A Deep-Q Learning Approach to Mobile Operator Collaboration

Athanasios Karapantelakis and Elena Fersman

Abstract: **Next-generation mobile connectivity services include a large number of devices distributed across vast geographical areas. Mobile network operators will need to collaborate to fulfill service requirements at scale. Existing approaches to multi-operator services assume already-established collaborations to fulfill customer service demand with specific quality of service (QoS). In this paper, we propose an agent-based architecture, where establishment of collaboration for a given connectivity service is done proactively, given predictions about future service demand. We build a simulation environment and evaluate our approach with a number of scenarios and in context of a real-world use case, and compare it with existing collaboration approaches. Results show that by learning how to adapt their collaboration strategy, operators can fulfill a greater part of the service requirements than by providing the service independently, or through pre-established, intangible service level agreements.**

Index Terms: **Agent-based architectures, deep reinforcement learning, mobile networks, 5G, 6G.**

I. INTRODUCTION

MOBILE communication technologies are continuously evolving, driven by new types of services and demand for higher data rates. To address these requirements, fifth generation mobile networks (5G) allocated and optimized use of higher frequency bands, and proposed a redesign of the core and radio access network [1].

Even though 5G is still in its later development stages and in early deployment, rapid development of data-driven processes for automation, such as machine learning (ML) and artificial intelligence (AI), has already instigated discourse on the vision and requirements of next generation networks (6G) [2]. One key area of focus is ubiquitous connectivity, requiring collaboration of multiple mobile network providers [3], [4].

Roaming, a technology introduced in 2G networks, allows mobile subscribers to use another public land mobile network (PLMN) to access connectivity services when they are out of coverage of their home PLMN, providing a technical solution to the ubiquitous connectivity issue. However, the way roaming is currently supported limits its applicability in context of a multi-service communications network. First, roaming agreements are setup manually between operators, and have multiple years lifetime [5]. In addition, it is common among operators to use lower quality of service (QoS) for roaming mobile subscribers [6]. This practice makes use of roaming questionable for next-generation "mission critical" services, which have strict QoS requirements on the mobile network (e.g. low latency, guaranteed bitrate, high-availability, etc.) [7].

Beyond roaming technology, 5G has evolved the architecture of mobile networks to support concurrent operation of multiple services with different QoS requirements. Network slicing is a virtual networking architecture, which allows multiple services to operate in isolation from each other on the same mobile network infrastructure [8]. A service is supported by one or more network slices, which can be deployed accross multiple operators, in what is known as multi-domain orchestration. Several approaches exist in the literature ranging from research projects to commercial solutions [9]. A common denominator for these approaches is the rigid nature of collaborations that remain the same during the lifetime of a service.

In this paper, we propose an approach that uses machine learning to predict future service demand. Based on this prediction, we suggest the optimal operator collaboration scheme to fulfill the service demand. In this way, operator collaboration schemes can change dynamically within the service lifetime. From a user perspective, the service key performance indicators (KPIs) remain fulfilled. We demonstrate that this approach yields benefits to both operators and service consumers, over the state of the art of fixed collaboration structure and current business practice of isolated service provisioning.

Paper Outline: Section II reviews related works, while Section III describes the theoretical formulation of our approach. In Section IV we present a system based on concepts introduced in the previous section, and in Section V we evaluate it against the state of art using a real-world use case from the automotive industry. Section VI concludes with a recapitulation of the work done and future research directions.

II. RELATED WORK

A. Importance of Operator Collaboration for 5G Services

In the extensive literature produced in the past, several use cases and case studies on 5G services identified the importance of establishing collaborations between operators as prerequisite for scaling these services to fulfill requirements of their customers.

Specifically, in the automotive domain, collaboration between operators was identified as one of the barriers for the growth of vehicle remote operation (*teleoperation*), and automated platooning use cases [10]. In the example reported in [11], multiple operators in Japan, collaborated to successfully launch

Manuscript received May 4, 2020; revised October 8, 2020; approved for publication by Periklis Chatzimisios, Guest Editor, November 9, 2020.

The authors are with the Machine Design department, School of industrial engineering and management, Royal Institute of Technology (KTH) and Ericsson AB, email: {athkar, fersman}@kth.se.

A. Karapantelakis is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2020.000032

1229-2370/19/$10.00 © 2020 KICS

Creative Commons Attribution-NonCommercial (CC BY-NC).
This is an Open Access article distributed under the terms of Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided that the original work is properly cited.

a rich communications services (RCS) platform, used for enhanced mobile messaging services.

Several more examples that draw similar conclusions on the value of operator collaboration for service success and scalability exist from case studies in healthcare [12], smart cities [13], mobile payments [14], but also e-health and e-government applications [15].

B. Approaches to Multi-Operator Collaboration

In section I we referred to multi-domain network orchestration as one of the approaches that could be used for technical realization of operator collaboration. The fact that network slicing heavily relies on virtualization technologies, such as network function virtualization (NFV) and software defined networking (SDN), allows for rapid allocation of network resources to services. There exist several frameworks that group these resources to services, accross multiple operators [9]. However, the question of when and how should operators collaborate is not addressed by these frameworks, which already assume a fixed, oftentimes hierarchical resource allocation decision structure.

Some relevant literature for multi-operator collaboration is based on the architectural concept of network sharing. The third generation partnership project (3GPP) has proposed a network sharing architecture where many network operators share the same radio access network and optionally, nodes of the core network [16]. Based on this architecture, several research approaches proposed methods for managing these collaborations from establishment, to operation and eventual tear-down. Ghazzai et al. suggested a process for eliminating redundant base stations in a shared network while maintaining the QoS [17]. Shafwat et al. proposed a method for roaming between radio base stations of multiple operators in order to reduce overall power consumption [18]. Finally, Shah et al. described a negotiation protocol between mobile network operators for sharing of network resources within small-cell networks [19].

A common denominator of the aforementioned approaches is their reactive nature, meaning that the described solutions are triggered after an event is observed. The solutions are also focused on different domains than service fulfillment. Specifically [17] and [18] focused on energy savings, whereas [19] did not describe the reason for initiating the trade of network resources, but rather the trade protocol itself.

C. Deep Q-Learning as an Approach for Multi-Operator Collaboration

We have found a gap in the state of the art for determining which operators should collaborate based on predicted service demand. Specifically, reinforcement learning (RL) has been successfully used in the past for learning optimal collaboration strategies between groups of agents based on observations of an environment [20], [21]. RL is supported by a reward and penalty-like system that forces an agent to consider both immediate and long term benefits of its decisions. An agent takes actions in an environment, which is interpreted into a reward and a representation of the state, which are fed back into the agent [22].

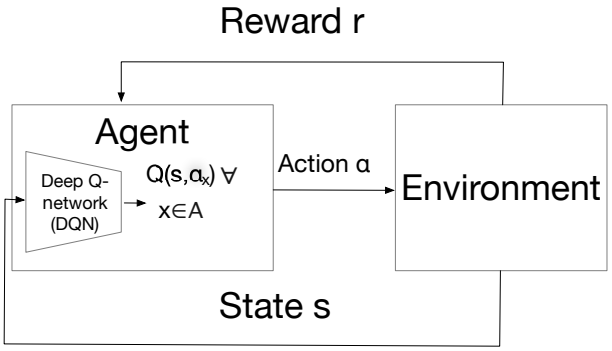


Fig. 1. Deep-Q Reinforcement Learning Loop. In every step, the model predicts the Q-values for all actions given a state. During model training, in some cases and according to an exploration-exploitation policy (such as epsilon-greedy policy described in subsection III.F), the action with highest Q-value is chosen (exploitation), while in others a random action is chosen to identify whether it yields better results (exploration).

In our case, we consider a software agent that makes collaboration decisions for operators in a geographical area (i.e. in an environment), based on observed patterns of service demand. Based on feedback received regarding how much of the service demand was actually fulfilled from the proposed collaboration (reward) over time, the agent gets better at predicting the future service demand, and eventually establishes operator collaborations proactively to meet this demand.

In RL-terms, the goal of the agent is to choose the best action (set of collaborating operators in our case), out of a set of possible actions (i.e. out of all possible combinations of collaboration between the operators). Q-learning is a popular RL algorithm, wherein an agent tries to identify the optimal action selection policy that maximizes the expected total future reward *r*. Constituents of this selection policy are the "Q-Values". Each Q-Value captures an expected future total reward that an agent in state *s* can receive by executing an action α. The goal is therefore to find the Q-Value with the maximum expected total future reward.

In classical Q-learning, a two-dimensional matrix of Q-values is created over time. The rows are possible actions, whereas the columns are states. Once complete, an agent in any state will be able to choose the action with the highest Q-Value. In context of our study, Q-learning has certain limitations:

- If the state space is too large, the learning process becomes inefficient.
- It is difficult to transfer knowledge between environments to reduce learning time, unless the environments are very similar.

For the above reasons, we opted to use Deep-Q RL. Using this approach, a neural network is trained in every step of the RL algorithm, instead of using a matrix [23]. This neural network is also able to make better predictions about the service demand in the future, as it is trained with more data. Fig. 1 illustrates a Deep-Q RL loop.

Based on our evaluation (see section V), the Deep-Q neural network (DQN) we trained, can support up to 40 operators in the same geographical area. Additionally and given an already-trained DQN, transfer learning can be used to reduce

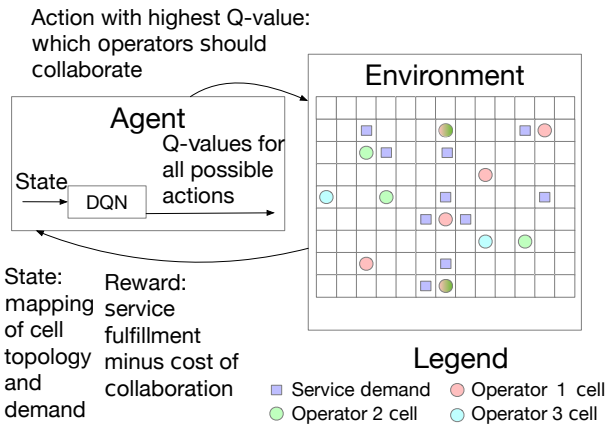


Fig. 2. Deep-Q RL loop in context of this study.

learning time when deploying the Deep-Q algorithm to another environment.

III. OVERVIEW OF APPROACH

A. Introduction

In this section, we describe the RL loop in greater detail. We start by defining key parameters of the environment and action space. We continue by presenting the algorithm for calculating the reward and describing the serialization of the environment state. Both serialized state and reward are sent to the agent in response to an action. We continue to describe the Deep-Q learning algorithm and its parameters and conclude by outlining our assumptions and delimitations of our approach.

B. Environment Parameters

Fig. 2 illustrates the Deep-Q RL loop used for training a DQN to select the optimal collaboration strategy.

The environment is a rectangular geographical area split in a number of smaller, equally sized rectangles, which we refer to as "blocks". Blocks may contain a cell site providing cellular service access to mobile devices in the area (also known as User Equipment, or UE). Cell sites consist of one or more cells that could belong to different operators. In reality, it is not uncommon for multiple operators to share the same cell site location (for example share the same radio tower and sometimes even the cabinet underneath). A cell is parameterized by its owner identifier (i.e. an operator, which has a unique global identifier[1]), throughput and bandwidth. Throughput is the actual amount of data traffic that is transferred through a cell, from and towards mobile devices attached to this cell and during the course of a single iteration. Bandwidth is the maximum amount of data traffic that can be transferred through a cell. Cells are stationary in the environment.

An environment also contains mobile devices, which may move between blocks from one iteration of the simulation to the next. Mobile devices generate throughput, provided that they are attached to a cell. In addition, they have a service identifier that identifies whether they belong to a service with QoS requirements or they are best effort type of devices, generating background mobile traffic.

Mathematically, an environment *E* consisting of *i* rows and *j* columns of blocks $b_{i,j}$ can be described as follows:

$$E_{i,j} = \{b_{1,1}, b_{1,2}, ..., b_{1,j}, b_{2,1}, ..., bi, j\} : i, j \in \mathbb{N}. \quad (1)$$

Each block $b_{x,y}$ with *x*, $y \in \mathbb{N}$ belonging to environment *E* is parameterized by zero or more cells:

$$b_{x,y} = \{c_1, ...c_n\} : |b_{x,y}| \geq 0. \quad (2)$$

Each cell has static parameters including bandwidth (bw) and its operator owner. In addition a cell is parameterized by the number of actual mobile subscribers connected to it; a parameter specific to a state. As UE is mobile, this number can change from iteration to iteration. Assuming a state s_m belonging to state space $S = \{s_1, ...sk\}$, a cell c_z belonging to a block $b_{x,y}$ can be parameterized as follows:

$$c_z = \{opid_{c_z}, bw_{c_z}, UE\_list_{c_z} | s_m\}. \quad (3)$$

In the environment, UE is represented by a unique identifier[2], the service they belong to (QoS-sensitive use-case or background traffic generated by other services), the amount of throughput they generate and their location (which block they are currently located at). Every UE_i belonging to the set of UE in the environment has the following parameters:

$$UE_i = \{service_{UE_i}, id_{UE_i}, thr_{UE_i} | s_m, b_{x,y} | s_m\}. \quad (4)$$

Every use case also has a service level agreement (SLA), which defines the QoS requirements about the service offered. In the context of this study, SLA defines a level of guaranteed throughput, thr_{SLA}, that all mobile devices belonging to the use case require.

C. Action Space

In context of this study, an action describes the agent's decision that defines how operators should collaborate. The decision may include any combination of operators or even individual operators in cases where the agent decides that collaboration will not yield higher future reward.

The size of the action space depends on the number of operators present in the environment. For example, assuming the setup in Fig. 2, size of action space is 7. In general, the size of action space increases exponentially by $(2^N - 1)$, where N is the number of operators in the environment. This includes the cases where operators provide the service without any collaboration.

D. Calculation of Reward

In order to calculate the reward for the agent's action, we introduce the "service demand" metric. For a given state $s_x \in S$, service demand is fraction of the served throughput of all QoS-aware service UE ("service UE"), by the total throughput service UE could generate. For the notation used in the algorithm, the reader may refer to subsection III.B.

[1]One example of such a unique global identifier is Public Land Mobile Network Identifier - PLMN id, standardized in [24].

[2]In a real environment, a typical UE identifier can be the globally-unique International Mobile Subscriber Identity, or IMSI, however in the simulation, we assign unique integers

Algorithm 1 Calculate reward for state $s_x \in S$

```
Initialize list of all UE UE_list = {UE_1, ...UE_N}
Initialize total_thr, served_thr to 0
for block b_{x,y} in environment E_{i,j} do
  if block b_{x,y} contains cells then
    for cell c_z in block b_{x,y} do
      Initialize cell_thr to 0
      for UE_k in UE_listc_z|s_x do
        cell_thr = cell_thr + thr_{UE_k}
        if service_{UE_k} belongs to a use case and does not
        generate background traffic then
          if (cell_trh + thr_{UE_k}) ≤ bw_{c_z} then
            served_thr = served_thr + thr_{UE_k}
          end if
          total_thr = total_thr + thr_{UE_k}
          Remove UE_k from UE_list
        end if
      end for
    end for
  end if
end for
for UE_k in UE_list do
  if service_{UE_k} belongs to a use case and does not generate
  background traffic then
    total_thr = total_thr + thr_{UE_k}
  end if
end for
reward = served_thr / total_thr − cost_collab
return reward
```

Algorithm 1 iterates all blocks in the environment and, for those blocks that contain cells, it calculates the amount of throughput generated from service UE. For the purposes of this study, we consider that all service UE is generating throughput equal to that described in the SLA (thr_{SLA} - see subsection III.B for its definition). This throughput is stored as "served throughput". If the data traffic of all UE attached to a cell exceeds the bandwidth of the cell or if the UE is not covered by the SLA ("background UE"), then the throughput created by these UE is not stored as served throughput. In any case, all data traffic generated from all service UE, is stored as "total throughput". This metric accounts for both UE attached to cells or UE that are not attached to a cell. This may be the case when UE is in blocks where there is no coverage from the operator or operators currently providing the service.

As algorithm 1 illustrates, the reward function is the ratio of "served throughput" divided by "total throughput", minus a cost of collaboration. The reason for adding a cost of collaboration is partly due to the fact that otherwise, all operators would always collaborate between them to cover maximum service demand as possible. Another part of introducing cost is due to the impact collaborations have on the revenue of the operators, as revenue for service will be shared among multiple operators.

In context of this paper, we consider the cost to be minimal if every operator covers the same amount of service demand. For example, assuming two collaborating operators, if each of the two operators covered 50% of service demand there would be no cost. Otherwise, the cost would increase as percentages of coverage divergence. An exemplary formula that we use for our own simulation (see section IV) is illustrated in equation (5).

$$cost\_collab = w_{collab} \cdot \frac{SD_{max} - SD_{min}}{SD_{max} + SD_{min}}. \tag{5}$$

In (5), we assume that SD_{max} is the largest service fulfillment amount among participating operators, whereas SD_{min} is the smallest. These service fulfillment amounts can be calculated using the information described in equations (1) to (4) in subsection III.B. All SD values are normalized from 0 to 1. The w_{collab} is a discount factor that, based on the use case, can be set to values closer to 1 or closer to 0, depending on how much gravitas the collaboration coefficient has in the calculation of the reward function.

Using this reward function design, we favour collaborations with, to the extent possible, equal operator participation. This type of collaborations will be more likely to be accepted by operators in reality, as it provides for a clearer cost structure than uneven participation.

E. State Serialization

In every iteration, the environment's status is serialized as a state and communicated to the agent. The size of the state space depends on the number of blocks. In the example illustrated in Fig. 2, there exist 117 blocks.

For each block, demand is calculated in terms of throughput. Specifically, in each block we aggregate throughput of all UE that belong to the QoS aware service and are located in that block. We also make the assumption that information about the location of the cells as well as the operator that owns the cells are known to the agent a priori, as this is static information that is unaffected by the different states in the simulation.

F. DQN Algorithm

The main challenge of Deep Q-learning is maintaining policy stability [3]. The first issue is the non-stationarity of target values. In supervised learning, the target (i.e. the output part of training data - Q values in our case) remains static during the training process. However, in RL, target changes in every iteration, as DQN's performance improves. Another issue is the close correlation of subsequent states and actions, which could result in DQN not only forgetting earlier experiences[4], but to risk overfitting as new experiences are closely related to each other.

Mnih et al. proposed experience replay and target neural network in order to increase stability of DQN, by addressing the two aforementioned issues [23]. Experience replay is simply a buffer that stores past experiences. It also creates mini-batches of a random number of stored experiences used in every iteration to train the DQN. A target network stores fixed Q-values. The target network is not trained, but is periodically updated after some iterations (usually thousands), by copying weights of the DQN to the target network. The DQN algorithm we use in this

[3]Policy is the function the agent uses to choose next action. In context of this study, the policy output is realised by a forward-pass of the DQN.

[4]An experience is a training data point for DQN, and includes the current state, action taken, as well as the reward and new state provided by the environment (see Fig. 1).

Algorithm 2 DQN for operator collaboration

Initialize replay buffer to M $\geq$ minibatch sample size N
Initialize value network (DQN) Q_θ, target value network $Q_{\theta'}$ with random weights
for iteration = 1 to K **do**
  Select action $\alpha_t = \begin{cases} max(Q_t(\alpha)) \; with \; P = 1 - \epsilon \\ random \; action(\alpha]) \; with \; P = \epsilon \end{cases}$
  Execute α_t and observe reward r_t, state s_{t+1}
  Store experience $(s_t, \alpha_t, r_t, s_{t+1})$ in replay buffer
  if Replay buffer has enough experiences AND for every T_{train} **then**
    Sample random minibatch of N transitions from replay buffer
    for every (s_i, a_i, r_i, s'_i) in N **do**
      Set target $y_i = \begin{cases} r_i \; if \; s'_i \; is \; terminal \\ r_i + \gamma \cdot max_{\alpha'} Q_{\theta'}(s'_i, \alpha') \end{cases}$
    end for
    Train Q_θ with loss function $(y - Q_\theta(s, \alpha))^2$
    if loss $\leq$ threshold **then**
      return Q_θ
    end if
  end if
  for every T_{update} **do**
    Update weights of $Q_{\theta'}$ with those of Q_θ: $\theta' \leftarrow \theta$
  end for
end for
return Q_θ

study is based on a DQN with target network and experience replay and is illustrated in Algorithm 2.

In every iteration *t*, a new action α is chosen in accordance to an epsilon-greedy policy. The algorithm either chooses a random action α from a set of actions A with probability 1 - ϵ, or does a forward pass of the DQN, given a state s_t in order to select a new action α, with probability ϵ. In this way, exploration is balanced with exploitation, in order to avoid situations where an under-trained DQN provides bad predictions.

The action is sent to the environment, which calculates a reward and serializes the next state s_{t+1} before sending them back to the agent (see section III.B for description on how the reward and state are created). The experience, meaning the quadruplet (s_t, a_t, r_t, s_{t+1}), is stored in the replay buffer. Subsequently a minibatch is pulled from the replay buffer. The target network is used to calculate the targets for every state in the minibatch. The reader should note that the size of the replay buffer is at least equal to the minibatch sample size, and that the replay buffer is circular, i.e. older experiences are removed to make room for newer experiences.

The DQN is trained using data from the minibatch and a minibatch gradient descent algorithm, but with a loss function that involves the calculated target values from the target network. If the loss function reports a loss value below a threshold, the algorithm converges and returns the DQN, otherwise the process continues until iterations reach a ceiling *K*. At this point the DQN is returned. The reader should note that proper training of DQN is done by monitoring the loss function and that *K* is used as a safeguard for the training to not continue ad infinitum. In practice, this means that a user of the system would need to set *K* to a large value that would allow for convergence of DQN with acceptable loss prior to iteration *K* being reached.

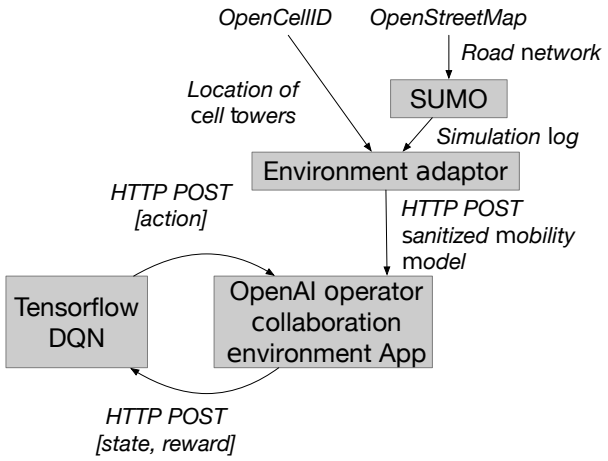


Fig. 3. Simulation environment.

G. Delimitations and assumptions

In this study, we make assumptions with regards to the environment. These assumptions were made in order to reduce complexity of the simulation and therefore simulation time. Specifically, we assume that all cells use similar radio access technology and have the same coverage as well as throughput capacity.

Furthermore, when it comes to traffic being generated from the mobile devices, we assume that there exist two types: background traffic and use-case traffic. The latter is the data traffic that is governed by an SLA and is used in the reward calculation, however the former impacts the total available bandwidth of the cells.

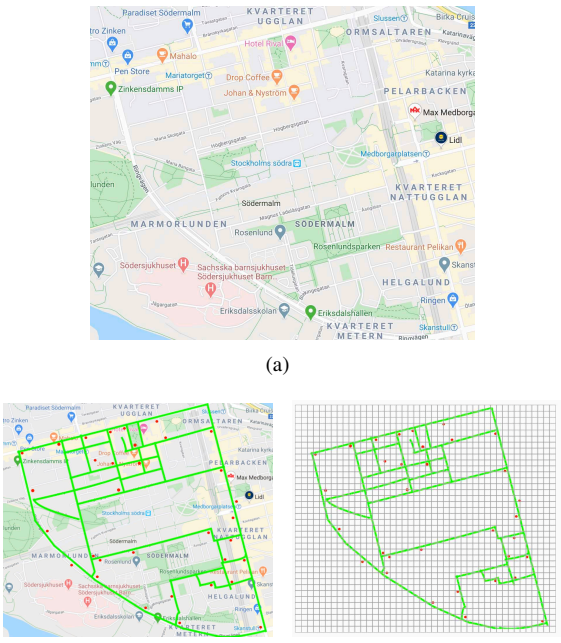
Also, the impact of signal fading to throughput (e.g. due to multi-path propagation, weather effects, obstacles, etc.) is not taken under account. We also assume that the cells provide the same coverage, which can be parameterized for every use-case. In the future we plan to support heterogeneous radio access technologies that have different capacity and range characteristics, but also take into account fading due to geographical morphology.

IV. SIMULATION ENVIRONMENT

A. Architecture and System Description

Fig. 3 illustrates the architecture of the simulation environment developed for this study. In the context of this study, we assume that in the environment, service demand is generated by vehicles in road networks. Therefore, every vehicle is a UE.

In order to simulate road traffic, we use SUMO [25], a vehicle traffic simulator. SUMO uses data from OpenStreetMap [26] in order to realistically model vehicle traffic on a real road network. Therefore the first task is to use SUMO in order to simulate mobile traffic. As vehicle traffic is not affected by



(a)

(b) (c)

Fig. 4. Method for augmenting road and cell tower location data: (a) Exemplary scenario map, (b) extracted SUMO road network and cell location, and (c) environment definition as a 38x44 block matrix.

radio cell tower coverage, we run the simulator offline. In order to find the location of the radio towers in the same area from where mobility is simulated, we use OpenCellID [27], a publicly available database. Fig. 4 illustrates the process which is used by the environment adaptor to create the environment model. First OpenStreetMap is used to import a map in SUMO, which in turn extracts information about the road network (see Fig. 4(b)). Then, OpenCellID is also used to designate the location of the cells in the area where the road network was extracted (see also Fig. 4(b)).

Subsequently, SUMO is executed on the extracted road network and produces a log file with the number of vehicles present on every block across the road network for every simulation step.

This mobility data together with the location of the cell towers are imported to the Environment Adaptor which uses the information to create the environment description. The demand is generated by multiplying the amount of throughput every vehicle generates with the number of vehicles. The traffic generated by every vehicle depends on the type of use case. The Environment Adaptor sends the environment description to OpenAI Gym [28], a framework that we used to create the environment-agent RL loop. A custom environment application was developed to compute the reward at each step of RL loop, and send it together with the updated environment state to the agent. The agent itself is also implemented in OpenAI, and trains a DQN implemented as a convolutional neural network using tensorflow machine learning framework [29].

B. Use Case and Scenarios Description

For purposes of this paper, we define a vehicle *teleoperation* use-case, which we previously described in [30]. In this use-case, self-driving vehicles are remotely monitored by a vehicle controller who can intervene if necessary (for example in case a vehicle encounters an obstacle that the onboard system cannot overcome). Table 1 illustrates the parameters of the DQN algorithm for the teleoperation use case. Section IV.D motivates the architectural choice of number and type of layers of the DQN.

Table 1. Parameters of DQN algorithm for the teleoperation use case

Parameter Name	Value
Training policy	ϵ-greedy (0.1)
Discount factor γ	0.99
Replay buffer size M	1000
Target network update T_{update}	800 iterations
DQN training T_{train}	100 iterations
Minibatch size N	32
Cost of collaboration w_{cost}	0.5
Type of DQN, target network	Multi-layer perceptron
Number of hidden layers	4
Type of layers	Convolutional 2D
Activation function	ReLu

Table 2. Parameters of environment

Parameter name	Value
Uplink per UE	11.3 Mbps
Downlink per UE	0.3 Mbps
Number of UE	67 (Urban), 35 (Rural), 18 (Highway)
Cell bandwidth	50 Mbps Uplink, 100 Mbps downlink
Number of operators	4 (Urban), 5 (Rural), 4 (Highway)

In addition to the DQN parameters, we defined several parameters of the RL environment (see Section III.B). Table 2 shows some of these parameters. As far as UE-generated data traffic characterization is concerned, the use case has vehicles sending high quality video to the controller but also receiving vehicle control traffic and some “keepalive" messages. The uplink data rate (i.e. data traffic transmitted from UE) is therefore much larger than the downlink (data transmitted to UE). For each cell, we assumed use of Long Term Evolution (LTE) technology and 100 MHz spectrum, with carrier aggregation (CA) of 5 carrier components (CC) of 20 MHz each, each channel having 100 Mbps downlink and 50 Mbps uplink. We also assume that the bandwidth of the cell is not exclusively allocated to the teleoperation service.

On the contrary, only one dedicated channel is used, while other channels are allocated to other types of services (e.g. mobile broadband, mission critical communication, machine-to-machine communication, etc.). These services are considered background traffic.

We modeled three different scenarios, an urban, a rural and a highway scenario, from different geographical locations. The purpose behind choosing these scenarios was their inherently different characteristics, as every scenario differs in terms of vehicle number and distribution in the environment, total volume of background traffic, as well as vehicle travel speed (for example in the highway scenario the vehicles travel much faster than in the urban scenario).

For the purposes of our simulation we consider as teleoperated vehicles buses and trucks. We have used Google Maps to identify the frequency and traffic schedules of the buses and information from an automotive partner to identify the

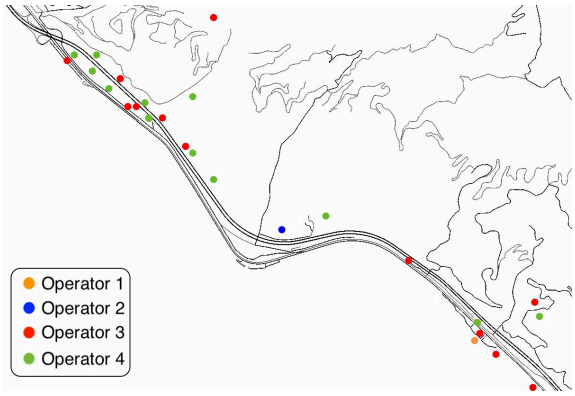


Fig. 5. Highway scenario, Ventura freeway and pacific coast highway outside of Los Angeles, California, United States

Fig. 6. Highway scenario, source map from OpenStreetMaps

Fig. 7. Rural scenario, Pequannock Township, New York State, United States

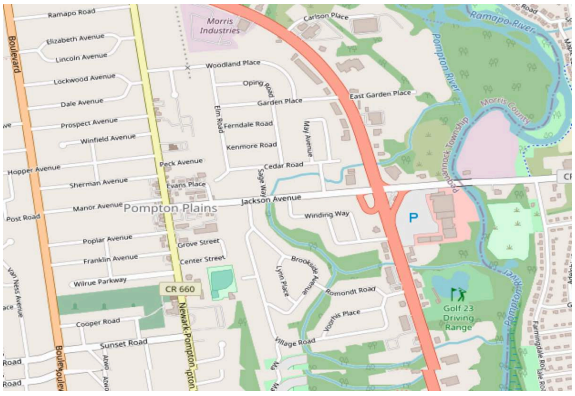


Fig. 8. Rural scenario, source map from OpenStreetMaps

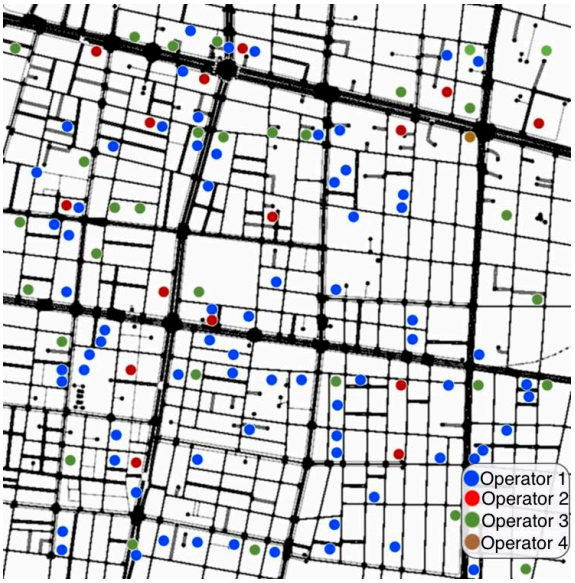


Fig. 9. Urban scenario, Taito City (part), Tokyo, Japan

Fig. 10. Urban scenario, source map from OpenStreetMaps

traffic schedules of trucks. Using OpenCellID, we pinpointed the geographical position of radio base stations in the three different environments. OpenCellID also contained information regarding the cell owner, in terms of home network identifier (HNI) number. We only chose to place those radio base stations with LTE technology (also known as e-Node B's or eNBs)

Table 3. Parameters of simulation

Parameter name	Value
SUMO total steps	21600
SUMO steps per DQN iteration	3600
Times of DQN forward-pass	6
Sampling rate for evaluation (“simulation cycle")	108

and not previous generations, as they were the only ones that could fulfill the uplink data traffic requirements presented in the previous paragraph.

Table 2 shows the parameters set for these scenarios. When it comes to number of UE, the requirement towards SUMO was that at any point in time the same number of vehicles specified above should run in the road network. Therefore, for every vehicle that leaves the simulation space, another one should enter the circuit. Regardless of the constant number, the vehicles themselves could be located in different places of the map at different iterations, as long as those places were covered by roads. In the future we plan to support a variable number of vehicles, for example, based on the time of day. Figs. 5 to 10 illustrate the cell topology, cell owner, road network and location of chosen scenarios.

C. Simulation Parameters

Table 3 illustrates the parameters used for conducting our experiments detailed in section V.

One aspect of the simulation we considered is the relation of steps in the SUMO simulator to the steps in the RL loop. Specifically, the SUMO simulator updates vehicle traffic every second (“step length"), however, this is too short for the RL algorithm to train and suggest a new action. If this was the case, then it would theoretically be possible for the collaborations between operators to change every second, which is not realistic. We therefore decided that a step in the RL loop would correspond to a simulation time of an hour, i.e. 3600 simulator steps. In real deployment scenarios, this step can be even larger, for example a week worth of measurements. For the purposes of this paper, we felt that an hour strikes a good compromise between a realistic scenario and simulation convergence time. Operators in many cases bill by the hour, and can create pricing plans they are already familiar with from mobile broadband connectivity services.

The mapping of one hour to one RL iteration raises another issue, regarding the training of the neural network: if the neural network is trained at a specific point in time every hour, then it is not going to learn about how the demand fluctuated in the environment during the hour. It will therefore be unable to make an accurate prediction about the future state. Our solution to the problem was to provide all 3600 data-points as reported state, i.e. input vectors to the neural network.

D. Simulator performance

In context of this study, we conducted some experiments to fine-tune the performance of the simulator, as the time spent in every step had an important effect on the overall convergence of the RL loop. Specifically and as Table 4 shows, most of the time was spent training the neural network, the authors tuned the hyper-parameters of the neural network to save as much training time as possible without affecting the precision of the model.

Table 4. Simulation Execution time for scenarios presented in Figs. 5–10

Scenario	SUMO traffic model	Neural network Training time per Step	Number of steps to converge	Total simulation time
Rural	1m 25s	1.2 s	5400	1hr 50m
Urban	4m 44s	1.2 s	12500	4hr 15m
Highway	2m 12s	1.2 s	8400	2hr 51m

Table 5. Deep-Q Model hyperparameter tuning: accuracy versus complexity

Number of layers (parameters per layer)	Accuracy	Average training time per step
4 Dense (9959)	0.86	5.42 s
5 Dense (12427)	0.89	6.32 s
6 Dense (15112)	0.88	8.02 s
4 Convolutional (6459)	0.84	1.2 s
5 Convolutional (9532)	0.86	1.96 s
6 Convolutional (12423)	0.87	3.12 s

The measurements were performed on a computer with 32GB of random-access memory (RAM), running Ubuntu Linux 16.04, with a graphics processing unit (GPU) of 8GB of GDDR5x memory and 1.6 GHz base frequency and a 4-core central processing unit (CPU) with 3.6GHz base frequency. All software components described in Fig. 3 were running in the same computer, therefore there were no network delays between the environment adapter and OpenAI. Tensorflow was running in accelerated (GPU) mode, making use of GPU vendor’s API.

Given a standard average time for training the neural network in every step and irrespective of the use case, the main differentiator between the use cases was the number of steps required for the neural network training to converge. As convergence we define the moment in the process of training the model, where further training does not increase its accuracy, but on the contrary may have a negative effect.

We observe that the total time needed to properly train the model is driven by the complexity of the use case. In Section V.C.2, we also described another way to reduce the simulation time using transfer learning.

The environment part of the RL loop is implemented in Python, using the OpenAI Gym API. It interfaces with another Python application, which uses Tensorflow Keras API to create and train a DQN. One of the early challenges we faced was the long training time for the DQN in every step of the RL algorithm. The original design of the neural network had four fully-connected layers (Dense in Tensorflow terminology), which were replaced by four Convolutional layers, which were not fully-connected and therefore computationally lighter.

As Table 5 illustrates, this change resulted in the greatest improvement in amount of time in relation to the model accuracy. The same figure illustrates that adding more layers of the same type did not affect the accuracy of the model by a significant amount. To measure accuracy we divide the number of correct predictions by the total number of predictions. We use this metric throughout this study as a measure of model performance.

V. EVALUATION

A. Overview of Experiments

In order to evaluate the proposed solution, we have designed a number of experiments.

The first experiment investigates the *performance* of the system by computing reward per step for the three use cases discussed in Subsetction IV.A. This is compared with state of the art and specifically operator-selfish (i.e. independent provisioning of services with no collaboration with another operator) and fixed collaboration (e.g. roaming) scenarios. For every scenario, we executed the simulation in SUMO for 21600 steps, with random vehicle starting positions. In order to plot performance, we added functionality to the agent to sample the sumo data points by batches of 108 and extract performance values by averaging the data points in those batches (see subsection V.B for an explanation of performance metric). We name this sampling activity "simulation cycle". For each simulation run, the RL DQN was executed 6 times, changing the collaboration of operators if necessary.

The second experiment examines the *scalability* of the proposed RL algorithm to support a greater number of vehicles and larger geographical areas, and to explore the limits of what can be supported.

Finally, the third experiment investigates the *portability* of the solution. Specifically we take advantage of a machine learning technique called transfer learning, and we show that the solution can be transferred to similar environments, wherein the algorithm can converge much faster, as it takes advantage of what has already been learned.

B. Performance Assessment

In this section, we describe the results of performance evaluation of our proposed solution and compare them against the state of art. As mentioned in section II, currently operators offer communications services on their own ("selfish" approach), or using pre-determined collaborations (either the traditional roaming approach, or in 5G, using a multi-vendor service orchestration approach). As *service performance* we define the ratio of fulfilled service demand versus total service demand (service demand is defined in subsection III.D). Service performance is quantified in a normalized scale from 0 to 1. Zero means no service demand fulfilled, whereas one means total service demand was fulfilled.

Before our experiments, we trained the a DQN model using Deep-Q RL approach presented in Section III.B, until it converged (i.e. the accuracy of the model plateaued - see also table 4). We then proceeded evaluate service demand for the three scenarios presented in section IV.B for three different types of collaboration models, which we also refer to as "strategies". Specifically, we evaluated against the following:

- A selfish strategy, wherein we consider that the communication service is delivered by one operator. For each scenario we select the operator that offers the most capacity and coverage.
- A fixed collaboration strategy, where we consider that the communication service is delivered by multiple operators, and vehicles can attach to a different operator in case of lack of coverage and/or capacity of their current operator. In case of our experiments and for every scenario, we chose the two operators providing the best combination of broader coverage and higher capacity in each scenario.
- A dynamic collaboration strategy, where, a pre-trained DQN constantly chooses the most suitable collaboration approach to maximize service fulfillment while ensuring fairness for the operator (in other terms, maximizing the reward function presented in Section III.B).

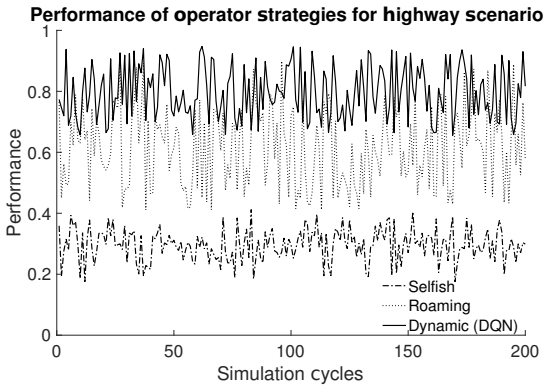


Fig. 11. Evaluation of operator strategies for the highway scenario.

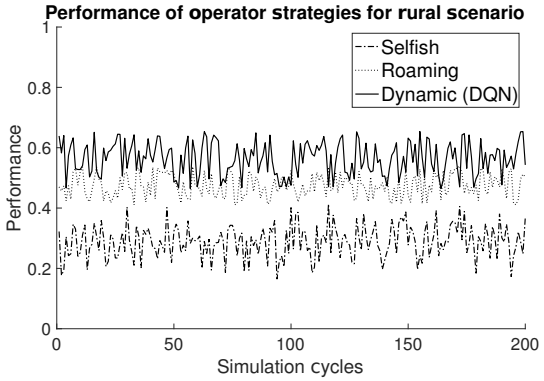


Fig. 12. Evaluation of operator strategies for the rural scenario.

Fig. 11 illustrates the comparison for the highway scenario. The Y-axis reports a normalized indication of the performance on a normalized scale of 0 to 1, whereas the X-axis is the step number.

The highway scenario is characterized by high dynamic vehicle traffic flows, but also has limited cellular coverage in certain locations. As an example, the large turn in the middle of the map is covered by only one operator (see Fig. 6). We observe that even though performance fluctuates during the simulations by 20% regardless of the strategy followed, the dynamic strategy generally works better. This is due to the predictability of the route in this scenario, as vehicles on highways such as the one simulated in this study have a predetermined route along the highway with few opportunities to change course. Therefore, the DQN network over time becomes better at predicting future demand of communications services.

Fig. 12 illustrates the evaluation of the operator strategies in the rural scenario. Contrary to the highway scenario, where

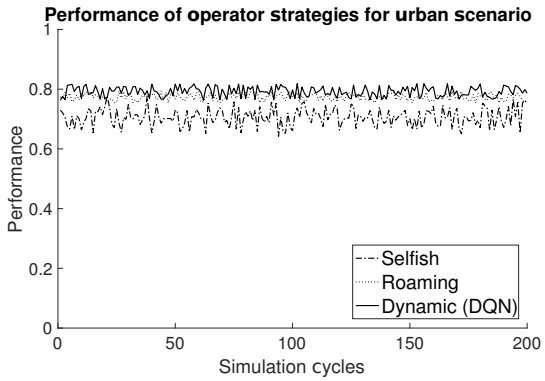


Fig. 13. Evaluation of operator strategies for the urban scenario.

Table 6. Average percentage increase using dynamic collaboration (DQN) approach over legacy approaches.

Scenario	Percentage increase using DQN
Urban, selfish	12.17 %
Urban, roaming	2.34 %
Highway, selfish	169.12 %
Highway, roaming	29.37 %
Rural, selfish	94.94 %
Rural, roaming	18.81 %

LTE coverage was poor in a few areas of the map, in this scenario, there exist more areas with no coverage, especially the residential areas towards the left of the map (see Fig. 7). This is reflected in the performance values, which are generally worse than those of the highway scenario. Another observation is that in this scenario, there is greater dispersion of operator cells in the map, each operator seemingly covering a different geographical area. In this scenario, the dynamic approach offers better performance - mainly in terms of enabling operators to collaboratively cover different areas of the map.

The urban scenario is illustrated in Fig. 13. This scenario is characterized by a dominant operator covering the largest area of the map with ample capacity. The dynamic strategy in this case provides marginal but still noticeable improvements over the roaming strategy.

Table 6 illustrates the average percentage increase that dynamic collaboration has over legacy approaches in all three scenarios. We observe a benefit from anywhere between 2% to 29% with regard to roaming scenario, and from 12% to 169% w.r.t. the selfish scenario.

C. Deployment in production networks

In this section, we evaluate the scalability and portability of our approach. This is important from the perspective of deploying this solution in production networks.

C.1 Scalability

From evaluating the simulation performance on the three use cases, we identified that traffic simulation is a small part of the overall simulation time - the rest is spent on training of the DQN (see table 4). While we investigated the role the configuration of the DQN in terms of number and type of hidden layers has on the model accuracy and training time, these investigations

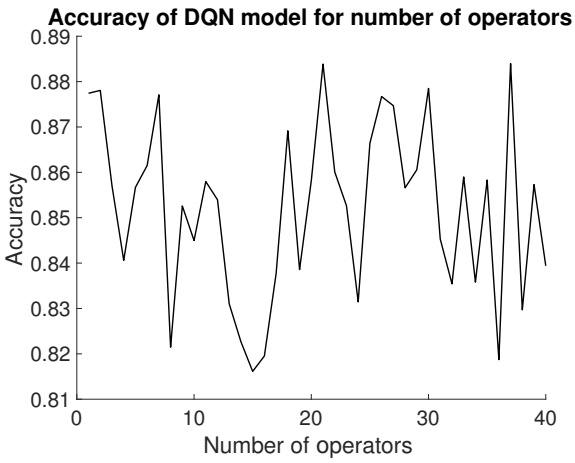


Fig. 14. Accuracy of model for different number of operators participating in the simulation scenario.

were limited to studying the use cases themselves, which were limited to a small number of mobile network operators. In reality, there can be situations where a much larger number of operators exists, especially considering the fact that many operators rent network infrastructure from others (mobile virtual network operators - MVNOs). The number of MVNOs globally increased 61% between 2010 and 2018, and currently there exist more than 1300 MVNOs worldwide [31].

Therefore, considering that the traffic simulation part is not the bottleneck, we investigate the impact the number of operators has on the accuracy of the DQN model. The number of operators also affects the action space which increases exponentially. For this experiment, we used the urban scenario as illustrated in Fig. 9, and we split the total 116 cells randomly and equally among hypothetical operators, starting from 2 up to 40. The range is considered to cover all potential cases in any area in the world based on the numbers presented in [31].

The experiments revealed small fluctuations in the accuracy of the model but no significant degradation (see Fig. 14). Therefore, the number of operators does not affect the performance of the proposed system, ergo, this solution could also be relevant for future potential collaborations between MVNO.

C.2 Portability

Another important aspect of the proposed approach, is its rapid potential deployment and use in production networks. One area to optimize to facilitate this, is the reduction of the DQN training time. As mentioned in Section IV.D, it is possible to apply transfer learning [5] in order to transfer the optimal collaboration strategies learned in one scenario to another, thus reducing the time needed for training.

For our experiments, we used the DQN trained in the rural scenario as baseline for training of the urban and highway scenarios. Table 7 shows the results of the experiment.

[5]In machine learning, transfer learning is the process of transferring knowledge gained via training of one model to another – the assumption being that the task of both models is related. It is very popular for Convolutional Neural Networks (CNNs)-based models, such as the one used in our solution.

Table 7. Simulation execution time for scenarios presented in section IV.B. Hidden layers of rural scenario are used as baseline.

Model	Accuracy	Training time
Urban	0.822	4h 11m
Rural	0.625	1h 35m
Highway	0.801	2h 49m
Urban & baseline	0.798	3h 04m
Highway & baseline	0.774	2h 10m

We observe a reduction training time for the urban scenario of 26% when using rural model as baseline, without significant loss in accuracy. In case of the highway scenario, the reduction is 23%.

When taking into account the large geographical areas mobile network operators cover today and the likelihood that the solution would need to be deployed hundreds of times in different geographical areas in reality, then the benefits mentioned above are compounded by the number of deployments and therefore have an even greater significance.

VI. CONCLUSION

In this paper, we have presented a solution based on deep-Q RL, for enhancing collaborations between operators to fulfill as much service demand as possible in a geographical area. In our experiments, we have shown a 2% to 169% improvement over the state of art of selfish service provisioning or pre-determined, static operator collaboration approaches. We have also shown that it is possible to scale the solution to large number of operators, which is relevant to a growing MVNO market. Finally we observed that as knowledge is accumulated from one environment, it can be transferred to another and lessen the time to deploy the solution anywhere from 23% to 26% .

In future work, we plan to introduce policies to the agents, as many operators have limitations with regards to which other operators they can collaborate with or not, but also have different agendas (e.g. prefer to serve larger amounts of mobile broadband mobile subscribers than enterprise customers). To this end, we plan to treat every operator as an agent with its own policy function and collaboration constraints, in a multi-agent RL loop. Finally, we also plan to attenuate the number of UE in the scenarios in order to simulate realistic traffic situations (e.g. heavy, light traffic, etc.).

REFERENCES

[1] M. S. *et al.*, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, pp. 1201–1221, June 2017.

[2] L. Zhang, Y. Liang, and D. Niyato, "6G visions: Mobile ultra-broadband, super internet-of-things, and artificial intelligence," *China Commun.*, vol. 16, pp. 1–14, Aug 2019.

[3] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Towards 6G networks: Use cases and technologies," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 55–61, Mar. 2020.

[4] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, Oct. 2019.

[5] OECD, "International Mobile Roaming Agreements," No. 223, 2013.

[6] BEREC, "BEREC Opinion on the functioning of the roaming market, as input to the Commission's evaluation," Tech. Rep. BoR (19) 101, body of European regulators for electronic communications (BEREC), June 2019.

[7] N. A. Mohammed, A. M. Mansoor, and R. B. Ahmad, "Mission-critical machine-type communication: An overview and perspectives towards 5G," *IEEE Access*, vol. 7, pp. 127198–127216, 2019.

[8] D. Soldani, "5G beyond radio access: A flatter sliced network," *Mondo Digitale*, vol. 17, pp. 1–20, Feb. 2018.

[9] R. Guerzoni *et al.*, "Analysis of end-to-end multi-domain management and orchestration frameworks for software defined infrastructures: An architectural survey," *Trans. Emerging Telecommun. Technol.*, vol. 28, pp. 1–19, Sept. 2016.

[10] A. Karapantelakis and J. Markendahl, "The role of mobile network operators in intelligent transport systems: Situation analysis, challenges and suggested approach," Regional ITS Conference, Los Angeles, Tech. Rep., Oct. 2015.

[11] GSMA, "Operator collaboration brings major RCS coverage to the Japanese market," whitepaper, GSMA, Sept. 2018.

[12] L. Andrés, *The internet of things in health, social care, and wellbeing*. PhD thesis, KTH (Royal Institute of Technology), School of Information and Communication Technology, 2017. QC 20180828.

[13] A. Ghanbari, A. Laya, and J. Markendahl, "Value creation and coopetition in M2M ecosystem - The case of smart city," in *Proc. IEEE PIMRC*, Sept. 2016.

[14] J. Markendahl, *Mobile network operators and cooperation: A tele-economic study of infrastructure sharing and mobile payment services*. PhD thesis, KTH (Royal Institute of Technology), School of Information and Communication Technology, February 2011. QC 20110121.

[15] J. J. P.-A. Hsieh, A. Rai, and M. Keil, "Understanding digital inequality: Comparing continued use behavioral models of the socio-economically advantaged and disadvantaged," *MIS Quarterly*, vol. 32, pp. 97–126, 2006.

[16] 3GPP, "3rd generation partnership project; Technical specification group services and system aspects; Network sharing; Architecture and functional description," Technical Specification (TS), Tech. Rep. 23.251, July 2020.

[17] H. Ghazzai, E. Yaacoub, and M. Alouini, "Multi-operator collaboration for green cellular networks under roaming price consideration," in *Proc. VTC*, 2014.

[18] M. A. Safwat, "Framework for multi-operator collaboration for green communication," *IEEE Access*, vol. 6, pp. 850–865, 2018.

[19] S. Shah, S. Kittipiyakul, Y. Lim, and Y. Tan, "Multi-operator small cells collaboration using simultaneous bilateral negotiation," in *Proc. IEEE ECTI-CON*, 2018.

[20] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. IMLS ICML*, 2018.

[21] A. Papangelis, Y. Wang, P. Molino, and G. Tür, "Collaborative multi-agent dialogue model training via reinforcement learning," Arxiv. abs/1907.05507, 2019.

[22] D. Soldani and S. Illingworth, *5G AI-enabled automation*, pp. 1–38. Wiley Online Library, May 2020.

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

[24] 3GPP, "3rd generation partnership project;Technical specification group core network and terminals;numbering, addressing and identification;(Release 16)," Technical Specification (TS), Tech. Rep. 23.003, Sept. 2020.

[25] "German aerospace center (dlr): Sumo - simulation of urban mobility." https://sumo.dlr.de (Date last accessed 5-May-2020).

[26] "Openstreetmap.org." https://www.openstreetmap.org/ (Date last accessed 5-May-2020).

[27] "Unwired labs: Open cellid." https://opencellid.org (Date last accessed 5-May-2020).

[28] "Openai Gym." https://gym.openai.com (Date last accessed 5-May-2020).

[29] "Tensorflow." https://www.tensorflow.org (Date last accessed 5-May-2020).

[30] R. Inam *et al.*, "Feasibility assessment to realise vehicle teleoperation using cellular networks," in *Proc. IEEE ITSC*, 2016.

[31] A. Rasmussen, "The state of mvno in 2018 – more than 1,300 active mvnos in 79 countries." http://www.weconnectthailand.com/news/the-state-of-mvno-in-2018-more-than-1300-active-mvnos-in-79-countries/ (Date last accessed 5-May-2020).

Athanasios Karapantelakis is a Master Researcher in Artificial Intelligence at Ericsson and an Industrial Doctoral candidate at Royal Institute of Technology (KTH), Stockholm, Sweden. He received his B.Sc. degree in 2005 in Computer Science from University of Crete, Greece, and his M.Sc., Lic.Eng. degrees from KTH in Sweden. Athanasios has co-authored over 65 patent families and over 15 peer-reviewed scientific conference publications. At Ericsson, he held various engineering-related positions, from software design to systems architecture and research.

Elena Fersman is a Research Director in Artificial Intelligence at Ericsson. She is responsible of a team of researchers located in Sweden, USA, India, Hungary and Brazil. She is a docent and an adjunct professor in Cyber-Physical Systems specialized in Automation at the Royal Institute of Technology in Stockholm. She holds a Ph.D. in Computer Science from Uppsala University, a Master of Science in Economics and Management from St. Petersburg Polytechnic University and did a postdoc at the University Paris-Saclay. At Ericsson, she had various positions ranging from product management to research leadership. Elena is a member of the Board of Directors of RISE Research Institutes of Sweden. Elena has co-authored over 50 patent families.