Distributing Intelligence for 6G Network Automation: Performance and Architectural Impact

Sayantini Majumdar[*+], Riccardo Trivisonno[*], Wint Yi Poe[*], and Georg Carle[+]
[*]Munich Research Center, Huawei Technologies, Germany
[+]Dept. of Informatics, Technical University of Munich, Germany
Email:[sayantini.majumdar, riccardo.trivisonno, wint.yi.poe]@huawei.com, carle@net.in.tum.de

***Abstract*—In future 6G networks, distributed management of network elements is expected to be a promising paradigm. Recent research progress in Artificial Intelligence (AI) is rapidly driving the adoption of distributed management. However, distributed management using intelligence or distributed AI inherently suffers from a number of issues – potential conflicts, signaling required to ensure cooperation and the convergence time of the algorithm. To this end, an early understanding and analysis of the overall effort to implement distributed AI in 6G, is still unexplored. This work, therefore, examines the impact of distributed AI, by analyzing its performance and how the existing 5G architecture could be enhanced to support it in 6G. We aim to understand the impact of distributed AI in 6G by selecting a relevant beyond 5G use case – auto-scaling virtual resources in a network slice. We present the performance and architecture analysis for two distributed algorithms from the domain of Reinforcement Learning – Q-Learning and Deep Q-Networks. We argue that despite its aforementioned issues, distributed AI brings benefits such as dynamic and adaptive decision-making, making it highly applicable for certain use cases in 6G.**

***Index Terms*—6G network automation, 6G architecture, auto-scaling, distributed intelligence, Reinforcement Learning, 3GPP management**

I. Introduction

The Third Generation Partnership Project (3GPP) has established the technical specifications for the Fifth Generation (5G) mobile network architecture. The management of 5G network elements (NEs) is still mostly performed in a centralized manner, in central cloud servers [1]. For example, the 5G management plane function, Management Data Analytics Function (MDAF), is cloud-native and predominantly centralized [2].

However, centralized management suffers from a number of issues. First, the monitoring data from the NEs needs to be communicated to a single, central server. This causes the server to become a single point of failure. Although identical servers running in parallel could introduce redundancy, this solution is not feasible in the long-term when NEs are distributed several thousands of kilometers away [3]. Especially in time-critical use cases, (*e.g*. in an edge computing factory automation scenario [3]), increased signaling increases the latency of decision-making [4] – a critical system Key Performance Indicator (KPI). Further, for management decisions localized in one part of the network (*e.g*. dynamic placement of VNF instances in an edge cloud [5]), monitoring data from all other NEs is unnecessary. Hence, to overcome the drawbacks of: 1) signaling to a single point of failure, and 2) processing redundant global data for local decision-making, a distributed management paradigm is gaining significant attention. Distributed management is capable of local, dynamic, fast decision-making, and is more efficient in terms of signaling and utilization of edge resources compared to a centralized paradigm. In fact, even pre-standardization bodies such as the European Telecommunications Standards Institute (ETSI) Zero Touch network & Service management are supporting a distributed approach [6].

One specific motivation of why distributed management is more promising than the centralized relates to the introduction of network slicing (NS) in 5G, enabled by Network Function Virtualization (NFV). NFV decouples the software, or Network Functions (NFs), from hardware – resulting in Virtual Network Functions (VNFs) that run on NFV Infrastructure (NFVI) [7]. An NS is a logical network composed of several NFs and several NSs may be deployed on shared NFVI. Therefore, any type of NEs (*e.g*. NS and NS subnet instances, NFs, base station functions (gnBs) etc.) can be virtualized [8]. While the NFs are managed by the 3GPP management system [8], the management and orchestration (MANO) of the NFVI and VNFs are standardized by ETSI NFV-MANO [9]. Hence, with regard to the 5G architecture that is inherently more heterogeneous and modular than previous generations, distributed management is a more efficient alternative.

Problems also exist in distributed management today. For example, 3GPP defines an automation function, that provides and consumes management-related services in a service-based architecture (SBA), called a Management Function (MnF) [8]. Since MnFs perform local decision-making in a distributed manner, coordinating these decisions towards a common goal, that both satisfies system KPIs and overcomes the issues of centralization is challenging. Further, sharing of underlying computing resources, enabled by virtualization, may introduce potential conflicts among MnFs, further complicating distributed network management [10]. Until recently, there was a consensus on the sub-optimal performance of distributed approaches [11]. However, recent works show that Artificial Intelligence (AI) algorithms applied for distributed management have the potential to reach near-optimal performance [10], [12] with reasonable convergence guarantees [13].

Related work in architecture analysis. Very few related works have analyzed the integration of AI from an architecture perspective. The authors in [10] describe the modularized system architecture of ETSI Experiential Networked Intelligence (ENI) group for closed loop automation, that enhances the network operator's experience [14]. This work also presents a mapping of ENI to the 3GPP and ETSI NFV-MANO architecture. Referring to specific use cases, such as slice-aware resource management, the authors propose embedding AI in the 3GPP Network Slice Management Function (NSMF) and Network Slice Subnet Management Function (NSSMF), aided by the MDAF. Next, the work in [15] proposes enhancements in the 5G architecture targeting the diverse requirements of vertical industries. It also proposes an AI-based data analytics framework, presenting a mapping between 3GPP, ETSI NFV-MANO and the proposed AI-based framework. The centralized AI engine, running the algorithm **DeepCog** [16], is embedded in the MDAF. However, these prior works do not consider *how distributed AI would fit in existing 3GPP or ETSI NFV-MANO architectures*, to understand the effort needed for the same in 6G.

This paper aims to understand the aspects of distributing AI in the future 6G system, by analyzing the system performance of distributed AI and how it fits into the existing 5G architecture. To this end, we focus on a specific beyond 5G (B5G)-relevant automation use case, auto-scaling virtual CPU resources of NFs in a NS, defined in [17]. In the automated system, a NS is composed of NFs, where the functionality of each NF is implemented on an individual VNF. The VNFs consume virtual computing resources (*i.e.*, CPUs) from a shared pool. The NS implements distributed admission control. Auto-scaling, therefore, ensures that for a given incoming NS load (of users), the load served is maximized, while keeping individual VNF CPU utilization as close to a pre-defined target as possible. To solve the auto-scaling problem, we proposed distributed AI solutions based on two Reinforcement Learning (RL) algorithms, Q-Learning for Cooperation (**QLC**) [17] and Deep Q-Network for Cooperation (**DQNC**) [18], and evaluated their system performance (in terms of the served load).

We summarize the main contributions of this work below:

1) For the auto-scaling use case, we examine, via extensive simulation scenarios, how well distributed AI, **QLC** and **DQNC**, perform compared to a centralized, optimum Mixed Integer Optimization (**MIO**) mechanism. We further show that distributed AI outperforms a threshold-based auto-scaling mechanism or **THR**, currently inbuilt in the Open Source MANO (OSM) [19] platform.
2) We investigate how the existing 5G architecture could be enhanced to support distributed AI in 6G. For this purpose, we select the auto-scaling use case.

II. RL-based Distributed Intelligence Algorithms

To analyze the performance of distributed intelligence and the architectural impact in the rest of the paper, in this section, we provide an overview of the distributed RL algorithms, Q-Learning for Cooperation (**QLC**) and Deep Q-Network for Cooperation (**DQNC**). All algorithmic aspects required to understand the contributions of this paper are described here. For precise details, the reader is directed to [12], [18], [20].

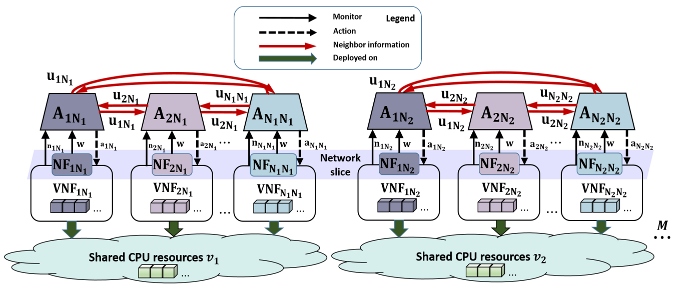


Fig. 1: Distributed intelligence-based auto-scaling system [12]

The managed system, in Fig. 1, consists of a NS composed of total N NFs. These N NFs are grouped into M groups, where M is the no. of shared CPU pools. The no. of NFs in the j^{th} group is denoted by N_j, where $1 \leq j \leq M$. NF$_{ij}$ is implemented on its corresponding VNF, denoted by VNF$_{ij}$, where $1 \leq i \leq N_j$. The NS implements distributed admission control, that ensures individual VNF CPU utilization u_{ij} does not exceed a threshold AC. During high incoming NS load, the VNFs may attempt to scale up their CPUs from their respective shared pool. When the CPUs available in the shared pool are insufficient, a **conflict** occurs. The reactive conflict resolution mechanism forces only *one* of the VNFs to scale up successfully, while the others end up with no extra CPUs required to serve the given incoming load. The result is an unbalanced resource allocation among VNFs, that degrades system KPIs. Therefore, an efficient, distributed auto-scaling mechanism should: i) maximize served load, ii) balance CPU resources and iii) reduce conflicts, for *any* number of NFs.

The system is managed by Intelligent Agents (IAs) distributed in the NS. An IA denoted by A_{ij} manages its own NF-VNF pair. In addition, IAs sharing a given resource pool v_j are defined as Neighbor Intelligent Agents (NIAs). The rationale is that an IA sharing resources is capable of influencing the learning of its neighbors through its auto-scaling actions (that often cause conflicts) hence some form of cooperation should be embedded in each IA [17]. The goal of each IA is to ensure that for a given incoming load, the maximum load is served by the NS, its corresponding VNF CPU utilization u_{ij} is as close as possible to a target utilization u_T and conflicts with its NIAs are reduced.

Monitored variables. Each IA monitors its own VNF utilization u_{ij}. In addition, to encourage cooperation, it monitors the utilization of its NIAs.

Discrete state. The state of an IA encodes: 1) how loaded the overall system is, taking the mean of the NIAs utilization, and 2) it calculates how balanced it is with respect to the mean utilization of its NIAs. Both aspects together determine the state the IA finds itself in.

Discrete action. Each IA is able to add/release CPUs from/to the shared pool or maintains the number of CPUs.

Reward model. Based on the chosen action at the current state, each IA receives a reward signal. An action is given a positive reward if it brings the utilization closer to u_T and it is penalized if it causes a conflict.

Q-Learning for Cooperation (QLC). For the given state, action and reward received, an IA running the **QLC** algorithm updates its "knowledge" at that state-action pair according to the Bellman Eqn. [21]. This knowledge represents the *quality* at that state-action pair, known as a Q-value, and these Q-values are stored in its own Q-table.

Deep Q-Network for Cooperation (DQNC). For the given state, action and reward received, an IA running the **DQNC** algorithm trains a neural network [22], updating its parameters via back-propagation. **DQNC** consists of an additional neural network, the target network, to stabilize the training and a data structure known as a replay buffer to break the temporal correlations between the samples collected in it [22].

Learning principle. The action selection strategy of an IA determines its learning principle. When an IA selects an action from $\mathcal{A}$ randomly, it is said to *explore* its environment. Conversely, when it selects the best action at a given state, or the action with the highest Q-value, it is said to *exploit* its currently built knowledge. The probability of exploration, ε, regulates the balance between exploration and exploitation. Hence, this mechanism is called the ε-greedy approach. The algorithms are trained over episodes.

Centralized alternative for comparison. The performance of distributed intelligence is compared to that of a centralized Mixed Integer Optimization (**MIO**) formulation, serving as the optimum. The objective of **MIO** is to maximize the load served while ensuring individual VNF utilization is close to the pre-defined target. The solver periodically allocates the optimal number of CPUs to the VNFs for the given incoming NS load. Since **MIO** orchestrates the scaling for each VNF, there are no conflicts in this approach. For the precise **MIO** formulation, the reader is directed to [12].

III. Distributing Intelligence for 6G

In this section, we investigate how distributed AI (or RL in this paper) maps to the existing 5G architecture, according to the 3GPP specifications [7], [8]. We focus on a specific example use case of auto-scaling resources in an NS for the same purpose. Additionally, we map the centralized approach in 3GPP for the auto-scaling use case for a comparison with the distributed paradigm. We refer to the 3GPP management system as 3GPP-MAN in this article for brevity.

A. Embedding Distributed Intelligence in existing 5G

To map distributed intelligence or the IAs for the auto-scaling use case, we consider 3GPP-MAN and ETSI NFV-MANO. We focus on ETSI NFV-MANO in this paper as it is a relevant reference architecture for auto-scaling in network slicing. To this end, we analyze two functional blocks:

Element Manager (EM) in 3GPP-MAN. According to TS 28.500 [7], the EM is responsible for the life-cycle management of VNFs on an application level. Based on monitoring information (info) it retrieves from the ETSI NFV-MANO, it triggers management actions for the resources associated to the VNF and the physical NE it is implemented on. Specifically, in the use case *"VNF instance scaling through operation request by 3GPP management system"* of TS 28.500 [7], that

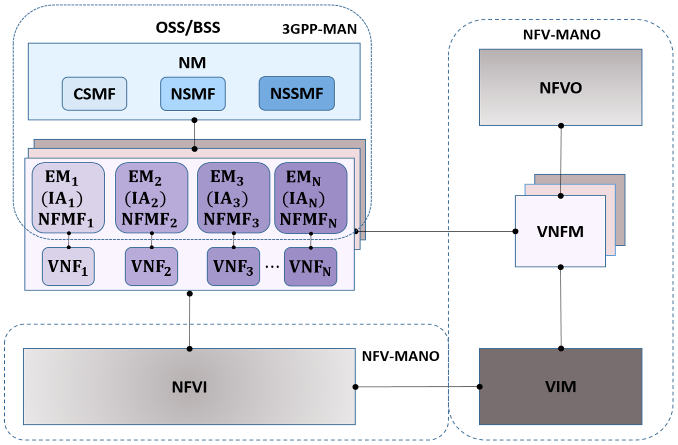


Fig. 2: Distributing AI in the 3GPP-MAN and NFV-MANO architectures

outlines the interactions between the 3GPP-MAN and ETSI NFV-MANO for auto-scaling, EM sends the scaling request to the ETSI NFV-MANO. Hence, the EM, in both the closed-loop decision-making and triggering the request, is actively involved in auto-scaling of its corresponding VNF(s).

Virtual Network Function Manager (VNFM) in ETSI NFV-MANO. The VNF Manager (VNFM) is also responsible for the life-cycle management of VNF instances, as stated in [9]. In other words, each VNF is associated to its own VNFM. Moreover, the VNFM monitors Key Performance Indicators (KPIs) during the life-cycle of its VNF(s), and may use this monitoring info for scaling decisions. Since EM requests the appropriate scaling action, VNFM needs to exchange this essential info regarding its VNF(s) and its resources with its associated EM.

The above analysis shows that since EM is the block triggering auto-scaling based on the monitoring info it receives from the VNFM, the IA logic should be embedded in EM. Moreover, in the context of network slicing, as described in TS 28.533 [8], 3GPP-MAN is further modularized, where several functional blocks may be involved in producing and consuming management services produced by other management blocks. The Network Slice Management Function (MF) or NSMF manages the end-to-end NS on a global level. The Network Slice Subnet MF (NSSMF) manages grouped NF instances, or a slice subnet, independently from the NS. Meanwhile, the Network Function MF (NFMF) manages NF instances. The Communication Service MF or CSMF consumes the management services provided by other blocks. In our proposed distributed AI-based auto-scaling system, we assume each IA manages its own NF-VNF. Since the NFMF is the lowest in the hierarchy that manages NFs or the building blocks of an NS, *we propose embedding each IA from Fig. 1 in an individual NFMF (or EM) instance.* The envisioned mapping is shown in Fig. 2.

We start building the sequence diagram from the **Service Interface**, a standardized service bus defined in the service-based 3GPP-MAN [8]. The proposed sequence diagram is shown in Fig. 3 and is divided into two logical phases. In Phase I, the functional blocks build initial associations with one

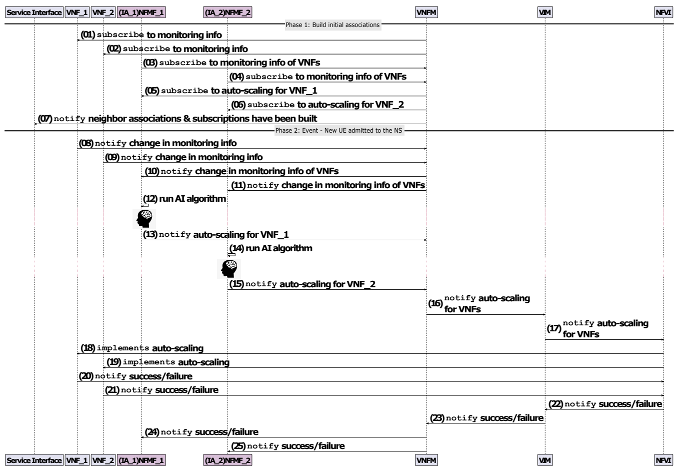

Service Interface
VNF_1
VNF_2
(IA_1)NFMF_1
(IA_2)NFMF_2
VNFM
VIM
NFVI
Phase 1: Build initial associations
(01) subscribe to monitoring info
(02) subscribe to monitoring info
(03) subscribe to monitoring info of VNFs
(04) subscribe to monitoring info of VNFs
(05) subscribe to auto-scaling for VNF_1
(06) subscribe to auto-scaling for VNF_2
(07) notify neighbor associations & subscriptions have been built
Phase 2: Event - New UE admitted to the NS
(08) notify change in monitoring info
(09) notify change in monitoring info
(10) notify change in monitoring info of VNFs
(11) notify change in monitoring info of VNFs
(12) run AI algorithm
(13) notify auto-scaling for VNF_1
(14) run AI algorithm
(15) notify auto-scaling for VNF_2
(16) notify auto-scaling for VNFs
(17) notify auto-scaling for VNFs
(18) implements auto-scaling
(19) implements auto-scaling
(20) notify success/failure
(21) notify success/failure
(22) notify success/failure
(23) notify success/failure
(24) notify success/failure
(25) notify success/failure

Fig. 3: Proposed sequence diagram for distributed AI for two IAs

another through the subscribe-notify model. In Phase II, the logical steps of auto-scaling are executed based on the event that a new UE is admitted to the NS. Due to the flexibility of the architectures, several implementation options are possible. Therefore, the following assumptions hold for the sequencing:

(a) VNFs sharing the same resource pool constitute a single neighbor group.
(b) Only one VNFM instance is associated to all VNFs in the same neighbor group.
(c) In Phase I, the neighbor NFMFs configuration is triggered by the NSSMF, abstracted by the **Service Interface**.

Considering an example with $N = 2$ VNFs sharing a resource pool, the auto-scaling interactions are summarized below:

Phase I. **(01)-(02)** VNFM subscribes to monitor the info of both VNF$_1$ and VNF$_2$. **(03)-(04)** NFMF$_1$ and NFMF$_2$ from the 3GPP-MAN subscribe to VNFM for the monitoring info of both VNFs (local and neighbor info). **(04)-(05)** VNFM needs to subscribe to both NFMFs for the auto-scaling decisions of their respective VNFs. **(07)** VNFM notifies its NSSMF that neighbor associations and subscriptions have been built.

Phase II. A new UE is admitted to the NS. **(08)-(09)** the VNFs notify VNFM about the change in the monitoring info (*e.g*. CPU utilization). **(10)-(11)** VNFM forwards the notification of the change in monitoring info of both VNFs to both NFMFs. **(12)** NFMF$_1$ executes the AI algorithm – calculation of the state, selecting the action, computing the reward and updating the Q-table or the neural network, for **QLC** and **DQNC** respectively. **(13)** NFMF$_1$ notifies the selected auto-scaling action (add, release or maintain CPUs) to VNFM. **(14)-(15)** NFMF$_2$ repeats the steps. **(16)** VNFM notifies the Virtualized Infrastructure Manager (VIM) of the auto-scaling commands. **(17)** VIM notifies the NFVI of the auto-scaling commands. **(18)-(19)** NFVI scales the resources allocated to the VNFs. **(20)-(21)** The VNFs notify the NFVI the result of the auto-scaling. In case of a conflict, the same is signaled to the NFVI. **(22)-(23)** NFVI notifies the same to VIM, and VIM to VNFM. **(24)-(25)** Finally, VNFM notifies both NFMFs the result.

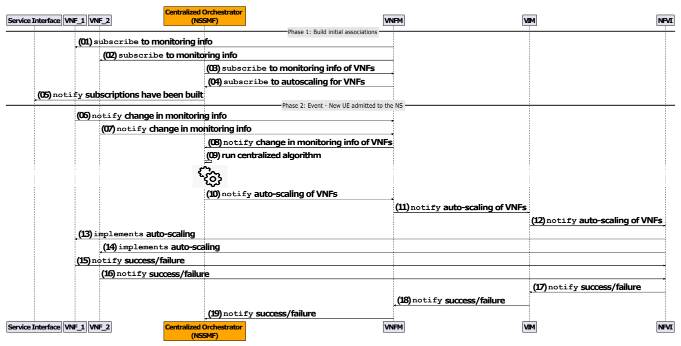

Service Interface
VNF_1
VNF_2
Centralized Orchestrator (NSSMF)
VNFM
VIM
NFVI
Phase 1: Build initial associations
(01) subscribe to monitoring info
(02) subscribe to monitoring info
(03) subscribe to monitoring info of VNFs
(04) subscribe to autoscaling for VNFs
(05) notify subscriptions have been built
Phase 2: Event - New UE admitted to the NS
(06) notify change in monitoring info
(07) notify change in monitoring info
(08) notify change in monitoring info of VNFs
(09) run centralized algorithm
(10) notify auto-scaling of VNFs
(11) notify auto-scaling of VNFs
(12) notify auto-scaling of VNFs
(13) implements auto-scaling
(14) implements auto-scaling
(15) notify success/failure
(16) notify success/failure
(17) notify success/failure
(18) notify success/failure
(19) notify success/failure

Fig. 4: Sequence diagram for centralized orchestrator

TABLE I: Simulation configuration parameters

Type	Parameter	Symbol	Value	Unit
System	Admission control threshold	AC	0.9	-
	CPU utilization target	u_T	0.5	-
	Initial no. of CPU per VNF	n_0	1	-
	Episode duration	T	10^4	s
	Scaling threshold HIGH	u_H	0.92	-
	Scaling threshold LOW	u_L	0.2	-
Load	Population of UEs	U	10^5	-
	Aggregated service requests	Λ_{in}	$[0.05, 1.5]$	s^{-1}
	Service duration (mean, sd)	$\bar{\theta}, \sigma_\theta$	60, 5	s
	Actual load per UE (mean, sd)	$\bar{\mu}, \sigma_\mu$	1, 0.02	-

B. *Centralized Management in 5GS*

In the centralized alternative, the NSSMF serves as the centralized orchestrator. The assumptions of the distributed AI are not valid anymore, as 1) there is no concept of neighbors and the NSSMF must decide auto-scaling for all VNFs in its subnet, and 2) in Phase I, the NSSMF notifies its NSMF when the subscriptions have been built. Fig. 4 depicts the corresponding sequence diagram. Similar to the distributed approach, the sequencing is shown in two phases and the same principles apply in this case.

From Fig. 3 and 4, we observe that the overall signaling of the distributed approach is higher than the centralized, due to the cooperation required between NIAs. Nevertheless, distributed intelligence would be architecturally compatible with 6G, requiring minor enhancements to data management in the existing 5GS.

IV. Evaluation and Discussion

The objective of our evaluation is to show how close the distributed intelligence algorithms based on RL, **QLC** and **DQNC**, perform with regard to a centralized optimum (**MIO**) alternative. We further show the performance gain of distributed AI over a reactive, threshold-based mechanism, **THR**. Additionally, we report the convergence time and the impact on conflicts of both AI algorithms. We remark that our evaluation in this work is built on [18], extending the results by considering varying scenarios of distributed AI deployment.

A. *Simulation Setup*

We develop a realistic 5G network management simulator in Python, specifically for the auto-scaling use case.

Distributed NS deployment. The NS consists of total $N = N_1 + N_2 + ... + N_M$ IAs, where M is the no. of shared

TABLE II: Configuration for each IA

Algorithm	Hyper-parameter	Symbol	Value
DQNC	No. of hidden layers	H	1
	No. of neurons in each hidden layer	n	32
	Loss function	$\mathcal{L}$	Huber loss
	Learning rate	lr	10^{-3}
	Gamma	γ	0.9
	Replay buffer capacity	B	10000
	Mini batch size	b	3000
	Target network update	C	5 episodes
	Initial, final, decay epsilon	$\varepsilon_{ini}, \varepsilon_{fin}, d$	0.9, 10^{-5}, 0.4
	No. of episodes	E	100
QLC	Alpha	α	0.5
	Gamma	γ	0.9
	Initial, final epsilon	$\varepsilon_{ini}, \varepsilon_{fin}$	0.9, 10^{-5}
	No. of episodes	E	200

pools or NIA groups. In this work, we consider six scenarios, where in Scenario 1, $N = 2$ IAs share $M = 1$ resource pool; in Scenario 2, $N = 4$ IAs share $M = 2$ pools and so on. Therefore, in Scenario 6, $N = 12$ IAs share $M = 6$ pools. Each resource pool, v_j, is initially configured to 20 CPUs, while all VNFs are initially assigned $n_0 = 1$ CPU.

Load generation. We simulate "loading" the NS using UE arrivals in time, *i.e.*, UEs arriving at the NS dynamically per unit time. Dynamic UE arrivals (or the service request rate) are modeled by a Poisson process, where $\Lambda_{in}(t)$ is the aggregated arrival rate of UEs from a population U at time instant t. $\bar{\theta}$ is the mean service duration of a UE. Over an episode duration $T = 10^4$s, $\Lambda_{in}(t)$ reflects a smooth increase and decrease of UE arrivals, emulating a real-life peak hour traffic scenario [16]. Therefore, $\Lambda_{in}(t)$ represents the NS incoming load at t. To introduce further randomness and complexity in the load generation, each UE contributes the load $\mu(t)$ from a normal distribution to the corresponding $u_{ij}(t)$. Table I summarizes the system and load configuration.

Algorithm configuration. The hyper-parameters of the distributed RL algorithms, **QLC** and **DQNC**, are tuned according to Table II, and they are trained for a total of E episodes. The specific algorithmic aspects are out of scope in this article and the reader is directed to [18] for precise details. We further implement a reactive, distributed, threshold-based auto-scaling mechanism, **THR**, inspired by the inbuilt functionality of OSM. **THR** scales up each VNF when u_{ij} exceeds a high threshold u_H and scales down when it is less than a low threshold u_L. Finally, we consider two benchmarks for the evaluation. The centralized Mixed Integer Optimization (**MIO**) formulation is the upper bound, where, at a given time interval, it computes the optimal CPU allocation for the given incoming load. Finally, the lower bound of the performance is a "no automation" scenario or **NO_AUT** that never scales CPUs of VNFs according to the incoming load and serves the maximum no. of UEs possible with the initial CPU configuration.

B. Evaluation & Discussion

Performance. The performance of an algorithm is evaluated based on the no. of UEs served by the NS per second, denoted as $\Lambda_{out}(t)$. Fig. 5 shows the performance comparison of the algorithms for four selected episodes. For a given incoming slice load $\Lambda_{in}(t)$, the optimum $\Lambda_{out}(t)$ is served by **MIO**, while **NO_AUT** performs the worst. At any episode, **THR** never reaches the optimum, as it attempts to allocate CPUs according to a greedy principle based on thresholds, thereby performing poorly throughout the episodes. Interestingly, both RL algorithms learn the environment incrementally, with **DQNC** reaching a performance at most 6% away from **MIO** at episode 36. Here, **QLC** still has not learned the optimum, but catches up at episode 61, where it is at most 5% away from the optimum. Finally, the performance of both algorithms is seen to be robust after convergence at an arbitrary episode 80.

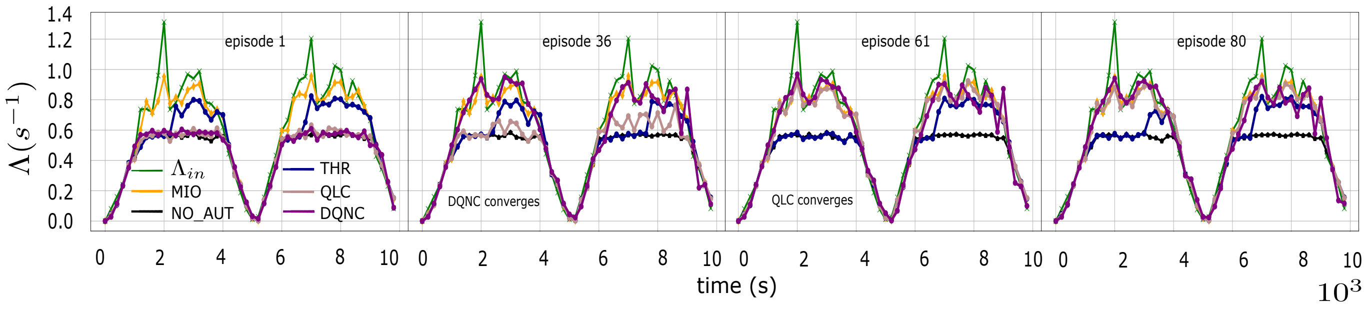
TABLE III: Average number of conflicts per episode after convergence

Scenario	# of IAs N	# of shared resource pools M	QLC	DQNC	THR
1	2	1	11.05	21.7	2
2	4	2	38.2	28.85	4
3	6	3	53.6	40.33	6
4	8	4	19.75	36.2	8
5	10	5	108.05	60	10
6	12	6	127	73	12

Conflicts. Table III reports the average no. of conflicts in an episode after convergence for all algorithms. In general, the average no. of conflicts increases with the no. of IAs. In 66% of the scenarios, **DQNC** is able to avoid more conflicts than **QLC**. Additionally, **THR** exhibits the least number of conflicts, however, its performance is the worst compared to distributed RL, as the scaling actions that are decided based on the thresholds are eventually sub-optimal.

Convergence time of distributed RL. Fig. 6 depicts the convergence time in terms of episodes for **QLC** and **DQNC**. We observe that as the no. of IAs in the scenario increases, both algorithms need more episodes for convergence. Moreover, owing to its tabular update mechanism, **QLC** takes longer or more episodes to converge than **DQNC**, which is a function approximation approach.

Discussion. Our evaluation and the architectural mapping in Sec. III confirm that although distributed AI brings benefits and avoids the pitfalls of a centralized approach, it still suffers from some non-negligible issues. First, both centralized and distributed approaches incur increased signaling with the no. of VNFs. Although the signaling grows linearly in both approaches (Sec III), the slope of increase is higher for distributed AI, due to NIA cooperation. Further, distributed AI requires additional time to converge in their learning, in contrast to centralized **MIO**. Finally, conflicts are inherent to a distributed paradigm. Ideally, after convergence, distributed AI should learn to take scaling actions that ultimately avoid conflicts and exhibit optimal system performance. In practice, however, as evidenced by our evaluation, conflicts are not eliminated. We remark that when signaling and convergence time are not critical, distributing AI in certain use cases could bring significant benefits. For example, in an edge computing factory automation scenario [3], distributed AI would avoid the latency incurred in the centralized, leverage compute resources at the edge instead of relying on a central cloud server and readily adapt to changes in the input data-set. Hence, we argue that for efficient management of 6G networks, one



(a) Performance $\Lambda_{out}(t)$ in s^{-1} of scenario 5 where $N = 10, M = 5$

Fig. 5: Incoming service requests to the NS $\Lambda_{in}(t)$ and corresponding load served $\Lambda_{out}(t)$ in s^{-1}

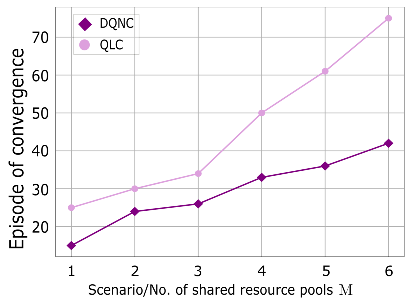


Fig. 6: Episode of convergence for all scenarios

approach cannot be considered "better" than the other. Rather, AI should be distributed depending on the context and the management use case. Moreover, to understand the effort needed to implement distributed AI in 6G, the analysis of the architectural impact may also be generalized to other use cases, apart from the specific auto-scaling one examined in this paper.

V. Conclusions

In this work, we investigated the overall effort needed to distribute AI in 6G. To this end, we analyzed the performance of distributed AI and how the existing 5G architecture should be enhanced to support it in 6G. We consider a relevant beyond 5G use case, auto-scaling resources in a network slice, and evaluate two RL algorithms based on Q-Learning and Deep Q-Networks, **QLC** and **DQNC** respectively. Via extensive simulation scenarios, we compared the performance of distributed AI to a centralized optimum **MIO** approach. Our findings show that the performance of **QLC** and **DQNC** is only 5% and 6% away from the optimum respectively. We observed distributed AI would be readily supported in 6G, with minor enhancements to 5GS data management (for inter-agent cooperation). Moreover, we caution the reader to few obvious issues in distributed AI – increased signaling with the no. of managed VNFs, conflicts not being completely eliminated and the convergence time being non-negligible. Despite these shortcomings, distributed AI has clear potential, as it brings dynamic decision-making and low complexity in processing data, making it an interesting 6G research stream.

References

[1] V. Passas *et al.*, "Artificial Intelligence for Network Function Autoscaling in a Cloud-native 5G Network," *Computers and Electrical Engineering*, vol. 103, p. 108327, 2022.

[2] E. Pateromichelakis *et al.*, "End-to-end Data Analytics Framework for 5G Architecture," *IEEE Access*, vol. 7, pp. 40295–40312, 2019.

[3] J. S. Walia *et al.*, "A Virtualization Infrastructure Cost Model for 5G Network Slice Provisioning in a Smart Factory," *Journal of Sensor and Actuator Networks*, vol. 10, no. 3, p. 51, 2021.

[4] X. Liu, J. Yu, Y. Liu, Y. Gao, T. Mahmoodi, S. Lambotharan, and D. H. K. Tsang, "Distributed intelligence in wireless networks," 2022.

[5] R. Cziva *et al.*, "Dynamic, Latency-optimal VNF Placement at the Network Edge," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 693–701, IEEE, 2018.

[6] "Zero-touch network and Service Management (ZSM); Means of Automation," Group Report ETSI GR ZSM 005 V1.1.1, ETSI, 2020.

[7] "Telecommunication management; Management concept, architecture and requirements for mobile networks that include virtualized network functions (Release 17)," Standard 3GPP TS 28.500, 3rd Generation Partnership Project, 2022.

[8] "Management and Orchestration; Architecture Framework (Release 17)," Standard 3GPP TR 28.533, 3rd Generation Partnership Project, 2022.

[9] "Network Functions Virtualisation (NFV); Management and Orchestration," Group Specification ETSI GS NFV-MAN 001 V1.1.1, ETSI, 2014.

[10] D. M. Gutierrez-Estevez *et al.*, "Artificial Intelligence for Elastic Management and Orchestration of 5G Networks," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 134–141, 2019.

[11] M. Li *et al.*, "Slicing-Based Artificial Intelligence Service Provisioning on the Network Edge: Balancing AI Service Performance and Resource Consumption of Data Management," *IEEE Vehicular Technology Magazine*, vol. 16, no. 4, pp. 16–26, 2021.

[12] S. Majumdar, R. Trivisonno, and G. Carle, "Scalability of Distributed Intelligence Architecture for 6G Network Automation," in *ICC 2022 - IEEE International Conference on Communications*, pp. 2321–2326, 2022.

[13] H. Chergui *et al.*, "Zero-Touch AI-Driven Distributed Management for Energy-Efficient 6G Massive Network Slicing," *IEEE Network*, vol. 35, no. 6, pp. 43–49, 2021.

[14] Y. Wang *et al.*, "Network Management and Orchestration using Artificial Intelligence: Overview of ETSI ENI," *IEEE communications standards magazine*, vol. 2, no. 4, pp. 58–65, 2018.

[15] A. Banchs *et al.*, "A 5G Mobile Network Architecture to Support Vertical Industries," *IEEE Communications Magazine*, vol. 57, no. 12, pp. 38–44, 2019.

[16] D. Bega *et al.*, "DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 280–288, IEEE, 2019.

[17] S. Majumdar, R. Trivisonno, and G. Carle, "A Distributed Intelligence Architecture for B5G Network Automation," 2021.

[18] S. Majumdar, L. Goratti, R. Trivisonno, and G. Carle, "Improving Scalability of 6G Network Automation with Distributed Deep Q-Networks." in press, IEEE Globecom, 2022.

[19] "Open Source MANO (OSM) Information Model," 2016. online.

[20] S. Majumdar, R. Trivisonno, and G. Carle, "Understanding Exploration and Exploitation of Q-Learning Agents in B5G Network Management," in *2021 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2021.

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[22] V. Mnih *et al.*, "Human-Level Control through Deep Reinforcement Learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.