component of the reward functions, the agents learn to prioritize the URLLC service over others.

Figure 4 show the results for scarce resources with differentiated penalties. From these results we observe that the URLLC Packet-loss ratio now achieves its intent, which indicates that penalties are a valid approach for balancing service preferences. Also, from the plot it can be observed that the other two services got degraded, i.e agents learned to trade-off, to enable URLLC service reach its intent.

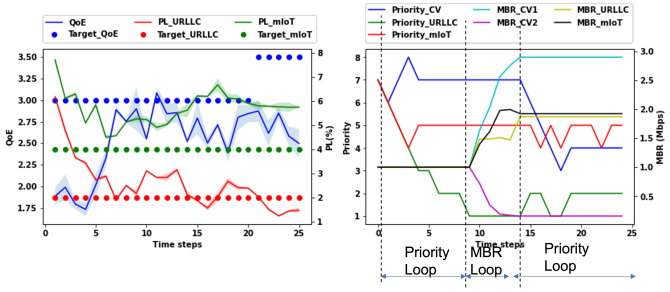


Fig. 4. Results with a supervisor agent, scarce resources and differentiated penalties. Agents learn to trade-off for a prioritized service.

## V. Conclusions

In this paper, we proposed a method based on MARL to fulfill objectives defined by multiple intents. Additionally, we formulated goal-conditioned MARL agents which helps to generalize agents' ability to seek goals across a wide range of values. The proposed approach is tested on a network emulator for three services CV, URLLC and mIoT for varying amount of resource availability. The experiments demonstrate that agents can learn the "Plan to Coordinate" in order to manage conflicts and promote cooperation towards maximizing the global goal. When there are enough resources available, the trained MARL agents coordinate to achieve all intents. Also, when resources are scarce, based on intent priorities (denoted by penalties in this work), the agents learn an optimal trade-off mechanism, which resulted in proportional degradation of an intent to prioritize another intent of higher priority. Finally with use of supervisor agent, we demonstrate a method to autonomously coordinate multiple MARL loops which could later be extended to hierarchical control across domains.

Future directions include extending the approach to larger number of agents, designing generic agents and testing the approaches with different reward functions.

## References

[1] J. Moysen, M. Garcia-Lozano, L. Giupponi, and S. Ruiz, "Conflict resolution in mobile networks: A self-coordination framework based on non-dominated solutions and machine learning for data analytics," *IEEE Computational Intelligence Magazine*, vol. 13, no. 2, pp. 52–64, May 2018.

[2] J. Niemoller, L. Mokrushin, S. Mohalik, V. Konchylaki, and Sarmonikas, "Cognitive processes for adaptive intent-based networking," Ericsson Technology Review, Kista, Sweden, Tech. Rep., 2020. [Online]. Available: https://www.ericsson.com/4abd97/assets/local/reports-papers/ericsson-technology-review/docs/2020/adaptive-intent-based-networking.pdf

[3] TM Forum Autonomous Network Project, "Ig1253 intent autonomous networks," https://www.tmforum.org/resources/how-to-guide/ig1253-intent-in-autonomous-networks-v1-1-0/.

[4] ETSI GS ZSM 009-1, "Zero-touch network and service management (zsm): Closed-loop automation; enablers group specification."

[5] J. Moysen and L. Giupponi, "From 4g to 5g: Self-organized network management meets machine learning," *Computer Communications*, vol. 129, pp. 248–268, 2018.

[6] P. Muñoz, R. Barco, and S. Fortes, "Conflict resolution between load balancing and handover optimization in lte networks," *IEEE Communications Letters*, vol. 18, no. 10, pp. 1795–1798, 2014.

[7] D. F. Preciado Rojas and A. Mitschele-Thiel, "A scalable son coordination framework for 5g," in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–8.

[8] O. Iacoboaiea, B. Sayrac, S. Ben Jemaa, and P. Bianchi, "Son coordination for parameter conflict resolution: A reinforcement learning framework," in *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2014, pp. 196–201.

[9] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[10] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

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

[12] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.

[13] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1312–1320.