Bakhshi, B., Mangues-Bafalluy, J. & Baranda, J. (7-11 Dec. 2021). *R-Learning-based admission control for service federation in multi-domain 5G networks* [proceedings]. 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain.

DOI: 10.1109/GLOBECOM46510.2021.9685936

R-Learning-Based Admission Control for Service Federation in Multi-domain 5G Networks

Bahador Bakhshi, Josep Mangues-Bafalluy, Jorge Baranda
Centre Tecnologic de Telecomunicacions de Catalunya (CTTC), Spain

***Abstract*—Network service federation in 5G/B5G networks enables service providers to extend service offering by collaborating with peering providers. Realizing this vision requires interoperability among providers towards end-to-end service orchestration across multiple administrative domains. Smart admission control is fundamental to make such extended offering profitable. Without prior knowledge of service requests, the admission controller (AC) either determines the domain to deploy the demand or rejects it to maximize the long-term average profit. In this paper, we first obtain the optimal AC policy by formulating the problem as a Markov decision process, which is solved through the policy iteration method. This provides the *theoretical* performance bound under the assumption of known arrival and departure rates of demands. Then, for practical solutions to be deployed in real systems, where the rates are not known, we apply the Q-Learning and R-Learning algorithms to approximate the optimal policy. The extensive simulation results show that learning approaches outperform the greedy policy and are capable of getting close to optimal performance. More specifically, R-learning always outperformed the rest of practical solutions and achieved an optimality gap of 3-5% independent of the system configuration, while Q-Learning showed lower performance and depended on discount factor tuning.**

***Index Terms*—Multi-domain 5G/B5G networks, Admission Control, Service Federation, MDP, Q-Learning, R-Learning**

I. Introduction

5G and beyond networks are expected to cost-effectively provide services with diverse quality requirements. To this end, enabler technologies including Network Function Virtualization (NFV) and Software Defined Networking (SDN) as well as architectural principles like network slicing, and multi-domain orchestration are incorporated in the architecture of the networks [1], [2]. The variety of services and stakeholders needs the definition of management and orchestration architectures for the cooperation of stakeholders that is required towards end-to-end multi-domain orchestration of such services. One clear example is service federation [3], [4].

Multi-domain orchestration enables the service provider, as an administrative domain, to collaborate with other domains for service provisioning in a federated environment [4]. The federation contract between the domains provides extra resources for the consumer domain at the cost of the federation. These resources are used to deploy (segments of) the requested network services, i.e., a network slice composed of a set of Virtual Network Functions (VNFs) and virtual interconnecting links, in other domains for different reasons such as the lack of sufficient local resources, load-balancing, and cost-efficiency.

The Admission Controller (AC) is the highest-level resource manager. For a given service demand, it either determines the domain to deploy the demand or rejects it. In this paper, we consider business profit as the objective of the admission control, and assume that there is only one provider domain with a limited reserved quota for service federation which is agreed in the federation contract. Under these assumptions, the AC should decide where to deploy the requested services to maximize the profit without knowing the future demands. This problem is referred to as the Admission Control for Service Federation (ACSF).

This problem is challenging, and trivial approaches like greedily accepting every demand do not provide the optimal solution, as shown in the simulation results. The AC should manage the local resources of the consumer domain and the federation quota of the provider domain to maximize the profit. While several AC algorithms in 5G networks have already been proposed [5], they consider single domain networks; and consequently are not directly applicable to the ACSF problem. This is the research gap that we aim to address.

Recently, AI/ML approaches have been extensively used in communication networks [6]. ACFS is an instance of the problem of sequential decision making under uncertainty, which can be efficiently approached by Reinforcement Learning (RL) solutions where an agent learns the policy via interaction with the environment [7]. The RL based admission control solutions have already been developed in other contexts rather than the multi-domain service federation problem [8]–[11]. In this paper, we utilize a special category of reinforcement learning, named *average reward* learning, to approximate the optimal policy that maximizes the average profit of the consumer domain. More specifically, we make the following contributions to the ACSF problem: i) The ACSF problem is formulated as a Markov Decision Process (MDP), which is solved by the Policy Iteration Dynamic Programming (DP) method to obtain the theoretical performance bound. ii) An average reward based learning algorithm is developed to approximate the optimal policy. iii) We show that the commonly used Q-Learning algorithm does not perform well in ACSF due to dependency on the discount factor.

The remainder of this paper is as follows. In Section II, we review the related works. The system models and the MDP formulation are presented in Section III. The model is solved by the Policy Iteration (PI) algorithm in Section IV. We present

This work has been partially funded by the MINECO grant TEC2017-88373-R (5G-REFINE), the EC H2020 5Growth Project (grant no. 856709), and Generalitat de Catalunya grant 2017 SGR 1195.

the learning approaches in Section V and evaluate them in Section VI. Section VII concludes this paper.

II. Related Work

Given the wide variety of stakeholders of future networks, service federation is expected to become a relevant component of 5G and beyond network architecture [1], since it allows deploying complex services in multiple administrative domains, whilst enabling end-to-end orchestration. At a theoretical level, the problem of service federation is formulated as an Integer Linear Programming model in [12], which is extended to consider energy efficiency [13]. From an architectural point of view, preliminary ideas of service federation were presented in [3] and [14]. The full-fledged architecture enabling service federation was developed in the 5G-Transformer [15] platform, which is capable of deploying composite NFV network services spanning across multiple domains and realizes the federation vision by designing the high-level concepts presented in ETSI specifications [4]. However, neither the theoretical nor the architectural research works address the admission control problem in service federation; i.e., they assume that the service has already been accepted and attempt to efficiently deploy it.

Using reinforcement learning for admission control has been the topic of several previous works. Adaptive call admission control (CAC) in multimedia networks using RL was studied in [8]. In the case of links with variable capacity, the CAC problem using RL was investigated in [16]. In [9], the authors utilized RL for CAC in CDMA networks. In [10], admission control in cellular networks, and in [17] network slice admission control were formulated as MDP. The network slice admission control in single domain networks was studied in [11], [18]. While these works approach the AC problem using RL, they cannot be applied directly to the ACSF problem since they consider single domain networks.

In [19], an initial attempt to apply Q-Learning to the service federation problem was presented. In this paper, we go beyond by formulating the problem as an MDP to obtain the optimal solution and we propose a new average-reward learning algorithm that outperforms Q-Learning under all evaluated scenarios and whose performance is not as sensitive to parameter tuning (e.g., discount factor) as Q-learning.

III. System Model and Problem Formulation

A. Assumptions and System Model

In this paper, service federation between a consumer domain and a provider domain is considered, where as mentioned, each requested service is a network slice composed of a set of VNFs and virtual interconnecting links. It is assumed that in the consumer domain, only one type of the resources (e.g., CPU) has a limited capacity LC. The domain offers a set of services, denoted by $\mathcal{I}$. Each service type $i \in \mathcal{I}$ is specified by (w_i, r_i) in the service catalog, where w_i is the total amount of required resources (e.g., the total number of CPU cores) by all the VNFs of the service type, and r_i is the revenue that the provider gains if it accepts a demand of this type.

For service federation, the provider domain reserved a resource quota with capacity PC units, which is agreed in the federation contract. The contract also specifies the federation cost ϕ_i, which is the cost of deploying an instance of service type i in the provider domain that is paid by the consumer domain. Demands beyond the capacity PC will be rejected.

Over time, customers request instances of the services. The load of service type i corresponds to a traffic class $(\lambda_i, \mu_i, w_i, r_i, \phi_i)$ where λ_i and μ_i are, respectively, the arrival and departure rates of the demands of the type, which are assumed to be a Poisson process. Accordingly, each demand δ_i of class i is determined by $(\tau^s_\delta, \tau^e_\delta, w_i, r_i, \phi_i)$ where the τ^s_δ and τ^e_δ are the start and end time of the demand, which are respectively specified by λ_i and μ_i.

Upon arrival of δ_i, the AC should determine in which domain to deploy the demand or to reject it. To accept the demand in the consumer domain, w_i units of the *local* resources are allocated to the demand and r_i units of money is earned. In the case of the federation, the demand is deployed in the provider domain; so, no resource is consumed in the local domain and the profit is $r_i - \phi_i$. If the demand is rejected, no resource is used and no revenue is earned.

In ACSF, we assume there is a set $\mathcal{D}$ of demands, which arrive one-by-one. At the arrival time of a demand, AC is not aware of the future demands and also does not know the expected life-time of the given demand. The objective is to find an AC policy that maximizes the average profit, which is

$$\frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{I}} \Big(\sum_{\delta_i \in \mathcal{L}} r_i + \sum_{\delta_i \in \mathcal{F}} (r_i - \phi_i) \Big), \tag{1}$$

where $\mathcal{L}$ and $\mathcal{F}$ are, respectively, the set of the demands that are deployed in the local consumer domain and in the federated provider domain.

B. The MDP Model of ACSF

In this section, under the assumption of knowing λ_i and μ_i $\forall i \in \mathcal{I}$, the ACSF problem is formulated as a Markov decision process. An MDP is a 4-tuple $\big(\mathcal{S},\ \mathcal{A},\ P_a(s,s'),\ R_a(s,s')\big)$ where $\mathcal{S}$ is the set of the states, $\mathcal{A}$ is the set of actions, $P_a(s,s')$ is the transition probability from state s to state s' if the agent takes action a in the state s, and $R_a(s,s')$ is the reward of the action a in state s that leads to transition to state s'.

In ACSF, the state of the environment is defined as $s = (\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{d})$, where $\boldsymbol{l}$ and $\boldsymbol{f}$ are vectors wherein the i-th element is the number of active (alive) demands of the class i in the local consumer domain and in the provider domain, respectively. Similar to [10], $\boldsymbol{d}$ is a vector whose i-th element is $+1$ if a demand of class i arrives, it is -1 if a demand of class i departs the network, and is 0 otherwise. Let $\mathcal{S}^+ = \{s \,|\, \exists i \text{ s.t. } \boldsymbol{d}[i] = +1\}$ and similarly $\mathcal{S}^- = \{s \,|\, \exists i \text{ s.t. } \boldsymbol{d}[i] = -1\}$; we have these constrains on the states to make the problem tractable: *i)* $\mathcal{S} = \mathcal{S}^+ \cup \mathcal{S}^-$ and $\mathcal{S}^+ \cap \mathcal{S}^- = \{\}$. *ii)* State $(\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{0})$ is an invalid state as there is not any arrival or departure events.

TABLE I: The Transition Probabilities and Rewards of the MDP

Current state s	Action in state s	Next event after this action	Next state s'	Transition probability $P_a(s, s')$	Reward $R_a(s, s')$
$(\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{e}_i)$	reject	Arrival of a demand of class j	$(\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{e}_j)$	$\frac{\lambda_j}{\Lambda(s')+M(s')}$	0
$(\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{e}_i)$	reject	Departure of a demand of class j	$(\boldsymbol{l}, \boldsymbol{f}, -\boldsymbol{e}_j)$	$\frac{(\boldsymbol{l}'[j]+\boldsymbol{f}'[j])\mu_j}{\Lambda(s')+M(s')}$	0
$(\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{e}_i)$	accept	Arrival of a demand of class j	$(\boldsymbol{l} + \boldsymbol{e}_i, \boldsymbol{f}, \boldsymbol{e}_j)$	$\frac{\lambda_j}{\Lambda(s')+M(s')}$	r_i
$(\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{e}_i)$	accept	Departure of a demand of class j	$(\boldsymbol{l} + \boldsymbol{e}_i, \boldsymbol{f}, -\boldsymbol{e}_j)$	$\frac{(\boldsymbol{l}'[j]+\boldsymbol{f}'[j])\mu_j}{\Lambda(s')+M(s')}$	r_i
$(\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{e}_i)$	federate	Arrival of a demand of class j	$(\boldsymbol{l}, \boldsymbol{f} + \boldsymbol{e}_i, \boldsymbol{e}_j)$	$\frac{\lambda_j}{\Lambda(s')+M(s')}$	$r_i - \phi_i$
$(\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{e}_i)$	federate	Departure of a demand of class j	$(\boldsymbol{l}, \boldsymbol{f} + \boldsymbol{e}_i, -\boldsymbol{e}_j)$	$\frac{(\boldsymbol{l}'[j]+\boldsymbol{f}'[j])\mu_j}{\Lambda(s')+M(s')}$	$r_i - \phi_i$
$(\boldsymbol{l}, \boldsymbol{f}, -\boldsymbol{e}_i)$	no_action	Arrival of a demand of class j	$(\boldsymbol{l} - \boldsymbol{e}_i, \boldsymbol{f}, \boldsymbol{e}_j)$	$\frac{\boldsymbol{l}[i]}{\boldsymbol{l}[i]+\boldsymbol{f}[i]} \times \frac{\lambda_j}{\Lambda(s')+M(s')}$	0
$(\boldsymbol{l}, \boldsymbol{f}, -\boldsymbol{e}_i)$	no_action	Arrival of a demand of class j	$(\boldsymbol{l}, \boldsymbol{f} - \boldsymbol{e}_i, \boldsymbol{e}_j)$	$\frac{\boldsymbol{f}[i]}{\boldsymbol{l}[i]+\boldsymbol{f}[i]} \times \frac{\lambda_j}{\Lambda(s')+M(s')}$	0
$(\boldsymbol{l}, \boldsymbol{f}, -\boldsymbol{e}_i)$	no_action	Departure of a demand of class j	$(\boldsymbol{l} - \boldsymbol{e}_i, \boldsymbol{f}, -\boldsymbol{e}_j)$	$\frac{\boldsymbol{l}[i]}{\boldsymbol{l}[i]+\boldsymbol{f}[i]} \times \frac{(\boldsymbol{l}'[j]+\boldsymbol{f}'[j])\mu_j}{\Lambda(s')+M(s')}$	0
$(\boldsymbol{l}, \boldsymbol{f}, -\boldsymbol{e}_i)$	no_action	Departure of a demand of class j	$(\boldsymbol{l}, \boldsymbol{f} - \boldsymbol{e}_i, -\boldsymbol{e}_j)$	$\frac{\boldsymbol{f}[i]}{\boldsymbol{l}[i]+\boldsymbol{f}[i]} \times \frac{(\boldsymbol{l}'[j]+\boldsymbol{f}'[j])\mu_j}{\Lambda(s')+M(s')}$	0

iii) State $(\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{d})$ where $\exists i, j$ s.t. $\boldsymbol{d}[i] \neq 0$ and $\boldsymbol{d}[j] \neq 0$ is an invalid state as two events cannot occur at the same time[1].

$\mathcal{A}(s)$ is the set of *valid* actions in state s. Define $\widetilde{LC}$ and $\widetilde{PC}$ as the current *available* capacity of resources in the consumer and provider domains, respectively. For $s \in \mathcal{S}^+$ where $\exists i$ s.t. $\boldsymbol{d}[i] = +1$, $\mathcal{A}(s)$ includes *i)* reject, *ii)* accept only if $\widetilde{LC} \geq w_i$, and *iii)* federate only if $\widetilde{PC} \geq w_i$. However, if $s \in \mathcal{S}^-$ then the only valid action is an artificial action, named no_action, since in this case, the agent does not do any action and a demand departs the system.

As mentioned above, it is assumed that parameters λ_i and μ_i are known and determine the transition *rates*. In MDP, the transition *probabilities* should be derived. According to the theory of competing exponentials, the probability of a transition in a state equals to the rate of the transition divided by the total transition rates in the state. The total transitions rates of state $s = (\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{d})$ is $\Lambda(s) + M(s)$ where $\Lambda(s) = \sum_{i \in \mathcal{I}} \lambda_i$ is the total arrival rate of demands and $M(s) = \sum_{i \in \mathcal{I}} (\boldsymbol{l}[i] + \boldsymbol{f}[i])\mu_i$ is the total departure rate of the demands in the state.

We assume the transition from s to s' occurs in two steps. At first, the agent makes a decision that takes place immediately in the system, i.e., $\boldsymbol{l}$ or $\boldsymbol{f}$ changes to $\boldsymbol{l}'$ or $\boldsymbol{f}'$ *before* occurring the next arrival/departure event. Then, in the second step, the environment brings up a new event, i.e., it changes $\boldsymbol{d}$ to $\boldsymbol{d}'$, and consequently the system goes from $s = (\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{d})$ to $s' = (\boldsymbol{l}', \boldsymbol{f}', \boldsymbol{d}')$. Therefore, to compute the transition probability from s to s', the total transition rates of state s' should be used as the rate of the next event depends on $\boldsymbol{l}'$ and $\boldsymbol{f}'$.

Let $\boldsymbol{e}_i$ be a vector with a 1 in the i-th element and 0 in the others. The transition probabilities and corresponding rewards are shown in Table I. They are computed as follows. First, we apply the action in state s that determines $\boldsymbol{l}'$ and $\boldsymbol{f}'$, then we utilize the competing exponentials theorem for the given next event. For example, the first row in the table is the case that a new demand of class i arrives, so the state is $s = (\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{e}_i)$. The agent decides to reject it; hence, the reward is 0, $\boldsymbol{l}' = \boldsymbol{l}$, and $\boldsymbol{f}' = \boldsymbol{f}$. The next event after the action is the arrival of a demand from class j, so the next state s' is $(\boldsymbol{l}, \boldsymbol{f}, \boldsymbol{e}_j)$ with probability of $\frac{\lambda_j}{\Lambda(s')+M(s')}$. If the next event is the departure of a demand of class j, the next state will be $s' = (\boldsymbol{l}, \boldsymbol{f}, -\boldsymbol{e}_j)$; the rate of this event is $(\boldsymbol{l}'[j] + \boldsymbol{f}'[j])\mu_j$, which is shown in the second row of the table.

IV. Optimal Policy by Dynamic Programming

Given the MDP model, Dynamic Programming (DP) algorithms, e.g., the Policy Iteration (PI) method, can solve it and find the optimal policy [7]. The method is depicted in Algorithm 1, where the parameter θ determines the precision and γ is the reward discount factor. In the policy iteration loop, the given policy π is evaluated by updating the state values $V(s)$ using the transition probabilities and rewards until the desired precision is achieved; then in the policy improvement loop, for each state, the old action $\bar{a}$ is compared against the new action obtained from the updated $V(s)$; if they are not the same, then these two loops are iterated.

To solve the ACSF problem using the PI algorithm, an important issue needs to be addressed properly. The optimal policy found by this algorithm in fact optimizes $V(s)$, which is the *discounted* state values. Therefore, the policy is optimal with respect to the discount factor γ. Different values of γ can/may lead to different policies; e.g., $\gamma = 0$ implies that the policy only aims to maximize the one-step/immediate reward $R_a(s, s')$, which is exactly the greedy policy does.

In ACSF, the objective is to maximize the average profit defined by (1), which is, indeed, the *average reward* rather than the discounted reward optimized by the PI algorithm. While some DP methods have been proposed in the literature to find the optimal average reward policy, it is also known that by $\gamma \rightarrow 1$, maximizing the discounted reward approximates the average reward [20]. So, in the ACSF problem, we set $\gamma \approx 1$ to approximate the optimal policy.

While the PI algorithm can find the optimal policy, in most practical circumstances, the arrival and departure rates of the demands are not known; moreover, the number of states exponentially grows in terms of $|\mathcal{I}|$, LC, and PC. Hence, the DP methods are not practical solutions. We will use them to obtain the theoretical performance bound for evaluating the practical solutions presented in the following sections.

[1] Please note that the AC only takes an action in states $s \in \mathcal{S}^+$; however, we consider states $s \in \mathcal{S}^-$ in the MDP as these states correspond to the departure of demands wherein the capacity of the resources changes. These states are used to facilitate deriving the transition probabilities.

Algorithm 1: PI($\mathcal{S}$, $\mathcal{A}$, P_a, R_a, θ, γ)

```
Arbitrarily initialize $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)\ \forall s \in \mathcal{S}$
while $\pi$ is not stable do
    while $\Delta > \theta$ do                ▷ The policy evaluation loop
        $\Delta \leftarrow 0$
        for $\forall s \in \mathcal{S}$ do
            $v \quad V(s)$
            $a \leftarrow \pi(s)$
            $V(s) \quad \sum_{s'} P_a(s,s')\big(R_a(s,s') + \gamma V(s')\big)$
            $\Delta \leftarrow \max(\Delta, |V(s) - v|)$
    for $\forall s \in \mathcal{S}$ do                ▷ The policy improvement loop
        $\bar{a} \quad \pi(s)$
        $\pi(s) \quad \text{argmax}_a \sum_{s'} P_a(s,s')\big(R_a(s,s') + \gamma V(s')\big)$
        if $\bar{a} \neq \pi(s)$ then
            $stable \quad false$
return $\pi$
```

Algorithm 2: Q-Learning(n, m, α, γ, ϵ)

```
Arbitrarily initialize $Q[s,a] \in \mathbb{R}\ \forall s \in \mathcal{S}$, $\forall a \in \mathcal{A}(s)$
for $n$ times do
    $\alpha \leftarrow 0.99\alpha$, $\epsilon \leftarrow 0.99\epsilon$
    $s \leftarrow$ environment state $(\mathbf{0}, \mathbf{0}, \boldsymbol{d})$
    for $m$ times do
        $a \quad$ action from $\mathcal{A}(s)$ by $\epsilon$-greedy strategy
        Action $a$ is performed by the environment
        $s', R_a(s,s') \quad$ next state and reward from the environment
        $Q[s,a] \quad (1-\alpha)Q[s,a] + \alpha\Big(R_a(s,s') + \gamma \max_{a'} Q[s',a']\Big)$
        $s \leftarrow s'$
$\pi(s) \quad \text{argmax}_a Q[s,a]\ \forall s \in \mathcal{S}$
return $\pi$
```

V. Practical Solution via Learning

In this section, RL approaches are applied to the ACSF problem to deal with the issues of the DP solutions. These approaches instead of finding the optimal policy by exploiting the transition probabilities, *learn* the policy via interaction with the environment over time; therefore, they need neither the transition probabilities nor enumerating all the possible states.

A. Q-Learning

Q-Learning is one of the well-known RL algorithms that use the concept of *temporal difference* to iteratively solve the Bellman optimality equations. Details of the algorithm are depicted in Algorithm 2 [7]. It maintains a table Q of values of each action in each state, denoted by $Q[s,a]$. At state s, the agent selects an action which is determined by the values $Q[s,.]$ and the exploration strategy. Then, it observes the next state s', gets reward $R_a(s,s')$ from the environment, and consequently updates the $Q[s,a]$ as follows

$$Q[s,a] = (1-\alpha)Q[s,a] + \alpha\Big(R_a(s,s') + \gamma \max_{a'} Q[s',a']\Big),$$

where α is the learning rate. These interactions take place in n learning episodes with m number of demands per episode.

The general Q-Learning algorithm is customized for the ACSF problem as follows. Action in each state is chosen by the ϵ-greedy strategy [7]. At the beginning of each episode, the values of parameters α and ϵ are decayed to rely more on the learned Q values over time. In the beginning, the large value of α causes the agent to learn faster and the large value of ϵ allows it to explore more. But later, decreasing these parameters in subsequent episodes forces the agent to pay more attention to the Q values that it has learned.

Unlike the parameters α and ϵ, the proper setting of γ is not straightforward. Similar to PI, $\gamma = 0$ turns Q-Learning to the greedy policy; however, $\gamma \approx 1$ doesn't work well because of *bootstrapping* where the value of the next state is overestimated as $\max_{a'} Q[s',a']$. While $\gamma > 0$ allows the agent to consider expected future rewards in decision making, $\gamma \to 1$ in combination with bootstrapping can cause the value of $\gamma \max_{a'} Q[s',a']$ surpasses $R_a(s,s')$ in the value update equation; and consequently, the agent underestimates the importance of the immediate rewards. In ACSF, it implies that while $R_{\textsf{accept}}(s,s') > R_{\textsf{federate}}(s,s')$, the agent may prefer federate instead of accept. This is problematic in the case of $LC \gg \sum_{i \in I}(\lambda_i/\mu_i)w_i$ where the optimal action is accept $\forall s \in \mathcal{S}^+$; but the Q-Learning policy incorrectly selects federate for some states that leads to a sub-optimal policy. The effect of γ is evaluated in Section VI.

B. R-Learning

In this section, to resolve the issue of the discount factor, which is needed in Q-Learning, we apply *average reward* RL to the ACSF problem, where the agent directly maximizes the average reward instead of the discounted reward [21].

The R-Learning algorithm is one of the average reward RL solutions [20]. The details of the algorithm are shown in Algorithm 3. Due to the similarities between the Q-Learning and R-Learning algorithms, we omit the explanation of the common steps and only emphasize the differences. Contrary to the Q-Learning, the state-action values, $Q[s,a]$, are not the expected discounted reward. The key idea of the R-Learning algorithm is that in the infinite horizon and ergodic MDPs, the average reward, which is denoted by ρ in the algorithm, is independent of the state. Therefore, the algorithm by $Q[s,a]$ keeps tracking the difference between ρ and the expected average reward of action a in state s. The $Q[s,a]$ value is updated according to the difference between the expected average reward ρ and the immediate reward $R_a(s,s')$ and also the value of the next state as follows:

$$Q[s,a] \quad (1-\alpha)Q[s,a] + \alpha\Big(\big(R_a(s,s') - \rho\big) + \max_{a'} Q[s',a']\Big)$$

Since the average reward is not known at the beginning, the algorithm also learns it. As seen in line 12, ρ is updated by the learning rate β. The conditional update is to avoid the skews made due to randomness of the exploration strategy [20].

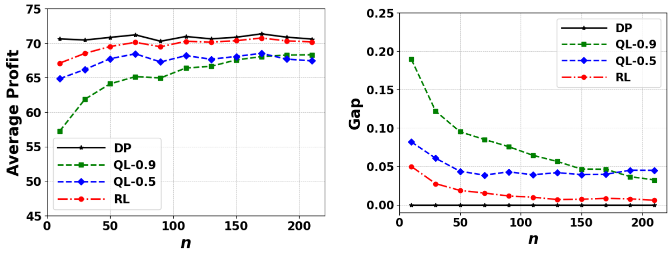
VI. Simulation Results

In this section, we evaluate the performance of the RL approaches in comparison to the greedy policy, where the default action is accept; and federate is taken only if the consumer domain has not sufficient resources. The metrics are the average profit (1), and the optimality gap $(AP_{DP} - AP_{Alg})/AP_{DP}$, Algorithm 3: R-Learning(n, m, α, β, ϵ)

```
Arbitrarily initialize $Q[s,a] \in \mathbb{R}\ \forall s \in \mathcal{S},\ \forall a \in \mathcal{A}(s)$, $\rho \quad 0$
for $n$ times do
    $\alpha \leftarrow 0.99\alpha$, $\epsilon \leftarrow 0.99\epsilon$, $\beta \leftarrow 0.99\beta$
    $s \quad$ environment state $(\mathbf{0}, \mathbf{0}, \boldsymbol{d})$
    for $m$ times do
        $a \quad$ action from $\mathcal{A}(s)$ by $\epsilon$-greedy strategy
        Action $a$ is performed by the environment
        $s', R_a(s,s') \quad$ next state and reward from the environment
        $Q[s,a] \quad (1-\alpha)Q[s,a] + \alpha\Big(\big(R_a(s,s') - \rho\big) + \max_{a'} Q[s',a']\Big)$
        if $Q[s,a] = \max_a Q(s,a)$ then
            $\rho \quad (1-\beta)\rho + \beta\Big(R_a(s,s') - \max_a Q[s,a] + \max_{a'} Q[s',a']\Big)$
        $s \leftarrow s'$
$\pi(s) \quad \text{argmax}_a Q[s,a]\ \forall s \in \mathcal{S}$
return $\pi$
```

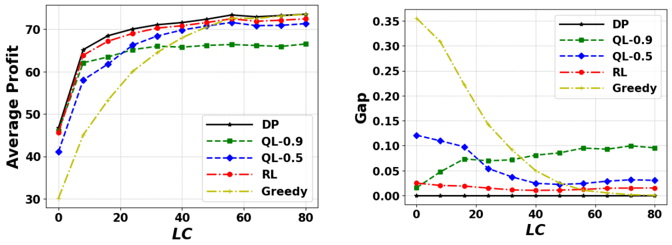
TABLE II: The Default Simulation Settings

LC	PC	λ_1	μ_1	w_1	r_1	ϕ_1	λ_2	μ_2	w_2	r_2	ϕ_2
30	20	10	4	2	100	30	5	0.5	4	20	5



(a) Average Profit (b) Optimality Gap

Fig. 1: The effect of the number of the episodes



(a) Average Profit (b) Optimality Gap

Fig. 2: The effect of the capacity of the consumer domain

where AP_{Alg} is the average profit of the given algorithm. In the following, the effect of the capacity of the local and the provider domains and also the impact of the federation cost are investigated. In these simulations, $\gamma = 0.99$ in PI, and $n = 200$, $m = 4000$, $\epsilon = 0.9$, and $\alpha = \beta = 0.9$ in the learning algorithms. The *default* configuration (if not stated otherwise) of the domains and traffic classes are shown in Table II. The following results are the average of 10 experiments.

A. The Effect of the Number of Episodes

The Q-Learning and R-Learning algorithms learn the policy over time via interaction with the environment. The number of the episodes determines the length of the learning period. The performance of the algorithms with respect to the number of the episodes n is shown in Figures 1a and 1b.

The results show that after a number of episodes, both Q-Learning and R-Learning converge. The RL algorithm outperforms QL not only in terms of optimality gap, which is less than 2%, but also in terms of the number of episodes needed to converge. These results also show the dependency of the performance of Q-Learning on the discount factor γ. In this setting, while at the beginning, QL with $\gamma = 0.5$ performs better than $\gamma = 0.9$, finally QL-0.9 converges to a better result.

B. The Effect of the Consumer Domain Capacity

Efficient management of the local resources of the consumer domain by the AC greatly influences the achievable profit. Figures 2a and 2b show the performance of the different policies with respect to the local domain capacity LC. Our simulations show the performances of the algorithms with respect to PC is similar (omitted due to space limitations).

The performance of the R-Learning algorithm is excellent independent of LC. When the local domain has not sufficient resource to accept all the offered load, i.e., $LC < \sum(w_i\lambda_i/\mu_i)$, the intelligent decisions by the RL algorithms greatly improve the performance in comparison to the greedy policy. In the case of very large capacities, all demands can be accepted in the consumer domain, so the greedy algorithm is also the optimal policy. Again, QL performance is affected by γ and a single γ is not the best choice for all cases. In cases of small capacities, large γ is better, but in the large capacities, it should be small.

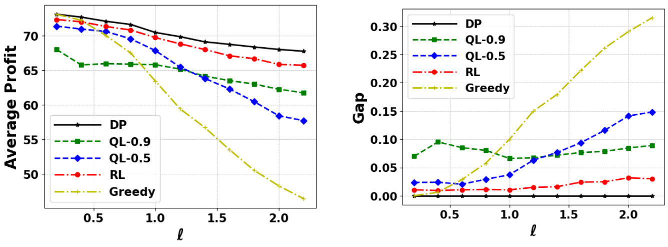
C. The Effect of the Offered Load

The AC, as the resource manager, should efficiently handle the offered load. In this section, by scaling the arrival rates of demands as $\ell\lambda_i$, the performance of the algorithms is investigated. The results are shown in Figures 3a and 3b.

In these results, it is seen that in the case of very small ℓ, where all demands can be accepted in the consumer domain as $LC \gg \ell\sum(w_i\lambda_i/\mu_i)$, the greedy policy performs well. However, by increasing the offered load, some demands should be sent to the provider domain or rejected. In these cases, the learning algorithms outperform the greedy policy by making smart decisions. Similar to the other results, the RL algorithm has a near-optimal performance. These results show that the appropriate value of the discount factor γ in Q-Learning also depends on the offered load.

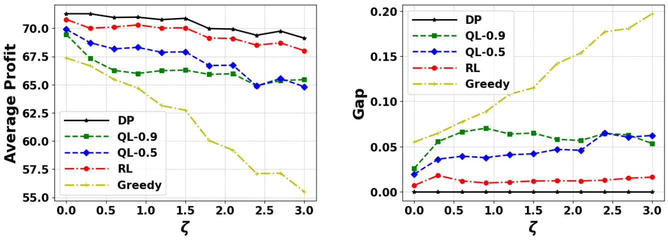
D. The Effect of the Federation Cost

As mentioned, for each service type i that is agreed in the federation contract, the provider domain charges the consumer domain an amount ϕ_i per demand that is deployed in the provider domain. The AC should take this cost into account. For example, if the federation cost is very high, the optimal



(a) Average Profit (b) Optimality Gap

Fig. 3: The effect of the offered load



(a) Average Profit (b) Optimality Gap

Fig. 4: The effect of the federation cost

decision would be to reject the demand. In this section, the admission control policies are evaluated with respect to the cost that is scaled $\zeta\phi_i$. Figures 4a and 4b show the average profit and the gap of the policies with respect to ζ.

These results show that by increasing the federation cost, as expected, the profits of all policies decrease. The optimality gap of R-Learning is independent of the scale ζ that implies it considers the federation cost properly in the admission control process. The Q-Learning algorithm attempts to maintain the gap; however, it does not perform as good as R-Learning. The optimality gap of the greedy policy increases by enlarging ζ since the non-optimal decisions by the greedy policy incur more cost in the case of larger ζ.

VII. Conclusion and Future Work

In this paper, we investigated the ACSF problem, where the admission controller determines the domain to deploy the service or rejects it in order to maximize the profit considering the federation cost. The optimal policy under the assumption of knowing the arrival and departure rates of the demands was obtained by solving the MDP model of the problem through the policy iteration algorithm. As practical solutions, we applied the Q-Learning and R-Learning algorithms to the problem where the former maximizes the discounted rewards while the latter attempts to maximize the average reward. The extensive simulations show the excellent performance of the R-learning independent of system configuration.

For future work, the next step would be taking into account the capacity of the intra-domain links in multiple provider domains context. These extensions will cause the exponential growth of the state space that needs to be handled by Deep RL solutions.

Acknowledgment

We thank Andres Garcia-Saavedra for reviewing and providing feedback on the draft version of the manuscript.

References

[1] N. Alliance, "5g end-to-end architecture framework, v3.0.8," *Tech. Rep.*, 2019.

[2] E. N. ISG, "Network function virtualisation (nfv): Management and orchestration: Report on architecture options to support multiple administrative domains," *ETSI GR NFV-IFA*, vol. 028, 2018.

[3] L. Valcarenghi, B. Martini, K. Antevski, C. Bernardos, G. Landi, M. Capitani, J. Mangues-Bafalluy, R. Martínez, J. Baranda, I. Pascual, *et al.*, "A framework for orchestration and federation of 5g services in a multi-domain scenario," in *Workshop on Experimentation and Measurements in 5G*, pp. 19–24, 2018.

[4] J. Baranda, J. Mangues-Bafalluy, R. Martinez, L. Vettori, K. Antevski, C. J. Bernardos, and X. Li, "Realizing the network service federation vision: Enabling automated multidomain orchestration of network services," *IEEE Vehicular Technology Magazine*, vol. 15, no. 2, pp. 48–57, 2020.

[5] M. O. Ojijo and O. E. Falowo, "A survey on slice admission control strategies and optimization schemes in 5g network," *IEEE Access*, vol. 8, pp. 14977–14990, 2020.

[6] M. E. Morocho-Cayamcela, H. Lee, and W. Lim, "Machine learning for 5g/b5g mobile and wireless communications: Potential, limitations, and future directions," *IEEE Access*, vol. 7, pp. 137184–137206, 2019.

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

[8] H. Tong and T. X. Brown, "Adaptive call admission control under quality of service constraints: a reinforcement learning solution," *IEEE Journal on selected Areas in Communications*, vol. 18, no. 2, pp. 209–221, 2000.

[9] D. Liu, Y. Zhang, and H. Zhang, "A self-learning call admission control scheme for cdma cellular networks," *IEEE transactions on neural networks*, vol. 16, no. 5, pp. 1219–1228, 2005.

[10] C. C. Wu and D. P. Bertsekas, "Admission control for wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 50, pp. 504–514, 2001.

[11] P. Caballero, A. Banchs, G. De Veciana, X. Costa-Pérez, and A. Azcorra, "Network slicing for guaranteed rate services: Admission control and resource allocation games," *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, pp. 6419–6432, 2018.

[12] G. Sun, Y. Li, D. Liao, and V. Chang, "Service function chain orchestration across multiple domains: A full mesh aggregation approach," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 1175–1191, 2018.

[13] G. Sun, Y. Li, H. Yu, A. V. Vasilakos, X. Du, and M. Guizani, "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks," *Future Generation Computer Systems*, vol. 91, pp. 347–360, 2019.

[14] X. Li, J. Mangues-Bafalluy, I. Pascual, G. Landi, F. Moscatelli, K. Antevski, C. J. Bernardos, L. Valcarenghi, B. Martini, C. F. Chiasserini, *et al.*, "Service orchestration and federation for verticals," in *IEEE WCNC Workshops*, pp. 260–265, 2018.

[15] H2020 5G-TRANSFORMER, "5g mobile transport platform for verticals." http://5g-transformer.eu/. Accessed: 2020-12-06.

[16] A. Pietrabissa, "A reinforcement learning approach to call admission and call dropping control in links with variable capacity," *European Journal of Control*, vol. 17, no. 1, pp. 89–101, 2011.

[17] B. Han, D. Feng, and H. D. Schotten, "A markov model of slice admission control," *IEEE Networking Letters*, vol. 1, no. 1, 2018.

[18] M. R. Raza, C. Natalino, P. Öhlen, L. Wosinska, and P. Monti, "A slice admission policy based on reinforcement learning for a 5g flexible ran," in *European Conference on Optical Communication*, pp. 1–3, 2018.

[19] K. Antevski, J. Martín-Pérez, A. Garcia-Saavedra, C. J. Bernardos, X. Li, J. Baranda, J. Mangues-Bafalluy, R. Martnez, and L. Vettori, "A q-learning strategy for federation of 5g services," in *IEEE ICC*, 2020.

[20] A. Schwartz, "A reinforcement learning method for maximizing undiscounted rewards," in *International conference on machine learning*, pp. 298–305, 1993.

[21] V. Dewanto, G. Dunn, A. Eshragh, M. Gallagher, and F. Roosta, "Average-reward model-free reinforcement learning: a systematic review and literature mapping," *arXiv preprint arXiv:2010.08920*, 2020.