Optimising 5G infrastructure markets: The Business of Network Slicing

Dario Bega*, Marco Gramaglia*, Albert Banchs*, Vincenzo Sciancalepore†, Konstantinos Samdanis‡ and Xavier Costa-Perez†

*Institute IMDEA Networks and University Carlos III of Madrid, Madrid, Spain
Email: {dario.bega, marco.gramaglia, albert.banchs}@imdea.org
†NEC Laboratories Europe, Heidelberg, Germany, Email: {vincenzo.sciancalepore, xavier.costa}@neclab.eu
‡Huawei Technologies, Munich, Germany, Email: konstantinos.samdanis@huawei.com

***Abstract*—In addition to providing substantial performance enhancements, future 5G networks will also change the mobile network ecosystem. Building on the network slicing concept, 5G allows to "slice" the network infrastructure into separate logical networks that may be operated independently and targeted at specific services. This opens the market to new players: the *infrastructure provider*, which is the owner of the infrastructure, and the *tenants*, which may acquire a network slice from the infrastructure provider to deliver a specific service to their customers. In this new context, we need new algorithms for the allocation of network resources that consider these new players. In this paper, we address this issue by designing an algorithm for the admission and allocation of network slices requests that (i) maximises the infrastructure provider's revenue and (ii) ensures that the service guarantees provided to tenants are satisfied. Our key contributions include: (i) an analytical model for the admissibility region of a network slicing-capable 5G Network, (ii) the analysis of the system (modelled as a Semi-Markov Decision Process) and the optimisation of the infrastructure provider's revenue, and (iii) the design of an adaptive algorithm (based on Q-learning) that achieves close to optimal performance.**

I. Introduction

By leveraging on novel concepts of virtualization and programmability, future 5G Networks [1] are expected to be reliable, high-performing and cost-efficient. However, the scope of 5G goes beyond just pure performance metrics, and incorporates profound changes in its architecture design which will change the mobile network ecosystem. One of the key novel concepts of the 5G architecture is that of *Network Slicing* [2], driven by very diverse requirements demanded by 5G. Indeed, there is a consensus in that accommodating such heterogeneous services using the same infrastructure will not be possible with the current, relatively monolithic architecture in a cost efficient way. In contrast, with network slicing the infrastructure can be divided in different *slices*, each of which can be tailored to meet specific service requirements.

With network slicing, different services (such as, e.g., automotive, mobile broadband or haptic Internet) can be provided by different network slice instances. Each of these instances consists of a set of virtual network functions that run on the same infrastructure with a especially tailored orchestration. In this way, very heterogeneous requirements can be provided by the same infrastructure, as different network slice instances can be orchestrated and configured according to their specific requirements. Additionally, this can be performed in a cost-efficient manner, as the different network slice tenants dynamically share the same infrastructure.

This novel approach does not just provide better performing and more efficient networks, but it also makes room for new players in the mobile network ecosystem. Network slices allow for a role separation between *infrastructure providers* (the ones who provide computational and network resources used by different network slices) and *network slice tenants* (the ones acquiring a slice to orchestrate and run network functions within that slice to provide a certain service to their customers).

The above model is currently being successfully applied by Infrastructure as a Service (IaaS) providers such as Amazon Web Services or Microsoft Azure, which sell their computational resources such as CPU, disk or memory for Virtual Network Function (VNF) purposes. While such an IaaS approach follows a very similar business model to ours, providing network resources is an intrinsically different problem, since (i) spectrum is a scarce resource for which over-provisioning is not possible, (ii) the actual capacity of the systems (i.e., the resources that can actually be sold) heavily depends on the mobility patterns of the users, and (iii) the Service Level Agreements (SLAs) with network slices tenants usually impose stringent requirements on the Quality of Experience (QoE) perceived by their users. Therefore, in contrast to IaaS, in our case applying a strategy where all the requests coming to the infrastructure provider are admitted is simply not possible.

In the above context, the new 5G ecosystem calls for novel algorithms and solutions for the allocation of the (scarce) network resources among tenants; this is the so-called spectrum market. In this paper we address this problem by designing a network capacity brokering solution, implemented by an infrastructure provider, that assigns resources to tenants and their respective network slices. When taking the decision of whether to admit or reject new network slices requests (and the associated resources), our solution aims at (i) maximising the revenue of a network infrastructure provider, and (ii) satisfying the service guarantees provided to the network slices.

While the network slicing concept has only been proposed recently [2], it has already attracted substantial attention. 3GPP has started working on the definition of requirements

for network slicing [3], whereas NGMN identified network sharing among slices (which is the focus of this paper) as one of the key issues to be addressed [4]. Despite these efforts, most of the work has focused on architectural aspects [5], [6] with only a limited focus on resource allocation algorithms. While there is a body of work on the literature on spectrum sharing (see e.g. [7]–[10]), these proposal are not tailored to the specific requirements of the 5G ecosystem.

The rest of the paper is structured as follows. We first introduce our system model in Section II. In Section III we conduct a performance analysis of our system in order to determine the amount network slice requests that can be sold by the infrastructure provider while meeting the service guarantees of the corresponding traffic types. Building on this model, in Section IV we address the problem of designing an admission control algorithm that maximises the infrastructure provider revenue while satisfying the desired service guarantees; to this end, we first analyse the revenue resulting from a given admission control policy and then obtain the optimal policy that maximises the resulting revenue. Building on this analysis, in Section V we design a practical adaptive algorithm that provides a performance close to that of the optimal admission policy. The performance of the adaptive and optimal algorithms are thoroughly evaluated via simulation in Section VI, and the paper concludes with some final remarks in Section VII.

II. System Model

In the following, we describe the various aspects related to our system model.

Players. In our system model, there are the following players: (i) the *infrastructure provider*, who is the owner of the network and provides *network slices* corresponding to a certain fraction of network resources to the tenants, (ii) the *tenants*, which issue requests to the infrastructure provider to acquire network resources, and use these resources to serve their users, and finally (iii) the *end-users*, which are served by their tenant or operator and run their applications in the tenant's slice.

Network model. Our network is composed of a set of base stations $\mathcal{B}$. For each base station $b \in \mathcal{B}$, we let C_b denote the base station capacity. We further refer to the system capacity as the sum of the capacity of all base stations, $C = \sum_{\mathcal{B}} C_b$. We let $\mathcal{U}$ denote the set of users in the network.[1] We consider that each user $u \in \mathcal{U}$ in the system is associated to one base station $b \in \mathcal{B}$, each of them with a nominal transmission rate C_b. We denote by f_{ub} the fraction of the resources of base station b assigned to user u, leading to a throughput for user u of $r_u = f_{ub}C_b$. We also assume that users are distributed among base stations with fixed probability P_b. Without loss of generality, unless otherwise stated we will assume uniform distributions: that is, a given user $u \in \mathcal{U}$ is associated with base station $b \in \mathcal{B}$ with $P_b = 1/|\mathcal{B}|$.

Traffic model. In this paper we focus on elastic and inelastic traffic. We let $\mathcal{I}$ denote the set of users that demand inelastic traffic, and $\mathcal{E}$ the set of users that demand elastic traffic. Inelastic users required a certain fixed throughput demand R_i which needs to be always satisfied with a fixed predetermined (small) outage probability P_{out}. In contrast to inelastic users, elastic users do not require any instantaneous throughput guarantees, but only average ones: they require that their expected average throughput over long time scales is above a certain threshold R_e. At any given point in time, the resources of each base stations are distributed among associated users as follows: inelastic users $u \in \mathcal{I}$ are provided sufficient resources to guarantee $r_u = R_i$, while the remaining resources are equally shared among the elastic users. In case there are not sufficient resources to satisfy the requirements of inelastic users, even when leaving elastic users with no throughput, we reject as many inelastic users as needed to satisfy the required throughput guarantees of the remaining ones.

Network slice model. The network is logically divided in different network slices, each of them belonging to a tenant. A network slice is characterized by (i) its traffic type (elastic or inelastic), and (ii) its number of users. When owning the corresponding network slice, a tenant is guaranteed that as long as he does not introduce more users than allowed by the slice size, its users will be provided with the service guarantees corresponding to their traffic type. While a network slice may be restricted to a certain geographical region (in which case the corresponding guarantees only apply to the users residing in the region), in this paper we focus on the general case in which network slices comprise the entire network.

In order to dynamically allocate network slices to tenants, we consider a bidding system in which tenants submit requests for network slices to the infrastructure provider, which may or may not accept these requests depending on the current load. Such network slices requests are characterized by:

- Network slice duration t: this is the length of the time interval for which the network slice is requested.
- Traffic type κ: according to the traffic model above, the traffic type of a slice can either be elastic or inelastic traffic.
- Network slice size s: the size of the network slice is given by the number of users it should be able to accommodate.
- Price ρ: the cost a tenant has to pay for acquiring resources for a network slice. The price is per time unit, and hence the total cost is given by $r = \rho t$.

The infrastructure provider defines a given set of network slice classes, each of them with predefined values for $\{\kappa, s, \rho\}$. When requiring network resources, a tenant may issue a request for a slice of one of the available classes, indicating the duration t for which it wishes to use the slice. Upon receiving a request, the infrastructure provider needs to decide whether to admit it or not, depending on the network slices already admitted. For each class c of network slices, we assume that requests are issued following a Poisson process of rate λ_c and t values follow an exponential random variable of rate μ_c.

[1] The users of the network are the *end-users* we referred to above, each of them being served by one of the *tenants*.

III. Performance Analysis

The rest of the paper focuses on the algorithms required by the infrastructure provider. Upon receiving a network slice request, the infrastructure provider needs to decide whether to admit it or not. While the goal of the infrastructure provider when doing so is to maximise the revenue it gets from the network, it also needs to know whether admitting a certain request would infringe the guarantees provided to the already admitted requests; indeed, if that was the case, the new request would have to be rejected regardless of any revenue considerations. We refer to the different combination of requests that can be admitted while satisfying all traffic guarantees as the *admissibility region* $\mathcal{A}$ of the system. In the following, we provide an analysis to determine this admissibility region as a first step towards the design of an algorithm to maximise the infrastructure provider revenue. Note that the algorithm addressed focuses on the admission of *slices*, in contrast to traditional algorithms focusing on the admission of *users*; once a tenant gets its slice admitted and instantiated, it can implement whatever algorithm it considers more appropriate to admit users into the slice.

A. Admissibility region

Let $|\mathcal{E}|$ be the number of elastic users in the system, and $|\mathcal{I}|$ the number of inelastic users. We say that a given combination of inelastic and elastic users belongs to the admissibility region (i.e., $\{|\mathcal{I}|, |\mathcal{E}|\} \in \mathcal{A}$) when the guarantees identified in the above section for elastic and inelastic traffic are satisfied for such combination of users in the network.

In order to determine whether the combination $\{|\mathcal{I}|, |\mathcal{E}|\}$ belongs to $\mathcal{A}$, we proceed as follows. Let $\mathcal{I}_b$ be the number of inelastic users associated to base station b. According to the system model explained in Section II, when we have a number of elastic and inelastic users at a given base station, inelastic users are provided with a fixed throughput equal to R_i independently of the number of elastic users in the base station. The only case in which they are not provided with this throughput is when the number of inelastic users itself is too large, i.e., when it exceeds $\lfloor C_b/R_i \rfloor$. Since (according to our inelastic traffic model) the probability that this happens cannot exceed P_{out}, we have the following constraint:

$$P(r_u < R_i) = P\left(|\mathcal{I}_b| > \left\lfloor \frac{C_b}{R_i} \right\rfloor\right) \leq P_{out}, \quad u \in \mathcal{I}_b$$

According to our network model, users associate to base stations with a fixed probability $1/|\mathcal{B}|$, therefore the number of inelastic users at a base station follows a binomial distribution. Hence, the probability that this number exceeds a certain threshold can be computed from

$$P\left(|\mathcal{I}_b| > \left\lfloor \frac{C_b}{R_i} \right\rfloor\right) = 1 - \sum_{j=0}^{\left\lfloor \frac{C_b}{R_i} \right\rfloor - 1} \binom{|\mathcal{I}|}{j} \left(\frac{1}{|\mathcal{B}|}\right)^j \left(1 - \frac{1}{|\mathcal{B}|}\right)^{|\mathcal{I}|-j}$$

Based on the above, the maximum number of inelastic users that can be admitted to the system, I_{max}, can be obtained from computing the largest $|\mathcal{I}|$ value that satisfies the following inequality

$$1 - \sum_{j=0}^{\left\lfloor \frac{C_b}{R_i} \right\rfloor - 1} \binom{|\mathcal{I}|}{j} \left(\frac{1}{|\mathcal{B}|}\right)^j \left(1 - \frac{1}{|\mathcal{B}|}\right)^{|\mathcal{I}|-j} \leq P_{out}$$

Note that I_{max} is independent of the number of elastic users in the network: indeed, inelastic users preempt elastic ones and receive the same throughput independent of the number of elastic users present in the network.

Having computed the maximum number of inelastic users that can be admitted, we now compute the maximum admissible number of elastic users, E_{max}. In contrast to the previous case, in this case the throughput available to elastic users depends on the number of inelastic users, and hence E_{max} will depend on the number of inelastic users admitted into the network, $\mathcal{I}$. Our key approximation when computing E_{max} will be to assume that the density of elastic users is sufficiently high so that the probability that there are no elastic users in a base station can be neglected. Note that, as elastic users consume as much throughput as possible, this assumption implies that the capacity of all base stations will always be fully used, i.e., $\sum_{u \in \mathcal{I} \cup \mathcal{E}} r_u = C$. Since inelastic users consume a fixed throughput equal to R_i, this yields $\sum_{u \in \mathcal{E}} r_u = C - |\mathcal{I}|R_i$ for elastic users. Over long time scales, all elastic users receive the same average throughput, and hence

$$r_u = \frac{C - |\mathcal{I}|R_i}{|\mathcal{E}|}$$

If we now impose the constraint on the average throughput of an elastic users, $r_u \geq R_e$, and compute from the above equation the maximum number of elastic users that can be admitted while satisfying this constraint, we obtain the following expression (which depends on the number of admitted inelastic users):

$$E_{max}(|\mathcal{I}|) = \left\lfloor \frac{C - |\mathcal{I}|R_i}{R_e} \right\rfloor$$

From the above, we have that the admissibility region $\mathcal{A}$ is given by all the combinations of inelastic and elastic users $\{|\mathcal{I}|, |\mathcal{E}|\}$ that satisfy: (i) $|\mathcal{I}| \leq I_{max}$; and (ii) $|\mathcal{E}| \leq E_{max}(|\mathcal{I}|)$. This terminates the admissibility region analysis.

In order to evaluate the above analysis, we compared the admissibility region obtained theoretically against the one resulting from simulations. To this end, we considered the scenario of ITU-T [11], which consists of $|\mathcal{B}| = 19$ base stations placed at a fixed distance of 200m. Users move in this area covered by these base stations following the Random Waypoint (RWP) mobility model, with a speed uniformly distributed between 2 and 3 m/s.

The association procedure of elastic and inelastic users with base stations is as follows. Inelastic users $u \in \mathcal{I}$ try to attach to the nearest base station $b \in \mathcal{B}$, if it has at least R_i capacity left. Otherwise they do not associate and generate an outage event,

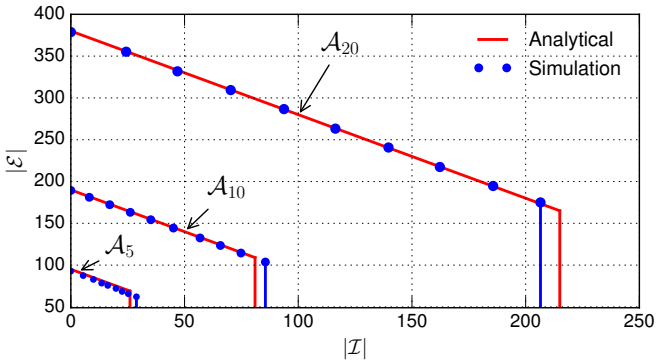


Fig. 1: Admissibility region: analysis vs. simulation.

joining again the network when their throughput guarantee can be satisfied. When associating, they consume a capacity R_i from the base station. Similarly, elastic users always associate to the nearest base station. All the elastic users associated with a base station, $u \in \mathcal{E}_b$, fairly share among them the capacity left over by inelastic users. Upon any association event, the throughput received by the users associated to the new and the old base station changes accordingly.

Following the above procedure, we have simulated all the possible combinations of inelastic and elastic users, $\{\mathcal{I}, \mathcal{E}\}$. For each combination, we have evaluated the average throughput received by elastic users, computed over samples of 10 seconds time windows, and the outage probability P_{out} of inelastic users, computed as the fraction of time over which they do not enjoy their guaranteed throughput. If these two metrics (average elastic traffic throughput and inelastic traffic outage probability) are within the guarantees provided to the two traffic types, we place this combination inside the admissibility region, and otherwise we place it outside.

Fig. 1 shows the boundaries of the admissibility region obtained analytically and via simulation, respectively, for different throughput guarantees for elastic and inelastic users ($\mathcal{A}_5 : R_i = R_e = C_b/5$, $\mathcal{A}_{10} : R_i = R_e = C_b/10$ and $\mathcal{A}_{20} : R_i = R_e = C_b/20$) and $P_{out} = 0.01$. We observe that simulation results closely match analytical ones, which confirms the accuracy of our analysis. We further observe that the admissibility region is limited by the most restrictive of the following conditions: (i) the maximum number of inelastic users that can be admitted, (ii) the sum of inelastic and elastic users, which has to be below a maximum number as well.

IV. Optimal admission algorithm

While the admissibility region computed above provides the maximum number of elastic and inelastic users that can be admitted, an optimal admission algorithm that aims at maximising the revenue of the infrastructure provider may not always admit all the requests that fall within the admissibility region. Indeed, when the network is close to congestion, admitting a request that provides a lower revenue may prevent the infrastructure provider from admitting a future request with a higher revenue associated. Therefore, the infrastructure provider may be better off by rejecting the first request with the hope that a more profitable one will arrive in the future.

In the following, we derive the optimal admission policy that maximises the revenue of the infrastructure provider. We first present an analysis of the revenue obtained by the infrastructure provider as a function of the admission policy, and then build on this analysis to find the optimal admission policy.

A. Revenue analysis

To analyse the revenue obtained by the infrastructure provider, we model our system as a Semi-Markov Decision Process (SMDP).[2] For simplicity, we first model our system for the case in which there are only two classes of slice requests of fixed size $s = 1$, i.e., for one elastic user or for one inelastic user. Later on, we will show how the model can be extended to include an arbitrary set of network slice requests of different sizes.

The Markov Decision Process theory [14] models a system as: (i) a set of states $s \in S$, (ii) a set of actions $a \in A$, (iii) a transition function $P(s, a, s')$, (iv) a time transition function $T(s, a)$, and (v) a reward function $R(s, a)$. At each event, the system can be influenced by taking one of the possible actions $a \in A$. According to the chosen actions, the system earns the associated reward function $R(s, a)$, the next state is decided by $P(s, a, s')$ while the transition time is defined by $T(s, a)$.

The inelastic and elastic network slices requests follow two Poisson processes $\mathcal{P}_i$ and $\mathcal{P}_e$ with associated rates of λ_i and λ_e, respectively. When admitted into the system, the slices occupy the system resources according to an exponentially distributed time of average $\frac{1}{\mu_i}$ and $\frac{1}{\mu_e}$. Additionally, they generate a revenue per time unit for the infrastructure provider of ρ_i and ρ_e. That is, the total revenue r generated by an e.g., elastic request with duration t is $t\rho_e$.

We define our space state S as follows. A state $s \in S$ is a three-sized tuple $(n_i, n_e, k \mid n_i, n_e \in \mathcal{A})$ where n_i and n_e are the number of inelastic and elastic slices in the system at a given decision time t, and $k \in \{i, e, d\}$ is the next event that triggers a decision process. This can be either a new arrival of a network slice request for inelastic and elastic slices ($k = i$ and $k = e$, respectively), or a departure of a network slice of any kind that left the system ($k = d$). In the latter case, n_i and n_e represent the number of inelastic and elastic slices in the system after the departure. Fig. 2 shows how the space state S relates to the admissibility region $\mathcal{A}$.

The possible actions $a \in A$ are the following: $A = G, D$. The action G corresponds to admitting the new request of an elastic or inelastic slice; in this case, the resources associated with the request are granted to the tenant and the revenue $r = \rho_{i,e}t$ is immediately earned by the infrastructure provider. In contrast, action D corresponds to rejecting the new request; in this case, there is no immediate reward but the resources remain free for future requests. Note that upon a departure

[2]This technique has already been used to address different admission control problems (see, e.g., [12], [13]).

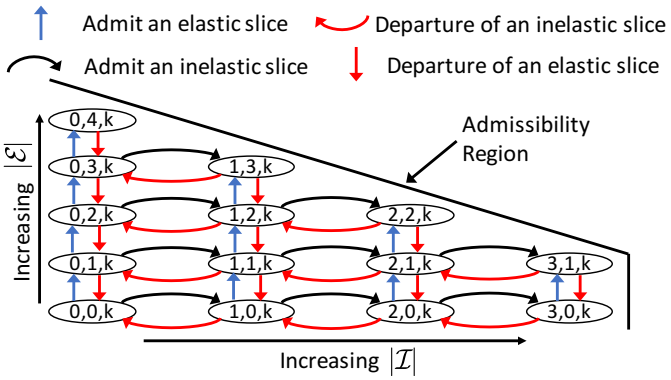


Fig. 2: Example of system model with the different states.

($k = d$), the system is forced to a fictitious action D that involves no revenue. Furthermore, we force that upon reaching a state in the boundary of the admissibility region computed in the previous section, the only available action is to reject an incoming request ($a = D$) as otherwise we would not be meeting the committed guarantees. Requests that are rejected are lost forever.

The transition rates between the states identified above are derived next. Transitions to a new state with $k = i$ and $k = e$ happen with a rate λ_i and λ_e, respectively. Additionally, states with $k = d$ are reached with a rate $n_i\mu_i + n_e\mu_e$ depending the number of slices already in the system. Thus, the average time the system stays at state s, $\bar{T}(s,a)$ is given by

$$\bar{T}(s,a) = \frac{1}{\upsilon(n_i, n_e)}$$

where n_i, and n_e are the number of inelastic and elastic slices in state s and $\upsilon(n_i, n_e) = \lambda_i + \lambda_e + n_i\mu_i + n_e\mu_e$.

We define a policy $\pi(S)$, $\pi(s) \in A$, as a mapping from each state s to an action A. Thus, the policy determines whether, for a given number of elastic and inelastic slices in the system, we should admit a new request of an elastic or an inelastic slice. With the above analysis, given such a policy, we can compute the probability of staying at each of the possible states. Then, the long term average revenue R obtained by the infrastructure provider can be computed as

$$R = P(n_i, n_e, k)(n_i\rho_i + n_e\rho_e)$$

where ρ_i and ρ_e are the price per time unit paid by an inelastic and an elastic network slice, respectively.

Our ultimate goal is to find a policy $\pi(S)$ that maximises the long term average revenue. In the next section, we build on the analysis provided here to find such an optimal policy.

B. Optimal policy

In order to derive the optimal policy, we build on Value Iteration [15], which is an iterative approach to find the optimal policy that maximises the average revenue of the system. According to the revenue analysis provided in the previous section, our system has the following transition probabilities $P(s,a,s')$. For $a = D$, $\forall s$:

$$P(s,a,s') = \begin{cases} \frac{\lambda_i}{\upsilon(n_i,n_e)}, & s' = (n_i, n_e, i) \\ \frac{\lambda_e}{\upsilon(n_i,n_e)}, & s' = (n_i, n_e, e) \\ \frac{n_i\mu_i}{\upsilon(n_i,n_e)}, & s' = (n_i - 1, n_e, d) \\ \frac{n_e\mu_e}{\upsilon(n_i,n_e)}, & s' = (n_i, n_e - 1, d) \end{cases}$$

For $a = G$, $s = (n_i, n_e, i)$:

$$P(s,a,s') = \begin{cases} \frac{\lambda_i}{\upsilon(n_i+1,n_e)}, & s' = (n_i + 1, n_e, i) \\ \frac{\lambda_e}{\upsilon(n_i+1,n_e)}, & s' = (n_i + 1, n_e, e) \\ \frac{(n_i+1)\mu_i}{\upsilon(n_i+1,n_e)}, & s' = (n_i, n_e, d) \\ \frac{n_e\mu_e}{\upsilon(n_i+1,n_e)}, & s' = (n_i + 1, n_e - 1, d) \end{cases}$$

For $a = G$, $s = (n_i, n_e, e)$:

$$P(s,a,s') = \begin{cases} \frac{\lambda_i}{\upsilon(n_i,n_e+1)}, & s' = (n_i, n_e + 1, i) \\ \frac{\lambda_e}{\upsilon(n_i,n_e+1)}, & s' = (n_i, n_e + 1, e) \\ \frac{n_i\mu_i}{\upsilon(n_i,n_e+1)}, & s' = (n_i - 1, n_e + 1, d) \\ \frac{(n_e+1)\mu_e}{\upsilon(n_i,n_e+1)}, & s' = (n_i, n_e, d) \end{cases}$$

Similarly, the reward function $R(s,a)$ is given by:

$$R(s,a) = \begin{cases} 0, & a = D \\ t\rho_{i,e} & a = G \end{cases}$$

Applying the Value Iteration algorithm [15] for SMDP is not straightforward. The standard algorithm cannot be applied to continuous time problem as it does not consider variable transition times between states. Therefore, in order to apply Value Iteration to our system, an additional step is needed: all the transition times need to be normalized as multiples

Algorithm 1 Value Iteration

1) Initialise the vector $V(s) = 0, \forall s \in S$. $V(s)$ represents the long term expected revenue for being in state s. Initialise the step number n to 1.
2) Update the expected reward at time $n+1$, $V_{n+1}(s)$ using the rule

$$V_{n+1}(s) = \max_{a \in A}\left[\frac{R(s,a)}{T(s,a)} + \frac{\tau}{T(s,a)}\sum_{s'} P(s,a,s')V_n(s') + \left(1 - \frac{\tau}{T(s,a)}\right)V_n(s)\right] \quad \forall s \in S$$

3) Compute the boundaries

$$M_n = \max_{s \in S}(V_{n+1}(s) - V_n(s))$$

$$m_n = \min_{s \in S}(V_{n+1}(s) - V_n(s))$$

and check the condition

$$0 \leq (M_n - m_n) \leq \epsilon m_n$$

4) If the condition in step 3 is not fulfilled, then repeat from step 2

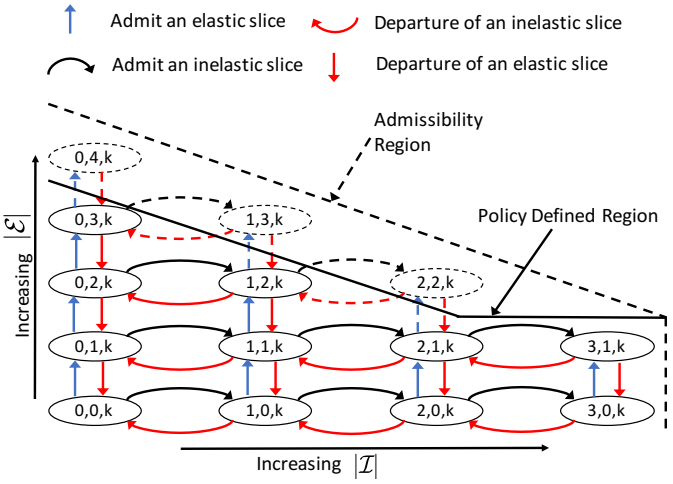


Fig. 3: Example of optimal policy for elastic and inelastic slices.

of a faster, arbitrary, fixed transition time τ [16]. The only constraint that has to be satisfied by τ is that it has to be faster than any other transition time in the system, which leads to

$$\tau < \min T(s,a), \qquad \forall s \in S, \forall a \in A$$

With the above normalization, the continuous time SMDP corresponding to the analysis of the previous section becomes a discrete time Markov Process and a modified Value Iteration algorithm may be used to devise the best policy $\pi(S)$ (see Algorithm 1). The discretised Markov Chain will hence be composed of transitions (at interval of τ) that may correspond to transitions in continuous time system or not.

The normalization procedure affects the update rule of step 2 in Algorithm 1. All the transition probabilities $P(s,a,s')$ are scaled by a factor $\frac{\tau}{T(s,a')}$ making them time-independent. Also, the revenue $R(s,a)$ is scaled by a factor of $T(s,a)$ to take into account the transitions in the sampled discrete time system that do not correspond to transitions in the continuous time one. This fact is taken into account in the last term of the equation (i.e., in case of a fictitious transition, keep $V_n(s)$).

As proven in the next section, Algorithm 1 is guaranteed to find the optimal policy $\pi(S)$. Such an optimal policy is illustrated in Fig. 3 for the case where the price of inelastic slice is higher than that of elastic slice ($\rho_i > \rho_e$). The figure shows those states for which the corresponding action is to admit the new request (straight line), and those for which it is to reject it (dashed lines). It can be observed that while some of the states with a certain number of elastic slices fall into the admissibility region, the system is better off rejecting those requests and waiting for future (more rewarding) requests of inelastic slice. In contrast, inelastic slice requests are always admitted (within the admissibility region).

The analysis performed so far has been limited to network slice requests of size one. In order to extend the analysis to requests of an arbitrary size, we proceed as follows. We set the space state to account for the number of slices of each different class in the system (where each class corresponds to a traffic type and a given size). Similarly, we compute the transition probabilities $P(s,a,s')$ corresponding to arrival and departures of different classes. With this, we can simply apply the same procedure as above (over an extended space state) to obtain the optimal policy.

C. Optimality and convergence analysis

In the following, we provide some insights on the optimality and convergence of Algorithm 1, showing that: (i) the algorithm converges to a certain policy, and (ii) the policy to which the algorithm converges performs arbitrarily close to the optimal policy. Theorem 6.6.1 in [17] proves that the policy $\pi(S)$ obtained using Algorithm 1 provides a long-run average reward $g_s(\pi(S))$ that is arbitrarily bounded by an ϵ value when compared to the optimal one g^*. Thus,

$$0 \leq \frac{g^* - g_s(\pi(S))}{g_s(\pi(S))} \leq \frac{M_n - m_n}{m_n} \leq \epsilon, \qquad \forall s \in S$$

The convergence of Algorithm 1 is guaranteed by the third term of the inequality above, that acts as a decreasing envelope of the second term, as shown by Theorem 6.6.3 in [17]:

$$m_{n+1} \geq m_n, \quad M_{n+1} \leq M_n, \qquad \forall n \geq 1.$$

By applying step 3 of Algorithm 1, the obtained $\pi(S)$ is ϵ-bounded to the optimal. While the aforementioned Theorems in [17] solve a cost minimisation problem, we adapted them to our revenue maximisation scenario. In our experiments, we set $\epsilon = 0.001$.

V. Adaptive Admission Algorithm

The Value Iteration algorithm described in Section IV-B provides the optimal policy for revenue maximisation under the framework described of Section IV-A. While this is very useful in order to obtain a benchmark for comparison, the algorithm itself has a very high computational cost, which makes it impractical for real scenarios. Indeed, as the algorithm has to update all the V values $V(s)$, $s \in S$ at each step, the running time grows steeply with the size of the state space, and may become too high for large scenarios.

Building on the analysis of the previous section, in the following we design an adaptive algorithm that aims at maximising revenue by learning from the outcome resulting from the previous decisions. In contrast the optimal policy of the previous section, this algorithm is executed online while taking admission control decisions, and hence does not require of high computational resources.

A. Q-Learning model

Our adaptive algorithm is based on the Q-Learning framework [18]. Before describing the algorithm itself, we describe how we model the algorithm under the Q-Learning framework.

Q-Learning is a machine learning framework for designing adaptive algorithms in SMDP-based systems such as the one analysed in Section IV-A. It works taking decisions that move the system to different states within the SMDP state-space and observing the outcome. Thus, it leverages the "exploration

vs. exploitation" principle: the algorithm learns by visiting unvisited states and takes the optimal decision when dealing with already visited ones.

Q-Learning provides two key advantages as compared to Value Iteration framework described in the previous section:

- The resulting algorithm is *model-free*. Indeed, it makes no assumptions on the underlying stochastic processes, but rather learns by observing the events that take place in the system.
- It is an *online* algorithm (in contrast to the offline planning algorithm of the previous section). The algorithm constantly learns the characteristics of the system by exploring it and taking decisions.

Our Q-Learning framework builds on the SMDP-based system model of Section IV-A. The Q-Learning space state is similar to the one of the SMDP model:

$$(n_i^\star, n_e^\star, k \mid o\left(n_i^\star, n_e^\star\right) \in \mathcal{A})$$

where $n_i^\star$ and $n_e^\star$ are defined as a n-dimension tuples $(n_1, n_2, \ldots, n_c)$ describing the number of slices of different sizes in the system for inelastic and elastic traffic types. Analogously, o is the occupation of the system, and $k \in \{i^\star, e^\star\}$ where $i^\star$ and $e^\star$ are the sets of events associated to an arrival of an inelastic or elastic slice request of a given size.

With Q-Learning, we do not need to include departures in the space state, since no decision is taken upon departures. Similarly, we do not need to include the states in the boundary of the admissibility region; indeed, in such states we do not have any option other than rejecting any incoming request, and hence no decisions need to be taken in these states either. Furthermore, the system is not sampled anymore, as all transitions are triggered by an arrival event and the subsequent decision $a \in A$.

The key idea behind the Q-Learning framework is as follows. We let $Q\left(s,a\right)$ denote the expected reward resulting from taking an action a at a certain state s. The system keeps memory for each state of $Q\left(s,a\right)$. It starts with empty $Q_0\left(s,a\right)$ and at the decision step n it takes an action a based on the past estimations of $Q\left(s,a\right)$. Hence, the system experiences a transition from state s at the decision step n, to state s' at decision step $n+1$. Then, once in step $n+1$, the algorithm has observed both the reward obtained during the transition $R\left(s,a\right)$ and a sample t_n of the transition time. The, algorithm updates the $Q\left(s,a\right)$ involved in the decision process at step n using the newly gathered reward and transition time information. After a learning phase, the optimal admission policy at a certain state will be the one that maximises the resulting expected revenue, i.e.,

$$V\left(s\right) = \max_{a \in A} Q\left(s,a\right)$$

B. Algorithm description

Building on the above model, we describe our Q-Learning algorithm in the following. The algorithm maintains the Q-values which are updated iteratively following a sample-based approach as described in Algorithm 2, in which new events are evaluated at the time when they happen. In addition to the procedure to update the Q-values described in Algorithm 2, the Q-Learning algorithm also relies on two other procedures: the *TD-learning* and *exploration - exploitation* procedures.

Algorithm 2 Q-Learning update procedure

1) Initialise the vector $Q\left(s,a\right) = 0,\ \forall s \in S,\ a \in A$.
2) An event is characterized by: s, a, s', r, t (the starting state, the action taken, the landing state, the obtained reward and the transition time).
3) Update the old estimate $Q\left(s,a\right)$ with the new sample observation as follows:
$$\omega = R\left(s,a,s'\right) - \sigma t_n + \max_{a'} Q\left(s',a'\right)$$
where t_n is the transition time between two subsequent states s and s' after action a
4) Integrate the new sample in a running exponential average estimation of $Q\left(s,a\right)$
$$Q\left(s,a\right) = (1-\alpha)Q\left(s,a\right) + \alpha\omega$$

TD-learning ensures the convergence of the algorithm by employing the α parameter, which is the learning rate. The requirements for setting α are two [19]: (i) $\sum_{n=0}^{\infty} \alpha_n = \infty$ and (ii) $\sum_{n=0}^{\infty} \alpha_n^2 < \infty$. The Q-values update process in step 4 of Algorithm 2 needs to build a correct estimation of the expected revenue obtained by choosing an action a while in state s. On the one hand, new samples ω (with more updated information) should be weighted by a larger weight than the estimation built on all the past samples $Q\left(s,a\right)$, especially if the first exploration steps did not provide a good result. On the other hand, α_n coefficients have to decrease with time, in order to eventually converge to a fixed set of $Q\left(s,a\right)$ values. When setting α according to these requirements, we make the following additional considerations: too slowly descending α sequences will delay the convergence of the algorithm, but too fast ones may make the algorithm unaware of new choices too soon. Based on all these requirements and considerations, we set $\alpha = \frac{0.5}{\eta(s,a)}$, where $\eta\left(s,a\right)$ is the number of times the action a was selected, being in state s.

Exploration - exploitation drives the selection of the best action to be taken at each time step. While choosing the action a that maximises the revenue at each step contributes to maximising the overall revenue (i.e., *exploitation* step), we also need to visit new (still unknown) states even if this may lead to a suboptimal revenue (i.e., *exploration* step). The reason for this is that the algorithm needs to explore all possible (s,a) options in order to evaluate the impact of the different decisions. The trade-off between exploitation and exploration is regulated by the γ parameter; in this paper we take $\gamma = 0.1$ in order to force that sometimes the wrong decision is taken and thus we learn all possible options, which ultimately improves the accuracy of the algorithm. The probability of taking wrong choices decreases as the α_n values become smaller, up to the point where no wrong decisions are taken any more, once the algorithm already visited all state s a number of times sufficiently large to learn the best Q-value.

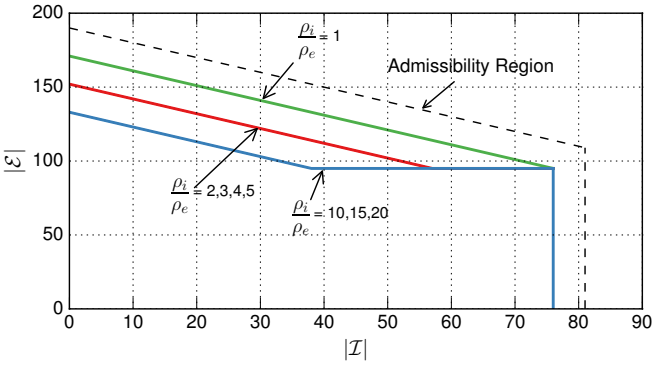


Fig. 4: Optimal admission policy for elastic traffic.

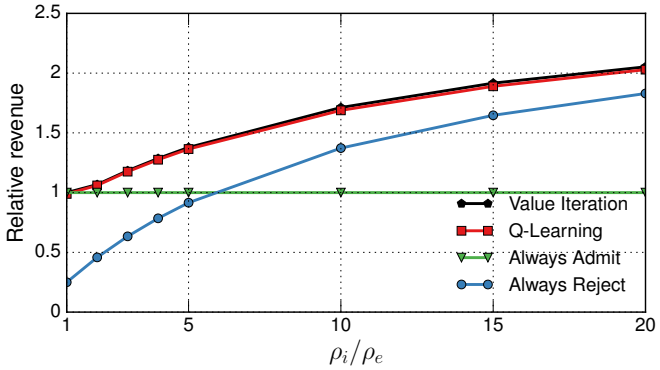


Fig. 5: Revenue vs. ρ_i/ρ_e.

VI. Performance Evaluation

In this section we evaluate the performance of the proposed algorithms via simulation. Unless otherwise stated, we consider a scenario with four slice classes, two for elastic traffic and two for inelastic. We set $\mu = 5$ for all network slices classes, and the arrival rates equal to $\lambda_i = 2\mu$ and $\lambda_e = 10\lambda_i$ for the elastic and inelastic classes, respectively. We consider two network slice sizes, equal to $C/10$ and $C/20$, where C is the total network capacity. Similarly, we set the throughput required guarantees for elastic and inelastic traffic to $R_i = R_e = C_b/10$. Two key parameters that will be employed throughout the performance evaluation are ρ_e and ρ_i, the average revenue per time unit generated by elastic and inelastic slices, respectively (in particular, performance depends on the ratio between them).

A. Optimal admission policy

We start by analysing the admission policy resulting from our optimal algorithm for different ratios between ρ_i and ρ_e. Note that, given that inelastic traffic is more demanding, it is reasonable to assume that it pays a higher price than elastic traffic $\rho_i \geq \rho_e$. As inelastic traffic provides a higher revenue, in order to maximise the total revenue, the infrastructure provider will always admit inelastic network slice requests. In contrast, it is to be expected that, while elastic traffic requests will be admitted when the utilisation is low, they may be rejected with higher utilisations in order to avoid losing the opportunity to admit future (and more rewarding) inelastic requests. Furthermore, it is to be expected that this behaviour will be exacerbated as the ρ_i/ρ_e grows larger.

The admission policy for elastic traffic resulting from our algorithm is shown in Fig. 4. As expected, we can observe that the region corresponding to the admission of elastic network slices requests is smaller than the admissibility region, implying that we are more restrictive in the admission of elastic traffic. Furthermore, and also as expected, this region becomes smaller for larger ρ_i/ρ_e ratios. These results thus confirm our intuitions on the optimal admission policy.

B. Revenue optimality

We next evaluate the performance of our adaptive algorithm by comparing it against: (i) the benchmark provided by the optimal algorithm, and (ii) two naive policies that always admit elastic traffic requests and always reject them, respectively. Fig. 5 shows the relative average reward obtained by each of this policies, taking as baseline the policy that always admit all network slice requests (as this would be the most straightforward algorithm). We observe from the figure that our adaptive algorithm performs very closely to the optimal policy, which serves to validate the algorithm design proposed in this paper. We further observe that the revenue improvements over the naive policies is very substantial, up to 100% in some cases. As expected, for small ρ_i/ρ_e the policy that always admits all requests is optimal, as in this case both elastic and inelastic slices provide the same revenue; in contrast, for very large ρ_i/ρ_e ratios the performance of the "always reject" policy improves, as in this case the revenue obtained from elastic traffic is (comparatively) much smaller.

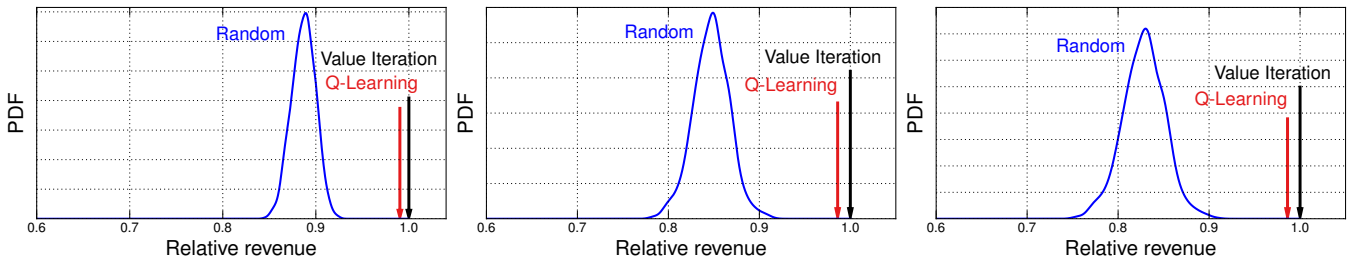
C. Revenue gains

While the result of the previous section shows that the proposed algorithm performs close to optimal, it is only compared against two naive policies and thus does not give an insight on the revenue gains that could be achieved over smarter yet not optimal policies. To this end, we compare the performance of our algorithm against a set of "smart" random policies defined as: inelastic network slices requests are always accepted ($k = i \Rightarrow a = G$), while the decision of rejecting an elastic request ($k = e \Rightarrow a = D$) is set randomly. Then, by drawing a high number of random policies, it is to be expected that some of them provide good performance.

Fig. 6 shows the comparison against 1000 different random policies. The results confirm that (i) none of the random policies outperforms our approach, further confirming the optimality of the approach, and (ii) substantial gains (around 20%) are obtained over the random policies. This result confirms that a smart heuristic is not effective in optimizing revenue, and very substantial gains can be achieved by using a close to optimal policy such as our adaptive algorithm.

D. Impact of estimation errors

The previous results have assumed that (i) arrivals and departures follow Poisson process with exponential times, and (ii) the optimal algorithm has a perfect estimation of the



(a) $\rho_i/\rho_e = 5$ (b) $\rho_i/\rho_e = 10$ (c) $\rho_i/\rho_e = 15$

Fig. 6: The distribution of the revenues obtained by random smart policies compared to the proposed algorithms.

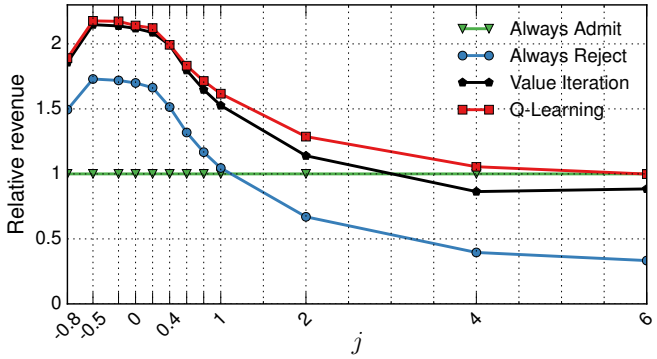


Fig. 7: Revenue in a perturbed scenario, $\rho_i/\rho_e = 5$.

statistics of this process. In this section we address a more realistic case in which neither of these assumption holds. We hence introduce two modifications: (i) arrivals and departures are Pareto-distributed, and (ii) we let the real arrival process $\hat{\lambda}$ deviate from the estimated one λ: $\hat{\lambda}(j) = \frac{\lambda}{j+1}$ as a function of a parameter $j > -1$. That is, the optimal policy obtained by Value Iteration under the original assumptions is computed offline, with the estimated parameter, and applied to the real system. Note that for negative j values, the system receives a number of request per time unit higher than the estimated λ, while positive j values indicate a lower requests arrival rate.The results, depicted in Fig. 7, show that our adaptive algorithm, which automatically learns the network slice behaviour on the fly and hence is not affected by possible estimation errors, substantially outperforms the optimal policy built upon flawed assumptions and estimations.

VII. Conclusion

One of the key concepts behind 5G is the network slicing model, which brings new players into the ecosystem: the network infrastructure provider and the network slices' tenants. Under this new business model, we need new resource allocation mechanisms that take into account the relationship between the various players. In this paper, we have addressed this issue by designing an admission control algorithm to be executed by the infrastructure provider when receiving network slice requests from the tenants. Building on an analytical model for the performance and revenue of the system, we have provided an optimal policy benchmark and an adaptive algorithm for practical usage. Our results show that our adaptive algorithm approximates the performance of the optimal policy, and provides substantial gains in revenue over potentially smart heuristics.

Acknowledgments

This research work has been performed in the framework of the H2020-ICT-2014-2 project 5G NORMA (Grant Agreement No. 671584). The work of A. Banchs was partially supported by the Spanish Ministry of Economy and Competitiveness under the THWART project (Grant TEC2015-70836-ERC).

References

[1] 5G PPP Architecture WG, "View on 5g architecture," White Paper, 2016.
[2] NGMN Alliance, "Description of network slicing concept," Public Deliverable, 2016.
[3] 3GPP, "Study on architecture for next generation system," TR 23.799, v0.5.0, May 2016.
[4] NGMN Alliance, "5g white paper," White Paper, Feb. 2015.
[5] X. Zhou, R. Li, T. Chen, and H. Zhang, "Network slicing as a service: enabling enterprises' own software-defined cellular networks," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, Jul. 2016.
[6] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5g network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, Jul. 2016.
[7] A. Gudipati, L. Li, and S. Katti, "Radiovisor: A slicing plane for radio access networks," in *Proc. of ACM HotSDN*, Aug. 2014.
[8] I. Malanchini, S. Valentin, and O. Aydin, "Generalized resource sharing for multiple operators in cellular wireless networks," in *Proc. of IEEE IWCMC*, Aug. 2014.
[9] R. Mahindra, M. A. Khojastepour, H. Zhang, and S. Rangarajan, "Radio access network sharing in cellular networks," in *Proc. of IEEE ICNP*, Oct. 2013.
[10] S. Rathinakumar and M. Marina, "Gavel: Strategy-proof ascending bid auction for dynamic licensed shared access," in *Proc. of ACM MobiHoc*, Jul. 2016.
[11] ITU-R, "Guidelines for evaluation of radio interface technologies for imt-advanced," Report ITU-R M.2135-1, 2016.
[12] T. O. Kim, C. N. Devanarayana, A. S. Alfa, and B. D. Choi, "An optimal admission control protocol for heterogeneous multicast streaming services," *IEEE Transactions on Communications*, vol. 63, no. 6, pp. 2346–2359, Jun. 2015.
[13] J. Buhler and G. Wunder, "Traffic-aware optimization of heterogeneous access management," *IEEE Transactions on Communications*, vol. 58, no. 6, pp. 1737–1747, Jun. 2010.
[14] R. Bellman, "A markovian decision process," DTIC, Tech. Rep., 1957.
[15] R. Howard, *Dynamic Programming and Markov Processes*. Technology Press-Wiley, 1960.
[16] S. Lippman, "Applying a new device in the optimization of exponential queuing systems," *Operation Research*, vol. 23, no. 4, pp. 687–710, Aug. 1975.
[17] H. Tijms, *A First Course in Stochastic Models*. J. Wiley & Sons, 2003.
[18] C. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
[19] E. Even-Dar and Y. Mansour, "Learning rates for q-learning," *Journal of Machine Learning Research*, vol. 5, pp. 1–25, Dec. 2003.