A Lexicographic Approach to Constrained MDP Admission Control

M. Panfili, A. Pietrabissa, G. Oddi, V. Suraci

Department of Computer, Control and Management Engineering "Antonio Ruberti", University of Rome "Sapienza", via Ariosto 25, 00185, Rome, Italy

M. Panfili and A. Pietrabissa are joint first authors.

A. Pietrabissa is the corresponding author:

e-mail: pietrabissa@diag.uniroma1.it

phone: 00390677274040

fax:00390677274033

A Lexicographic Approach to Constrained MDP Admission Control

This paper proposes a reinforcement learning-based lexicographic approach to the call admission control problem in communication networks. The admission control problem is modeled as a multi-constrained Markov decision process. To overcome the problems of the standard approaches to the solution of constrained Markov decision processes, based on the linear programming formulation or on a Lagrangian approach, a multi-constraint lexicographic approach is defined, and an on-line implementation based on reinforcement learning techniques is proposed. Simulations validate the proposed approach.

Keywords: Stochastic control, Markov decision processes, Reinforcement learning, Communication networks, Call admission control

1 Introduction

In recent years, the application of control-based techniques to resource management in communication networks enabled the evolution from heuristic implementations (F. Delli Priscoli, 1999) to more sophisticated approaches, e.g., just to name a few, (De Cicco, Mascolo, & Niculescu, 2011; Francesco Delli Priscoli & Pietrabissa, 2004; Manfredi, 2012). Among the resource management procedures, call admission control (CAC) is in charge of deciding upon the admission or blocking of the calls which request for transmission capacity: if the CAC algorithm blocks too many calls, the network utilization is poor as well as the revenue for the operator; if, on the contrary, the CAC algorithm admits too many calls, the quality of the call experienced by the users becomes unsatisfactory. In current communication networks, Classes of Service (CoS) are defined to differentiate the calls based on their different call priorities and characteristics. An additional task for the CAC algorithms is then the one of guaranteeing CoS-level requirements, generally expressed in terms of maximum thresholds for call blocking probabilities (Kalyanasundaram, Chong, & Shroff, 2001).

Traditionally, he CAC problem is crucial in networks characterized by a limited amount of capacity shared among the users, such as terrestrial wireless networks, e.g., 3G, 4G networks *ad-hoc* networks (see (Khoukhi, Badis, Merghem-Boulahia, & Esseghir, 2013) and the references therein), and satellite networks (Pillai, Hu, & Halliwell, 2013; Zhou, Sun, Liu, Zhang, & Xiao, 2014). However, it is a fundamental task also in emerging scenarios, such as cloud networks (Oddi, Panfili, Pietrabissa, Zuccaro, & Suraci, 2013), where economics considerations are becoming more and more relevant and, therefore, CoS differentiation is one of the key topics for telco operators.

In the literature, several methodologies have been used to cope with the CAC problem, such as queueing theory (Klessig, Fehske, & Fettweis, 2014), game theory (Altman, Boulogne, El-Azouzi, Jiménez, & Wynter, 2006), statistical approaches (Jiao, Sheng, Lui, & Shi, 2014) and so on. The CAC problem has been successfully modelled also as a Markov Decision Process (MDP) (Choi, Kwon, Choi, & Naghshineh, 2000). Among the cited approaches, the MDP formulation is the only one that is able to analytically enforce constraints on call blocking probabilities, since the class constraints can be formulated within the model itself. The cost function of the MDP formulation of the CAC problem is then aimed at minimizing the blocking probability, subject to constraints on the maximum blocking probability tolerated by each CoS.

MDPs are stochastic control processes, used for optimization problems involving random event and decision makers (Altman, 2002). To solve an unconstrained MDP, Dynamic Programming (DP) or Linear Programming (LP) algorithms can be used to compute the optimal policy (e.g., (Bertsekas, 2005; Puterman, 1994)). Beside standard DP and LP approaches, Reinforcement Learning (RL) algorithms can also be used to solve unconstrained MDPs (Sutton & Barto, 1998); even if RL approaches converge to the optimal solution only under given conditions, such as infinite number of visits of each state, in practice they achieve approximate solutions to problems which are intractable for other methods, as DP and LP approaches, that are more subject to the scalability problems of MDPs. In fact, RL is a model-free approach and it does not require *a priori* knowledge of traffic statistics, since it relies on measured statistics.

Likewise, the solution of constrained MDPs can be sought by several approaches, as discussed in (Geibel, 2007): the LP formulation (Hillier & Lieberman, 2001; Pietrabissa, 2008b, 2009b), the weighted or Lagrangian approach – also based on a LP formulation (Pietrabissa, 2011), and the lexicographic approach (Gábor, Kalmár, & Szepesvári, 1998), based on DP methods:

- In the LP formulation, CoS-level control in terms of blocking probabilities can be enforced by means of inequality constraints. The LP approach is then capable of explicitly controlling the blocking probabilities by computing an optimal admission policy which minimizes the global blocking probability subject to constraints on the maximum tolerated blocking probability of each CoS (Pietrabissa, 2008a, 2008b).
- The Lagrangian or weighted approach uses the Lagrangian relaxation of the LP problem to define an unconstrained problem, where the inequality constraints are substituted by additional cost terms in the cost function, weighted by the so-called Lagrange multipliers, and which penalize the violations of the constraints.
- Lexicographic approaches define additional cost functions to model the problem constraints; these functions are optimized along with the primary cost function with a prioritization technique.

The main drawbacks of the LP-based approaches is that they are not suitable for implementation in real network equipment, due to the scalability problem of the MDP algorithms – the so-called "curse of dimensionality", i.e., the state space explosion as the link capacity and the number of supported classes increases (Bertsekas & Tsitsiklis, 1989). The Lagrangian approach, however, thanks to the lack of constraints in the formulation, has the advantage that it can be used to develop RL admission control algorithms: the overall cost function (global blocking probabilities plus the terms weighted by the Lagrange multipliers) are directly used to build the (action,state)-dependent cost function needed by the RL algorithms. The drawback is that, unless the multipliers are obtained by solving the Lagrangian (constrained) dual problem, this method is only able to achieve a suboptimal solution: in particular, considering the CAC problem, the actual performance of these algorithms heavily depends on the choice of the multipliers, as discussed in (Pietrabissa, 2011). Finally, the lexicographic approach is used in DP algorithm, thanks to the fact that it does not require constraints; as explained in the following, also the lexicographic approach achieves suboptimal policies.

The main innovation is that this paper proposes a CAC algorithm based on lexicographic approach. We consider the Lexicographic Approach to develop a RL-based CAC algorithm which counteracts the drawbacks of the other two approaches, i.e., the scalability problem of the LP- and DP-based ones, and the difficulty in enforcing the constraints (i.e., in the tuning the Lagrange multipliers) of the RL algorithm based on the Lagrangian approach. In a nutshell, the proposed approach finds a lexicographically sub-optimal solution of the CAC problem with class constraints by learning on-line the policy via RL-based update rules[1].

The paper is organized as follows: in Section 2 the CAC problem is defined as a stochastic control problem; Section 3 defines the constrained MDP formulation and introduces RL; in Section 4, the proposed solution approach is described; Section 5 shows some simulation results; finally, Section 6 draws the conclusions.

The notation is the following: vectors are denoted in boldface letters, matrices in capital boldface, and $\boldsymbol{\delta}_c$ denotes a *C*-vector of zeros but the *c*-th element equal to 1.

2 Problem statement

We consider a generic link characterized by its available capacity, denoted with η_{link}, which supports C CoSs, each one characterized by a transmission bitrate $b_c, c = 1, \dots, C$. Let $\boldsymbol{s}(t)$ be the state of the system, represented in Figure 1 as a discrete-time control system. The state $\boldsymbol{s}(t)$ represents the vector of on-going calls, defined by the number $s_c(t)$ of calls of each class c on-going at time t:

$$\boldsymbol{s}(t) = \left(s_1(t), s_2(t), \dots, s_C(t)\right), s_c \in \mathbb{N}_{\geq 0}, c = 1, \dots, C. \tag{1}$$

[1] A preliminary version of the algorithm was presented in (Panfili & Pietrabissa, 2013); in the present paper, the algorithm is further detailed and comprehensively evaluated against the optimal DP solutions and the Lagrangian RL approach.

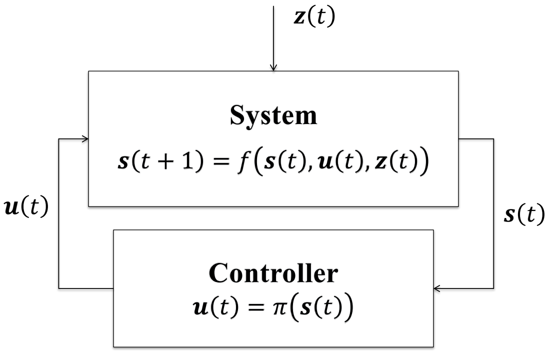
$\boldsymbol{z}(t)$
System
$\boldsymbol{s}(t+1) = f(\boldsymbol{s}(t), \boldsymbol{u}(t), \boldsymbol{z}(t))$
$\boldsymbol{u}(t)$
$\boldsymbol{s}(t)$
Controller
$\boldsymbol{u}(t) = \pi(\boldsymbol{s}(t))$

Figure 1. Discrete-time model of the system and the admission controller.

The link throughput of the system associated to $\boldsymbol{s}(t)$ is then $\eta(\boldsymbol{s}(t)) = \sum_{c=1,\dots C} b_c s_c(t)$.

At time t, for each class $c = 1, \dots, C$, two types of events may occur: a call request or a call termination. Let the disturbance $\boldsymbol{z}(t)$ represent the call attempts and terminations events, and let $\boldsymbol{u}(t)$ be the control action, relevant at a call attempt at time t; $\boldsymbol{u}(t)$ is the admission decision:

$$\boldsymbol{u}(t) = (u_1(t), \dots, u_C(t)), u_c \in \{0,1\}, c = 1, \dots, C, \tag{2}$$

whose element $u_c(t), c = 1, \dots, C$, is equal to 1 if the decision is to accept the event, to 0 if it is to reject it.

In general, the system dynamics is function of the state $\boldsymbol{s}(t)$, of the control action $\boldsymbol{u}(t)$ and of the disturbance $\boldsymbol{z}(t)$:

$$\boldsymbol{s}(t+1) = f(\boldsymbol{s}(t), \boldsymbol{u}(t), \boldsymbol{z}(t)). \tag{3}$$

The controller task is then to decide the control action $\boldsymbol{u}(t)$ based on the current state $\boldsymbol{s}(t)$; the control action is a map from the state space to the action space: $\boldsymbol{u}(t) = \pi[\boldsymbol{s}(t)]$. The control objective is to minimize average blocking probability, or, equivalently, to maximize the acceptance probability, while enforcing constraints on the blocking probabilities of the different Classes of Service.

3 Discrete-time MDP formulation and RL

The algorithm is conceived as a statistical-based call control, where call attempts and terminations are characterized as follows: for each class c, call attempts are distributed according to a Poisson process with mean arrival frequency λ_c; the call holding time of class c is distributed according to an exponential distribution, and the mean termination frequency is μ_c[(2)].

[2] Poisson call attempts and exponential call holding time are widely used in the literature and are adequate at least for voice users, but further research is needed in the area of Markov regenerative decision processes to justify it for the new traffic services (Kalyanasundaram, Chong, & Shroff, 2002; Krishnamurthy & Leung, 2006). However,

We are interested in minimizing the average blocking probability while keeping the blocking probabilities of each CoS below a given threshold. The constrained MDP with multiple constraints is then defined by the tuple $\{S, A, \mathbf{T}, \rho, \boldsymbol{d}, \boldsymbol{K}\}$, where S is the finite state space, A is the action space, $\mathbf{T}$ is the transition probability matrix, ρ is the one-step cost function which accounts for the blocking probability, $\boldsymbol{d}$ is a set of cost functions d_c, which are the one-step cost functions which account for the blocking probability of class c, and $\boldsymbol{K}$ is a vector of C constants K_c, which are the threshold which limits the average costs of class, $c = 1, \dots, C$. The tuple elements will be described in the following Sections 3.1-3.5; in section 3.6 RL is introduced.

3.1 State space

The finite state space S is the set of the feasible on-going call vectors, defined by equation (1), i.e., the vectors $\mathbf{s} = (s_1, \dots, s_C) \in S$ whose throughput $\eta(\mathbf{s}) = \sum_{c=1,\dots C} b_c s_c$ is less than the link capacity:

$$S = \{\mathbf{s} = (s_1, \dots, s_C) | \eta(\mathbf{s}) \leq \eta_{link}\} \tag{4}$$

3.2 Action space

In the generic state $\boldsymbol{s} = (s_1, \dots, s_C) \in S$, the controller might decide to *accept* or *reject* a new call request. The decision is expressed by the vector $\boldsymbol{u}$, defined by equation (2). Note that, in the state $\boldsymbol{s}$, a new call request of class c may be accepted only if the state $\boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{\delta}_c \in S$ (i.e., if the state $\boldsymbol{s}' = (s_1, \dots, s_{c+1}, \dots, s_C)$ exists). The action space of an available state is defined as follows:

$$A(\boldsymbol{s}) = \left\{\boldsymbol{u} = (u_1, \dots, u_C) \middle| u_c \in \begin{cases} \{0,1\}, \text{if } \boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{\delta}_c \in S \\ \{0\}, \text{otherwise} \end{cases}, c = 1, \dots, C\right\}, \boldsymbol{s} \in S. \tag{5}$$

If, every time the system is in state $\boldsymbol{s}$**,** the controller chooses the control action $\boldsymbol{u} \in A(\boldsymbol{s})$ with the same probability, for all the states $\boldsymbol{s} \in S$, then the controller is said to operate under a stochastic stationary policy. A stochastic policy π defines a probability distribution on the action space, and the probability that decision $\boldsymbol{u}$ is taken in state $\boldsymbol{s}$ is denoted as $\pi(\boldsymbol{s}, \boldsymbol{u})$.

Policies can also be deterministic if, in each state $\boldsymbol{s} \in S$, one of the actions is choosen with probability 1; therefore, a deterministic policy associates a unique action $\boldsymbol{u}' \in A(\boldsymbol{s})$ to each state $\boldsymbol{s} \in S$: $\pi(\boldsymbol{s}, \boldsymbol{u}) = \begin{cases} 1, \text{if } \boldsymbol{u} = \boldsymbol{u}' \\ 0, \text{otherwise} \end{cases}$. For the sake of simplicity, the selected action $\boldsymbol{u}'$ in state $\boldsymbol{s}$ of a deterministic policy π will be denoted as $\pi(\boldsymbol{s}) = \boldsymbol{u}'$.

3.3 Transition matrix

The elements of the transition matrix $\mathbf{T}$ are the probabilities of the transitions between

note that traffic statistics are necessary to develop the MDP framework but are not required by the RL approach – see for example (Lilith & Dogancay, 2005), whose RL-based admission control succeeded in controlling traffic characterized by a self-similar distribution.

state couples; the transition probabilities are inferred from the above-stated assumptions on call birth frequencies and holding times, and from the above-defined action space.

Moreover, to define a discrete-time MDP, an *uniformization* procedure is considered (Pietrabissa, 2009b): all the transition frequencies are divided by a constant ν, and a self-transition is added to each state to let the total state outgoing probability equal to 1. It can be demonstrated that, if ν is larger than the maximum output frequency among the ones of all the states, the obtained discrete-time MDP is statistically equivalent to the continuous-time MDP generated by the transition frequencies. Considering a generic state $\boldsymbol{s} = (s_1, \dots, s_C) \in S$ and the policy π, the transition probabilities are then:

$$p^{\pi}(\boldsymbol{s}, \boldsymbol{s}') = \begin{cases} \frac{\lambda_c}{\nu}\pi(\boldsymbol{s}, u_c), \text{ if } \boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{\delta}_c \\ \frac{\mu_c}{\nu}s_c, \text{ if } \boldsymbol{s}' = \boldsymbol{s} - \boldsymbol{\delta}_c \\ 1 - \sum_{s' \neq s} p^{\pi}(\boldsymbol{s}, \boldsymbol{s}'), \text{ if } \boldsymbol{s}' = \boldsymbol{s} \\ 0, \text{ otherwise} \end{cases}, \boldsymbol{s}, \boldsymbol{s}' \in S \qquad (6)$$

3.4 Cost function

The main control objective is to drive the evolution of the discrete-time Markov process $\{\boldsymbol{s}_k\}_{k\,=\,1,2,\dots}$, where $\boldsymbol{s}_k$ is the state visited at time step k, to minimize the blocking probability. Let the *c*-th (action,state)-dependent cost function be defined as:

$$\rho_c(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{s}') = \begin{cases} 1 - u_c, \text{ if } \boldsymbol{s}' = \boldsymbol{s} + \boldsymbol{\delta}_c \\ 0, \text{ otherwise} \end{cases}, \boldsymbol{u} = (u_1, \dots, u_C) \in A(\boldsymbol{s}), \boldsymbol{s}, \boldsymbol{s}' \in S, c = 1, \dots, C; \qquad (7)$$

The cost function is then:

$$\rho(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{s}') = \sum_{c=1,\dots,C} \rho_c(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{s}')\,, \boldsymbol{u} \in A(\boldsymbol{s}), \boldsymbol{s}, \boldsymbol{s}' \in S \qquad (8)$$

Therefore, to minimize the blocking probability, we should minimize the expected per-stage cost of the system, computed as $\lim_{T\to\infty}\frac{1}{T}\sum_{k=1,\dots,T}\rho(\boldsymbol{s}_k, \boldsymbol{u}_k, \boldsymbol{s}_{k+1})$, where ρ_k is the cost observed at step k. However, to make use of standard RL algorithms (see Section 4), we approximate the above limit, by using the Tauberian approximation, with the expected discounted total cost (Gábor et al., 1998), defined as:

$$J_{\rho}^{\pi,\mathcal{X}} = E_{\pi,\mathcal{X}}\left\{\sum_{k=0,1,\dots,\infty}\gamma^k\rho(\boldsymbol{s}_k, \boldsymbol{u}_k, \boldsymbol{s}_{k+1})\right\} = \sum_{\boldsymbol{s}_0\in S}\mathcal{X}(\boldsymbol{s}_0)E_{\pi}\left[\rho(\boldsymbol{s}_0, \boldsymbol{u}_0, \boldsymbol{s}_1) + \sum_{k=1,2,\dots,\infty}\gamma^k\rho(\boldsymbol{s}_k, \boldsymbol{u}_k, \boldsymbol{s}_{k+1})\right], \qquad (9)$$

where $\gamma \in (0,1)$ is the discount factor, which weights immediate costs versus delayed costs, $\mathcal{X} \in \mathrm{X}$ is the probability distribution of the initial state $\boldsymbol{s}_0$ over the state set S, X is the set of feasible initial probability distributions, the operators $E_{\pi}\{\cdot\}$ and $E_{\pi,\mathcal{X}}\{\cdot\}$ are the expected value when the system operates under policy π and the expected value when the system operates under policy π and the initial state distribution is $\mathcal{X}$, respectively.

3.5 Value functions, cost constraints and lexicographic approach

In unconstrained MDPs, state-value functions are used to evaluate the policies. The state-value function $V_\rho^\pi(\boldsymbol{s})$ is the expected cost when starting in $\boldsymbol{s}$ and following policy π:

$$V_\rho^\pi(\boldsymbol{s}) = E_\pi\left[\sum_{k=0,1,\dots,\infty} \gamma^k \rho(\boldsymbol{s}_k, \boldsymbol{u}_k, \boldsymbol{s}_{k+1}) \mid \boldsymbol{s}_0 = \boldsymbol{s}\right], \boldsymbol{s} \in S. \qquad (10)$$

By comparing equations (9) and (10), it follows that the expected discounted total cost is the expected value of the value functions V_ρ^π, over the initial distribution χ of the starting state:

$$J_\rho^{\pi,\mathcal{X}} = E_\chi\left[V_\rho{}^\pi(\boldsymbol{s})\right] = \sum_{s \in S} \chi(\boldsymbol{s}) V_\rho{}^\pi(\boldsymbol{s}) \qquad (11)$$

In constrained MDPs, additional cost functions are defined to enforce the constraints (to avoid confusion, hereafter the cost (8) will be referred to as primary cost). In our problem, we have C constraints on the C CoS blocking probabilities. Considering the per-class costs (7), the c-th (action,state)-dependent cost function is then defined as:

$$d_c(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{s}') = \rho_c(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{s}'), \boldsymbol{u} \in A(\boldsymbol{s}), \boldsymbol{s}, \boldsymbol{s}' \in S, c = 1, \dots, C, \qquad (12)$$

Likewise, the value functions are:

$$V_c{}^\pi(\boldsymbol{s}_0) = E_\pi\left[\sum_{k=0,1,\dots,\infty} \gamma^k d_c(\boldsymbol{s}_k, \boldsymbol{u}_k, \boldsymbol{s}_{k+1})\right], c = 1, \dots, C, \qquad (13)$$

We are interested in controlling the CoS blocking probabilities P_c, in such a way that their values remains below the given maximum blocking probabilities $P_c^{max}, c = 1, \dots, C$. The blocking probabilities can be computed by using equation (12) as the expected per-stage costs of the system: $P_c = \lim_{T \to \infty} \frac{1}{T} \sum_{k=1,\dots,T} d_c(\boldsymbol{s}_k, \boldsymbol{u}_k, \boldsymbol{s}_{k+1})$. Similarly to the primary cost, we approximate the blocking probability computation by using the expected discounted total costs, obtaining the following constraints:

$$J_c^{\pi,\mathcal{X}} = E_\pi\left\{\sum_{k=0,1,\dots,\infty} \gamma^k d_c(\boldsymbol{s}_k, \boldsymbol{u}_k, \boldsymbol{s}_{k+1})\right\} = E_\chi[V_c{}^\pi(\boldsymbol{s})]$$
$$= \sum_{s \in S} \chi(\boldsymbol{s}) V_c{}^\pi(\boldsymbol{s}) \le K_c, c = 1, \dots, C, \qquad (14)$$

where $K_c = \frac{P_c^{max}}{1-\gamma}$ [3].

The constrained MDP is then formulated as the following optimization problem:

$$\min_\pi J_\rho^{\pi,\mathcal{X}}$$
$$s.t.\ J_c^{\pi,\mathcal{X}} \le K_c, c = 1, \dots, C, \qquad (15)$$

which is written in terms of value-functions as follows (see equations (11) and (14)):

[3] In fact, by choosing a discount factor close enough to 1, it holds that $E^\pi\left\{\sum_{k=0,1,\dots,\infty} \gamma^k d_c(\boldsymbol{s}_k, \boldsymbol{u}_k)\right\} \approx \sum_{k=0,1,\dots,\infty} \gamma^k P_c = (1-\gamma)P_c$ (Gábor et al., 1998).

$$\begin{gathered}\min_{\pi} \sum_{\boldsymbol{s}\in S} \chi(\boldsymbol{s}) V_{\rho}{}^{\pi}(\boldsymbol{s}) \\ s.t.\ \ \sum_{\boldsymbol{s}\in S} \chi(\boldsymbol{s}) V_{c}{}^{\pi}(\boldsymbol{s}) \leq K_c, c = 1, \dots, C,\end{gathered} \tag{16}$$

In (Gábor et al., 1998), a single constraint in the form $V_c^{\pi}(\boldsymbol{s}) \leq K_c$ is enforced by defining a lower-bounded value function:

$$\boldsymbol{V}^{\pi}(\boldsymbol{s}) = \begin{pmatrix} \max\left(K_c, V_c^{\pi}(\boldsymbol{s})\right) \\ V_{\rho}^{\pi}(\boldsymbol{s}) \end{pmatrix}. \tag{17}$$

The lexicographic approach has the following goals: if $V_c^{\pi}(\boldsymbol{s}) > K_c$ (i.e, the constraint is not met), the objective is to minimize $V_c^{\pi}(\boldsymbol{s})$; otherwise, the objective is to minimize the primary cost value function $V_{\rho}^{\pi}(\boldsymbol{s})$. According the lexicographic approach, two policies π and π' can be compared. The policy π' is better than π, i.e., $\pi > \pi'$, if either $V_c^{\pi}(\boldsymbol{s}) > K_c$ and $V_c^{\pi'}(\boldsymbol{s}) < V_c^{\pi}(\boldsymbol{s})$ or $V_c^{\pi}(\boldsymbol{s}) < K_c$, $V_c^{\pi'}(\boldsymbol{s}) < K_c$ and $V_{\rho}^{\pi'}(\boldsymbol{s}) < V_{\rho}^{\pi}(\boldsymbol{s})$.

Note that the lexicographic approach is conservative, since it checks the constraint on each state, actually solving the following problem:

$$\begin{gathered}\min_{\pi} \sum_{\boldsymbol{s}\in S} \chi(\boldsymbol{s}) V_{\rho}{}^{\pi}(\boldsymbol{s}) \\ s.t.\ V_{c}{}^{\pi}(\boldsymbol{s}) \leq K_c, c = 1, \dots, C, \boldsymbol{s} \in S.\end{gathered} \tag{18}$$

In fact, if $V_c^{\pi}(\boldsymbol{s}) \leq K_c, \boldsymbol{s} \in S$, then it holds that $J_c^{\pi,\mathcal{X}} = \sum_{\boldsymbol{s}\in S} \chi(\boldsymbol{s}) V_c{}^{\pi}(\boldsymbol{s}) \leq K_c, \mathcal{X} \in \mathrm{X}$, but the opposite implication is not true: hence solving (18) leads to a conservative sub-optimal solution of problem (15). Therefore, by using the lexicographic approach within DP algorithms, a stationary deterministic policy is found, which is lexicographically optimal with respect to $\left(V_{\rho}^{\pi}(\boldsymbol{s}), V_c^{\pi}(\boldsymbol{s})\right)$ in every state $\boldsymbol{s}$; the stationary policy is also a suboptimal feasible solution of the problem (15) (Geibel, 2007).

In our multi-constraint problem, in Section 4 we will define a vector of $(C + 1)$ (action,value)-functions.

3.6 Reinforcement learning

Unconstrained MDPs can be solved on-line by RL algorithms. In particular, the common Q-learning approach (Sutton & Barto, 1998) considered in this paper computes the control policy on-line by estimating the (action,state)-value functions. The (state,action)-value function $Q^{\pi}(\boldsymbol{s})$ is the expected cost starting from $\boldsymbol{s}$, taking action $\boldsymbol{u}$, and thereafter following policy π:

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{u}) = E_{\pi}\left[\sum_{k=0,1,\dots,\infty} \gamma^k \rho(\boldsymbol{s}_k, \boldsymbol{u}_k, \boldsymbol{s}_{k+1}) \,|\boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{u}_0 = \boldsymbol{u}\right], \boldsymbol{s} \in S, \boldsymbol{u} \in A(\boldsymbol{s}). \tag{19}$$

The Q-learning iteratively estimates the (action,state)-value functions on-line, exploiting the Bellman equations:

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{u}) = \sum_{\boldsymbol{s}'\in S} p^{\boldsymbol{\pi}}(\boldsymbol{s}, \boldsymbol{s}')\left[\rho(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{s}') + \gamma Q^{\pi}\left(\boldsymbol{s}', \pi(\boldsymbol{s}')\right)\right], \boldsymbol{s} \in S, \boldsymbol{u} \in A(\boldsymbol{s}). \tag{20}$$

The algorithm is outlined hereafter.

STEP 0. Extract an initial state $\boldsymbol{s} \in S$ from the distribution $\mathcal{X}$, an initial policy π, and initial estimates for the (action,state)-value functions $Q(\boldsymbol{s}, \boldsymbol{u}), \boldsymbol{u} \in A(\boldsymbol{s}), \boldsymbol{s} \in S$.

STEP 1. Choose an action $\boldsymbol{u} \in A(\boldsymbol{s})$ according to π, following the so-called the ε-greedy policy approach:

$$\boldsymbol{u} = \begin{cases} \pi(\boldsymbol{s}), \text{with probability } (1-\varepsilon) \\ \text{a random action in the set } A(\boldsymbol{s}), \text{with probability } \varepsilon \text{'} \end{cases} \qquad (21)$$

with $\varepsilon \in (0,1)$. The parameter ε addresses the trade-off between the need of exploring the state-space, i.e., to discover new policies, and to exploit the current estimates of the (action,state)-values.

STEP 2. Observe the new state $\boldsymbol{s}'$ and the cost $\rho(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{s}')$.

STEP 3. Update the estimates the (action,state)-value functions by the following update rule:

$$Q_\rho(\mathbf{s}, \mathbf{u}) \leftarrow (1-\alpha) Q_\rho(\mathbf{s}, \mathbf{u}) + \alpha \left[r(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{s}') + \gamma \min_{\boldsymbol{u}' \in A(\boldsymbol{s}')} Q_\rho(\boldsymbol{s}', \boldsymbol{u}') \right], \qquad (22)$$

where the learning rate $\alpha \in (0,1)$ determine the convergence speed and accuracy. Note that α can also be dependent on the state and on the stage.

STEP 4. Update the policy with the so-called policy-improvement step:

$$\pi(\boldsymbol{s}) \leftarrow argmin_{\boldsymbol{u} \in A(\boldsymbol{s})} Q_\rho(\boldsymbol{s}, \boldsymbol{u}). \qquad (23)$$

STEP 5. Set $(\mathbf{s}, \mathbf{u}) \leftarrow (\mathbf{s}', \mathbf{u}')$ and return to STEP 1.

If the learning rate α is chosen appropriately, the Q-learning algorithm is shown to converge to a stationary deterministic optimal policy in the long run (for a detailed analysis, see (Sutton & Barto, 1998)). Assumptions on the learning rate are loose – it suffices that $\sum_{k=0,1,\dots,\infty} \alpha_k = \infty$ and $\sum_{k=0,1,\dots,\infty} \alpha_k^2 < \infty$, where k is the stage index – but, obviously, the infinite visits assumption is impractical. In many practical cases, however, it was observed that it rapidly achieves effective (even if suboptimal) policies in a reasonable amount of time.

4 Lexicographic Q-Learning

We consider the approach developed in (Gábor et al., 1998) and (Geibel, 2007) to find a lexicographically sub-optimal solution for the above described constrained MDP by RL methods. Recalling that $V^\pi(\boldsymbol{s}) = Q^\pi(\boldsymbol{s}, \pi(\boldsymbol{s}))$ (as it results by comparing equations (10) and (19)), problem (15) can be written in terms of (action,value)-functions as follows:

$$\begin{gathered} \min_\pi E_\chi\left[{Q_\rho}^\pi(\boldsymbol{s}, \pi(\boldsymbol{s})) \right] \\ s.t.\ E_\chi\left[{Q_c}^\pi(\boldsymbol{s}, \pi(\boldsymbol{s})) \right] \leq K_c, c = 1, \dots, C. \end{gathered} \qquad (24)$$

As in (Gábor et al., 1998), the constraints are enforced in a lexicographic fashion (see equation (17)); in our case, multiple constraints are represented as follows:

$$\boldsymbol{Q}^{\pi}(\boldsymbol{s},\boldsymbol{u}) = \begin{pmatrix} \max\left(K_1, Q_1^{\pi}(\boldsymbol{s},\boldsymbol{u})\right) \\ \vdots \\ \max\left(K_C, Q_C^{\pi}(\boldsymbol{s},\boldsymbol{u})\right) \\ Q_{\rho}^{\pi}(\boldsymbol{s},\boldsymbol{u}) \end{pmatrix} \qquad (25)$$

where the first C elements of (25) are lower-bounded (action,value)-functions. As analyzed in Section 3.5, the lexicographic approach is conservative since it solves the following problem (the counterpart of problem 14b):

$$\begin{gathered} \min_{\pi} E_{\chi}\left[Q_{\rho}{}^{\pi}\left(\boldsymbol{s},\pi(\boldsymbol{s})\right)\right] \\ s.t.\ Q_c{}^{\pi}\left(\boldsymbol{s},\pi(\boldsymbol{s})\right) \le K_c, c = 1,\dots,C, \boldsymbol{s} \in S. \end{gathered} \qquad (26)$$

The lexicographic approach relies on the ordering of the value functions in equation (25), which defines the priority of the objectives. Without loss of generality, we assume that the constraints are ordered in descending order of priority; then the lexicographic approach has the following goals:

(1) if $Q_1^{\pi}(\boldsymbol{s},\boldsymbol{u}) > K_1$ (i.e, the first constraint is not met), the objective is to minimize the first cost (action,value)-function $Q_1^{\pi}(\boldsymbol{s})$;

(2) if $Q_c^{\pi}(\boldsymbol{s},\boldsymbol{u}) \le K_c, c = 1,2,\dots,g-1$, with $g = 2,\dots,C$, and $Q_g^{\pi}(\boldsymbol{s},\boldsymbol{u}) > K_g$ (i.e, the first $g-1$ constraints are met and the g-th constraint is not met), the objective is to minimize the g-th (action,value)-function $Q_g^{\pi}(\boldsymbol{s},\boldsymbol{u})$;

(3) if $Q_c^{\pi}(\boldsymbol{s},\boldsymbol{u}) \le K_c, c = 1,2,\dots,C$, (i.e., all the constraints are met) the objective is to minimize the cost (action,value)-function $Q_{\rho}^{\pi}(\boldsymbol{s},\boldsymbol{u})$.

In brief, the objective is to minimize the primary cost only when the blocking probabilities of all CoSs do not exceed the maximum thresholds; if the constraint if not met by one or more CoSs, the objective is to minimize the value function of the CoS with the highest priority.

According the lexicographic approach, two policies π and π' can be compared. The policy π' is better than π, i.e., $\pi > \pi'$, if one of the following conditions hold:

(1) $Q_1^{\pi}(\boldsymbol{s},\boldsymbol{u}) > K_1$ and $Q_1^{\pi'}(\boldsymbol{s},\boldsymbol{u}) < Q_1^{\pi}(\boldsymbol{s},\boldsymbol{u})$;

(2) $\min\left(Q_c^{\pi}(\boldsymbol{s},\boldsymbol{u}), K_c\right) = \min\left(Q_c^{\pi'}(\boldsymbol{s},\boldsymbol{u}), K_c\right) = K_c, c = 1,\dots,g-1$, and $Q_g^{\pi'}(\boldsymbol{s},\boldsymbol{u}) < Q_g^{\pi}(\boldsymbol{s},\boldsymbol{u}), g = 2,3,\dots,C$;

(3) $\min\left(Q_c^{\pi}(\boldsymbol{s},\boldsymbol{u}), K_c\right) = \min\left(Q_c^{\pi'}(\boldsymbol{s},\boldsymbol{u}), K_c\right) = K_c, c = 1,2,\dots,C$, and $Q_{\rho}^{\pi'}(\boldsymbol{s},\boldsymbol{u}) < Q_{\rho}^{\pi}(\boldsymbol{s},\boldsymbol{u})$.

In words, the policy π' improves the policy π if: i) policy π does not meet constraint 1 and the first cost function is decreased by π'; ii) the first $c-1$ constraints are met by both policies and the c-th cost function is decreased by π'; iii) all the constraints are met by both policies and the expected cost is decreased by π'. Note that only one of the above-stated conditions holds at a time.

The idea is the integration of a RL algorithm – namely, in this case, the Q-learning one – into the lexicographic approach: in other terms, the proposed approach finds a lexicographically optimal solution of the constrained MDP by on-line estimating the (action,value)-functions generated by the primary cost function and by the cost functions.

The standard Q-learning update rule (22) is applied:

$$\begin{cases} Q_c(\boldsymbol{s},\boldsymbol{u}) \leftarrow (1-\alpha)Q_c(\boldsymbol{s},\boldsymbol{u}) + \alpha\left(d_c(\boldsymbol{s},\boldsymbol{u}) + \gamma \underset{\boldsymbol{u}'\in A(\boldsymbol{s}')}{min} Q_c(\boldsymbol{s}',\boldsymbol{u}')\right), c = 1,\dots,C \\ Q_\rho(\boldsymbol{s},\boldsymbol{u}) \leftarrow (1-\alpha)Q_\rho(\boldsymbol{s},\boldsymbol{u}) + \alpha\left(d_r(\boldsymbol{s},\boldsymbol{u}) + \gamma \underset{\boldsymbol{u}'\in A(\boldsymbol{s}')}{min} Q_\rho(\boldsymbol{s}',\boldsymbol{u}')\right) \end{cases}. \tag{27}$$

Note that, in (27), every (action,value)-function is updated at every step.

To describe the policy improvement step, we define the restricted action sets $\tilde{A}_c(\boldsymbol{s}) \subseteq A(\boldsymbol{s}), \boldsymbol{s} \in S$, as follows:

$$\tilde{A}_c(\boldsymbol{s}) = \{\boldsymbol{u} \in A(\boldsymbol{s}) | Q_g(\boldsymbol{s},\boldsymbol{u}) \leq K_g, g = 1,\dots,c\}, \boldsymbol{s} \in S. \tag{28}$$

$\tilde{A}_c(\boldsymbol{s})$ is then the set of the actions which meet the constraints $1,\dots,c$. Considering the restricted action sets $\tilde{A}_c$, the standard Q-learning update rule (23) and the definition (25) of the vector $\boldsymbol{Q}^\pi(\boldsymbol{s},\boldsymbol{u})$, the proposed update rule is defined as follow:

$$\pi(\boldsymbol{s}) \leftarrow \begin{cases} \underset{\boldsymbol{u}\in\tilde{A}(\boldsymbol{s})}{\operatorname{argmin}}\{Q_1(\boldsymbol{s},\boldsymbol{u})\}, \text{if } \tilde{A}_1(\boldsymbol{s}) = \emptyset; \\ \underset{\boldsymbol{u}\in\tilde{A}(\boldsymbol{s})}{\operatorname{argmin}}\{Q_c(\boldsymbol{s},\boldsymbol{u})\}, \text{if } \tilde{A}_{c-1}(\boldsymbol{s}) \neq \emptyset \text{ and } \tilde{A}_c(\boldsymbol{s}) = \emptyset, c = 2,\dots,C-1; \\ \underset{\boldsymbol{u}\in\tilde{A}(\boldsymbol{s})}{\operatorname{argmin}}\{Q_\rho(\boldsymbol{s},\boldsymbol{u})\}, \text{if } \tilde{A}_C(\boldsymbol{s}) \neq \emptyset. \end{cases} \tag{29}$$

In words, the policy improvement works as follows:

(1) if no action exists such that the first constraint is met (in the sense that $Q_1(\boldsymbol{s},\boldsymbol{u}) > K_1, \forall\boldsymbol{u} \in A(\boldsymbol{s})$), pick the action $\boldsymbol{u} \in A(\boldsymbol{s})$ which minimizes the first cost $Q_1(\boldsymbol{s},\boldsymbol{u})$;

(2) if no action exists such that the first c constraints are met but there exists a non-empty set of actions $\tilde{A}_{c-1}(\boldsymbol{s})$ such that the first $c-1$ constraints are met, pick the action $\boldsymbol{u} \in \tilde{A}_{c-1}(\boldsymbol{s})$ which minimizes the c-th cost $Q_c(\boldsymbol{s},\boldsymbol{u})$;

(3) if there exists a non-empty set of actions $\tilde{A}_C(\boldsymbol{s})$ such that all the constraints are met, pick the action $\boldsymbol{u} \in \tilde{A}_C(\boldsymbol{s})$ which minimizes the cost $Q_\rho(\boldsymbol{s},\boldsymbol{u})$.

The proposed lexicographic Q-learning algorithm is summarized below:

STEP 0. Extract an initial state $\boldsymbol{s} \in S$ from the distribution $\mathcal{X}$, an initial policy π (e.g., the greedy policy which accepts all the calls whenever possible) and an initial estimates for the (action,state)-value functions $Q_c(\boldsymbol{s},\boldsymbol{u}), \boldsymbol{c} = 1,\dots,C$, and $Q_\rho(\boldsymbol{s},\boldsymbol{u}), \boldsymbol{u} \in A(\boldsymbol{s}), \boldsymbol{s} \in S$ (e.g., $Q_c(\boldsymbol{s},\boldsymbol{u}) = 0$ and $Q_\rho(\boldsymbol{s},\boldsymbol{u}) = 0, \boldsymbol{u} \in A(\boldsymbol{s}), \boldsymbol{s} \in S$).

STEP 1. Choose $\boldsymbol{u} \in A(\boldsymbol{s})$ according to π, with the ε-greedy policy (21).

STEP 2. Observe the new state $\boldsymbol{s}'$, the cost $\rho(\boldsymbol{s},\boldsymbol{u},\boldsymbol{s}')$ and the costs $d_c(\boldsymbol{s},\boldsymbol{u},\boldsymbol{s}'), c = 1,\dots,C$.

STEP 3. Update the estimates of the (action,state)-value functions with the update rule (27).

STEP 4. Update the policy with the policy-improvement step (29).

STEP 5. Set $(\mathbf{s},\mathbf{u}) \leftarrow (\mathbf{s}',\mathbf{u}')$ and return to STEP1.

For each (action,value)-function of the algorithm, the convergence properties are the same as the ones of the Q-learning. Therefore, by appropriately choosing the learning rate α, the algorithm converges, in the long run (the same observations for the standard Q-learning algorithm hold, see Section 3.6), to a stationary deterministic policy, which is a suboptimal solution of the constrained problem (15). It is interesting to note that, in case no feasible policies exist, i.e., no policies exist which satisfy all the CoS blocking probability constraints, no solution is returned by the LP method, whereas the lexicographic RL approach converges to a policy which satisfies the maximum number of ordered constraints.

The effectiveness of the proposed lexicographic RL approach is evaluate din the following Section by simulations.

5 Simulations

Numerical simulations have been performed with the aim of evaluating the effectiveness of the proposed approach. The simulated link supports $C = 3$ different CoSs, characterized by a set of parameters: transmission bitrates b_c, arrival rates λ_c and termination rates μ_c, c = 1, 2, 3. The classes are numbered in descending order of priority.

The first simulations aim at evaluating the policy obtained by proposed lexicographic RL algorithm. Five scenarios were setup, characterized by different values of offered traffic load, computed as:

$$\eta_{off} = \sum_{c=1,\dots,C} \frac{\lambda_c}{\mu_c} b_c \qquad (30)$$

For each scenario, 10 runs were executed, each one $24 \cdot 7$ hours long. In each run, class parameters were used to generate an event list (call births/terminations); at each call birth event, the admission controller uses an admission policy to decide whether to accept or not the call. Each simulation run was executed four times: the first time, a heuristic policy was implemented, referred to as *Greedy*, which, in a greedy fashion, always accepts the calls whenever enough capacity is available; the second time, the optimal policy computed off-line by means of the LP approach (Pietrabissa, 2008b), referred to as *LP*, was implemented; the third time, the sub-optimal policy computed on-line by means of the proposed approach, referred to as *RL-lex*, was implemented; the fourth time, the sub-optimal policy computed on-line by means of the RL approach using the optimal Lagrange multipliers (computed off-line by solving the Lagrangian dual problem of each LP), referred to as *RL-lag*, was implemented.

Since the *LP* and the *RL-lag* require to solve a LP problem, they are subject to the mentioned scalability problem; in practice, we had to execute the simulations in 'small' scenarios, characterized by a link capacity $\eta_{link} = 3$ Mbps, leading to a state space dimension $|S| = 5.3 \cdot 10^3$. The following values of the offered traffic load were considered: $\eta_{off} = \{1.98, 2.01, 2.04, 2.07, 2.1\}$ Mbps. For the sake of convenience, the scenarios will be denoted as *Very Low* , *Low*, *Medium*, *High* and *Very High* load scenarios.

Table 1 collects the simulations parameters, whereas

Table 2 collects the RL parameters used in the RL-based algorithms. For each run of the RL-based algorithms, a training interval has been considered, during which the exploration rate ε was kept much higher than in the rest of the simulation. The initial policies of all RL algorithms were set equal to the greedy policy.

Table 1. Simulations parameters (σ is a parameter such that, for each scenario, the desired value of the offered load η_{off} is obtained by means of equation (30)).

Parameter	Value
η_{link} [Mbps]	3
Simulation length [h]	7·24
η_{off} [Mbps]	{1.98, 2.01, 2.04, 2.07, 2.1}
C	3
b_c [kbps], c =1,2,3	{330, 156, 64}
λ_c [min^{-1}] , c =1,2,3	{0.5σ, 1.25σ, 1.125σ}
μ_c [min^{-1}] , c =1,2,3	{0.2, 0.2, 0.333}

Table 2. RL algorithms parameters.

Parameter	Value
Training interval [hours]	6
$K_c, c = 1,2,3$	{0.075, 0.075, 0.075}
ε	0.1 during the training interval 10^{-4} otherwise
γ	0.9
α	$\frac{0.005}{\text{algorithm iteration}}$

Figure 2 shows the average results of the simulations sets in terms of average blocking probability for each service class. The figure shows that, as the offered traffic increases, i) the *Greedy* policy cannot control the blocking probabilities, and the Class 3 blocking probability exceeds the maximum threshold, ii) the optimal LP policy strictly enforces the blocking probability constraints, iii) the sub-optimal *RL-lex* and *RL-lag* policies effectively control the blocking probabilities – the average Class 3 blocking probability is negligibly over the threshold in the *Medium*, *High* and *Very High* scenarios.

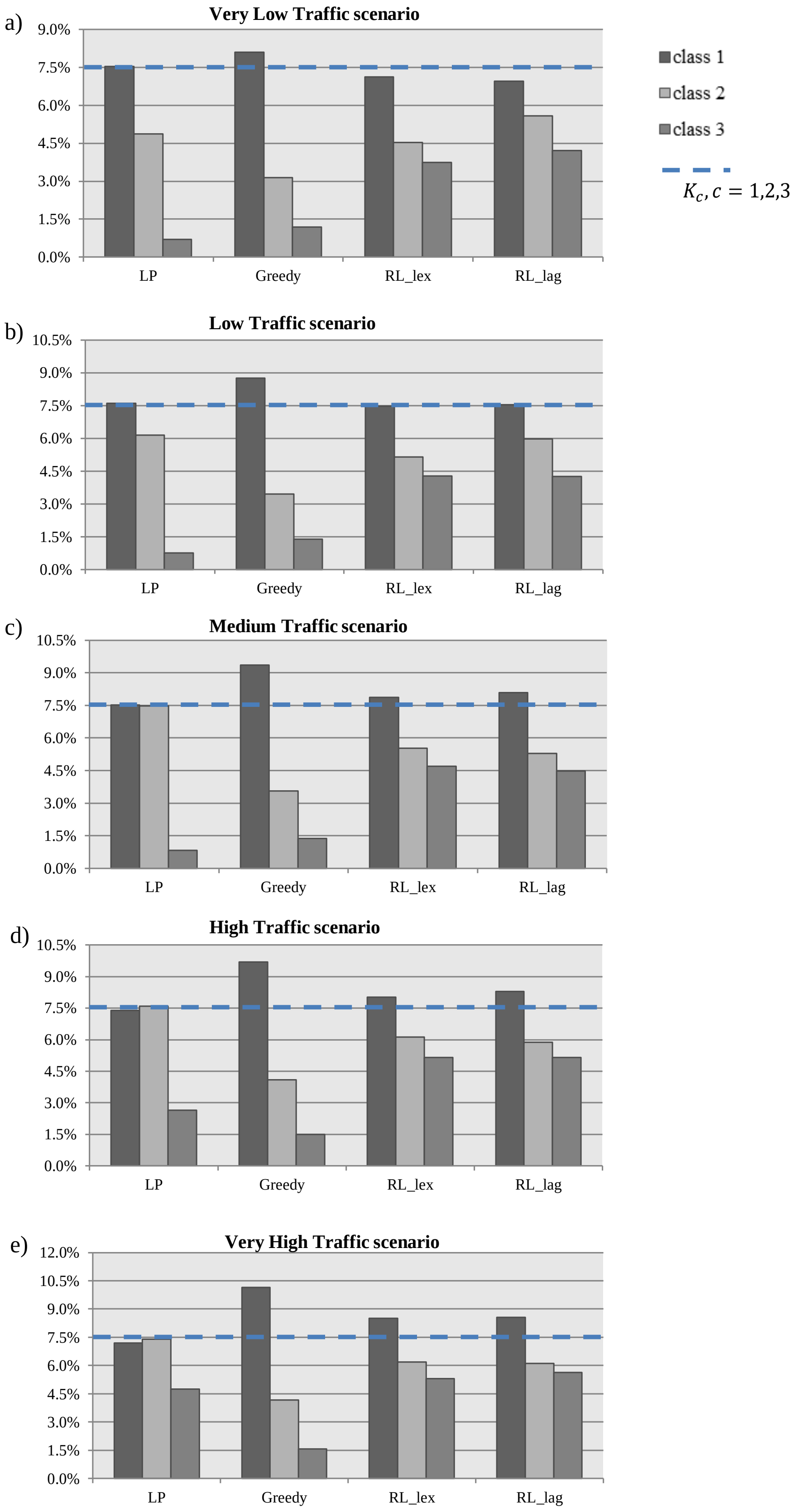

Figure 2. Blocking probabilities the Very Low (a), Low (b), Medium (c), High (d) and Very High (e) traffic load scenarios.

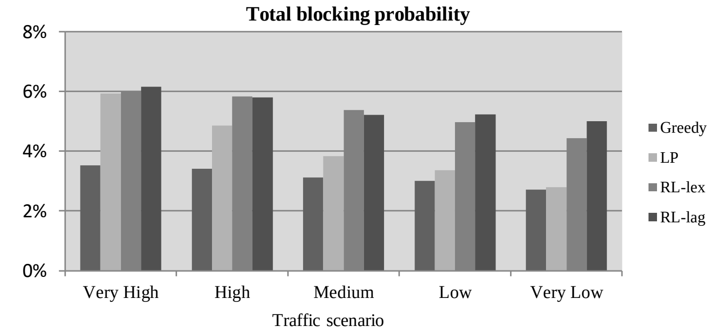

Figure 3. Average percentage of blocked calls per traffic scenario.

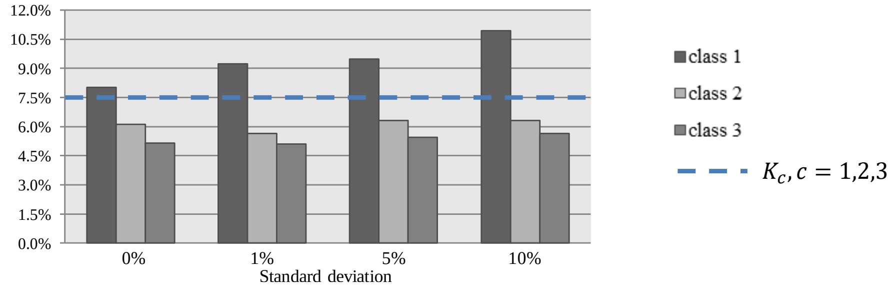

Figure 4. Average results of the *RL-lag* algorithm with different Lagrange multiplier

The total blocking probabilities of the simulation sets are shown in Figure 3. The results show that the *Greedy* policy has the lowest average total blocking probability (as it should be, since no blocking constraints are enforced by the greedy heuristic), followed by the *LP* algorithm and by the *RL-lex* and *RL-lag* ones, which achieve similar blocking probabilities.

Simulations show that the performances of the proposed *RL-lex* approach are similar to the ones of the *RL-lag* approach. However, the *RL-lag* policies were obtained by computing off-line the optimal values of the Lagrange multipliers. To evaluate how the *RL-lag* performance changes with Lagrange multiplier variations, we considered the medium traffic scenario. 20 simulation runs, each one $24 \cdot 7$ hours long, were executed 4 times: the first time, the optimal Lagrange multipliers were used; in the second, third and fourth times, the Lagrange multipliers were selected randomly, using a normal distribution, with mean equal to optimal Lagrange values and standard deviation equal to 1%, 5% and 10%, respectively.

Figure 4 shows the average results of the simulations runs in terms of average blocking probabilities: it is evident that the *RL-lag* approach is effective only if the Lagrange multipliers are accurately computed. Also the primary cost worsen if the multipliers are not exact; the obtained average total blocking probabilities are 5.80%, 5.82%, 6.25% and 6.56% for standard deviations of 0%, 1%, 2%, 5% and 10%, respectively.

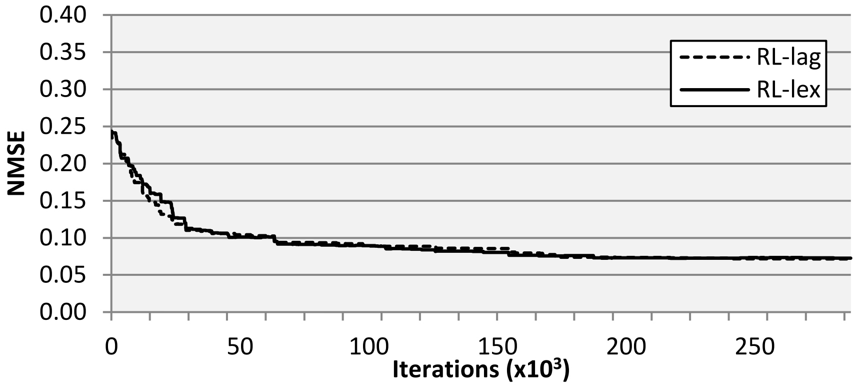

Figure 5. NMSEs *RL-lex* and *RL-lag* value functions with respect to the optimal value function.

Figure 6 shows, for one of the high-load simulation runs, the Normalized Mean Squared Error (NMSE) of the RL approaches, normalized with respect to the optimal value function (computed by solving the DP problem), calculated as $NMSE^{RL-lex} := \frac{1}{|S|}\sum_{\mathbf{s}\in S}\left(\frac{V_{\rho}^{RL-lex}(\mathbf{s})-V^{*}(\mathbf{s})}{V^{*}(\mathbf{s})}\right)^{2}$ and $NMSE^{RL-lag} := \frac{1}{|S|}\sum_{\mathbf{s}\in S}\left(\frac{V^{RL-lag}(\mathbf{s})-V^{*}(\mathbf{s})}{V^{*}(\mathbf{s})}\right)^{2}$, respectively, where $V^{*}(\mathbf{s})$ is the optimal value function in state $\mathbf{s}$, $V_{\rho}^{RL-lex}(\mathbf{s})$ is the last cost value function computed by the lexicographic algorithm in state $\mathbf{s}$, and $V^{RL-lag}(\mathbf{s})$ is the last value function computed by the Lagrangian algorithm in state $\mathbf{s}$. The figure shows that the proposed lexicographic RL approach has similar convergence dynamics with respect to the Lagrangian RL approach. The convergence velocity of the RL approaches is one of the key points in their effectiveness, and is tightly linked to the scalability (larger state and action spaces need more time for the exploration phase); a brief discussion of possible methods to improve scalability is provided in Section 6.

The second simulation is aimed at testing the proposed approach in a larger scenario, where the *LP* and *RL-lag* methods cannot be used due to the scalability issue (of the LP problem itself and of the Lagrangian dual problem, respectively). The *RL-lex* policy is then compared to the *Greedy* heuristic. The scenario parameters are the same as the ones shown in Table 1, but the link capacity $\eta_{link} = 10$ Mbps, which leads to a state space dimension $|S| = 1.65 \cdot 10^{5}$, and the offered load $\eta_{off} = 9$ Mbps. Figure 5 shows the results in terms of blocking probability for each CoS, averaged over 10 simulation runs, each one $24 \cdot 7$ hours long. The figure shows that the proposed *RL-lex* approach is effective in controlling the blocking probabilities, whereas, with the *Greedy* heuristic, CoS 1 blocking probability grows almost to 10%, well above the threshold 7.5%. The price of this control effort is paid by the average total blocking probability, which increases from 3.6% of the *Greedy* policy to 5.3%.

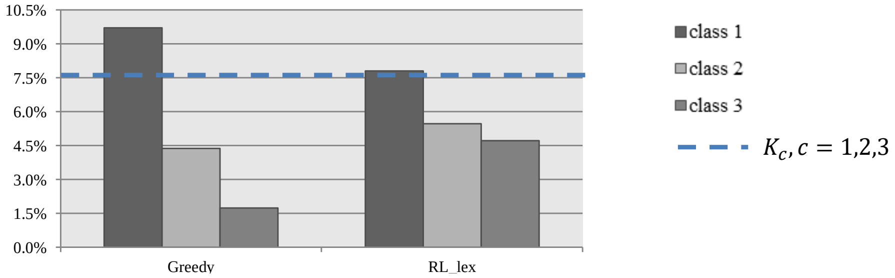

Figure 6. Average blocking probabilities in the second simulation

6 Conclusions

This paper proposes a novel proposed approach to the CAC problem modeled as a multi-constrained MDP. The proposed algorithm finds a lexicographically sub-optimal solution of the constrained MDP by on-line estimating the (action,value)-functions generated by the primary cost function, aimed at minimizing the total admission probability, and by the cost functions, aimed at controlling the blocking probabilities of each class of service.

The proposed approach lightens the scalability problems of the standard LP approach used to solve constrained MDP, and has the advantage over the Lagrangian approach that it does not require a preliminary estimate of the values of the Lagrangian multipliers.

The main aspects of the proposed algorithm that need further research in order to render it suitable for the implementation in real networks are the scalability and the fact that an effective CAC algorithm must be adaptive with respect to time-varying traffic statistics. In this last scenario, where the LP method is obviously not applicable, the RL algorithm is no more aimed at converging to a stationary optimal policy, but it must be able to continuously compute a (sub-optimal) policy which ‘follows’ the variations of the statistical characteristics. Effective methodologies exist in the literature for improving the RL scalability, such as using functional approximation of the (state,action)-value function and/or of the mapping π (see (Busoniu, Babuska, De Schutter, & Ernst, 2010; Xu, Zuo, & Huang, 2014) and references therein), or implementing state-space and policy-space approximation techniques (as in the CAC algorithms in (Pietrabissa, 2008a, 2009a)). Also, we are considering recent advances in sampling theory and in model-based RL ((Gheshlaghi Azar, Munos, & Kappen, 2013; Szita & Szepesvári, 2010)). Model-based RL relies on a generating model of the environment, that is continuously used to sample different paths with respect to the one actually explored on-line. In this framework, for the sake of the scalability, we are considering local generating models. To obtain an adaptive approach, the CAC algorithm must compute estimates of the current traffic characteristics, in terms of arrival rate, duration, bitrate; the generating model itself is then updated on-line (e.g., as in (Adam, Buşoniu, & Babuška, 2012; Grondman, Vaandrager, Busoniu, Babuska, & Schuitema, 2012)), based on the most recent estimates.

Acknowledgments

Research partially supported by the EU FP7-ICT MONET project, under Grant Agreement n° 247176. The authors thanks Prof. F. Delli Priscoli, Dr. A. Fiaschetti, Dr. G. Oddi, Dr. G. Tamea, Dr. M. Proia, Prof. C. Bruni and the researchers who worked in the MONET project for their help and advices.

References

Adam, S., Buşoniu, L., & Babuška, R. (2012). Experience replay for real-time reinforcement learning control. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, *42*(2), 201–212. doi:10.1109/TSMCC.2011.2106494

Altman, E. (2002). Applications of Markov decision processes in communication networks. *Handbook of Markov Decision Processes*.

Altman, E., Boulogne, T., El-Azouzi, R., Jiménez, T., & Wynter, L. (2006). A survey on networking games in telecommunications. *Computers & Operations Research*, *33*(2), 286–311. doi:10.1016/j.cor.2004.06.005

Bertsekas, D. P. (2005). Dynamic Programming and Suboptimal Control: A Survey from ADP to MPC. *European Journal of Control*, *11*(4-5), 310–334. doi:10.3166/ejc.11.310-334

Bertsekas, D. P., & Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation*. *Integers*. Prentice Hall Inc.

Busoniu, L., Babuska, R., De Schutter, B., & Ernst, D. (2010). *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press.

Choi, J. C. J., Kwon, T. K. T., Choi, Y. C. Y., & Naghshineh, M. (2000). Call admission control for multimedia services in mobile cellular networks: a Markov decision approach. *Proceedings ISCC 2000. Fifth IEEE Symposium on Computers and Communications*. doi:10.1109/ISCC.2000.860701

De Cicco, L., Mascolo, S., & Niculescu, S.-I. (2011). Robust stability analysis of Smith predictor-based congestion control algorithms for computer networks. *Automatica*, *47*(8), 1685–1692. doi:10.1016/j.automatica.2011.02.036

Delli Priscoli, F. (1999). Design and implementation of a simple and efficient medium access control for high-speed wireless local area networks. *IEEE Journal on Selected Areas in Communications*, *17*(11), 2052–2064. doi:10.1109/49.806833

Delli Priscoli, F., & Pietrabissa, A. (2004). Design of a bandwidth-on-demand (BoD) protocol for satellite networks modelled as time-delay systems. *Automatica*, *40*(5), 729–741. doi:10.1016/j.automatica.2003.12.013

Gábor, Z., Kalmár, Z., & Szepesvári, C. (1998). Multi-criteria reinforcement learning. In *ICML '98 Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 197–205). Madison, Wisconsin, USA, CA: Morgan Kaufmann.

Geibel, P. (2007). Reinforcement Learning Approaches for Constrained MDPs. *International Journal of Computational Intelligence Research*, *3*(1), 16–20. doi:10.5019/j.ijcir.2007.78

Gheshlaghi Azar, M., Munos, R., & Kappen, H. J. (2013). Minimax PAC bounds on the sample complexity of reinforcement learning with a generative model. *Machine Learning*, *91*(3), 325–349. doi:10.1007/s10994-013-5368-1

Grondman, I., Vaandrager, M., Busoniu, L., Babuska, R., & Schuitema, E. (2012). Efficient Model Learning Methods for Actor-Critic Control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*. doi:10.1109/TSMCB.2011.2170565

Hillier, F. S., & Lieberman, G. J. (2001). *Introduction to Operations Research*. *Foundations* (Vol. 6). McGraw-Hill. doi:10.1016/j.jacc.2004.11.012

Jiao, W., Sheng, M., Lui, K.-S., & Shi, Y. (2014). End-to-End Delay Distribution Analysis for Stochastic Admission Control in Multi-hop Wireless Networks. *IEEE Transactions on Wireless Communications*, *13*(3), 1308–1320. doi:10.1109/TWC.2013.013014.122055

Kalyanasundaram, S., Chong, E. K. P., & Shroff, N. B. (2001). Admission control schemes to provide class-level QoS in multiservice networks. *Computer Networks*, *35*(2-3), 307–326. doi:10.1016/S1389-1286(00)00173-0

Kalyanasundaram, S., Chong, E. K. P., & Shroff, N. B. (2002). Optimal resource allocation in multi-class networks with user-specified utility functions. *Computer Networks*, *38*(5), 613–630. doi:10.1016/S1389-1286(01)00275-4

Khoukhi, L., Badis, H., Merghem-Boulahia, L., & Esseghir, M. (2013). Admission control in wireless ad hoc networks: a survey. *EURASIP Journal on Wireless Communications and Networking*, *2013*(1), 109. doi:10.1186/1687-1499-2013-109

Klessig, H., Fehske, A., & Fettweis, G. (2014). Admission control in interference-coupled wireless data networks: A queuing theory-based network model. In *2014 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, WiOpt 2014* (pp. 151–158). doi:10.1109/WIOPT.2014.6850293

Krishnamurthy, V., & Leung, V. C. M. (2006). Cross-Layer optimal connection admission control for variable bit rate multimedia traffic in packet wireless CDMA networks. *IEEE Transactions on Signal Processing*, *54*(2), 542–555. doi:10.1109/TSP.2005.861785

Lilith, N., & Dogancay, K. (2005). Using Reinforcement Learning for Call Admission Control in Cellular Environments featuring Self-Similar Traffic. In *TENCON 2005 - 2005 IEEE Region 10 Conference* (pp. 1–6). IEEE. doi:10.1109/TENCON.2005.300835

Manfredi, S. (2012). A consensus based rate control scheme for ATM networks. *International Journal of Control, Automation and Systems*, *10*(4), 817–823. doi:10.1007/s12555-012-0418-1

Oddi, G., Panfili, M., Pietrabissa, A., Zuccaro, L., & Suraci, V. (2013). A Resource Allocation Algorithm of Multi-cloud Resources Based on Markov Decision Process. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science* (Vol. 1, pp. 130–135). IEEE. doi:10.1109/CloudCom.2013.24

Panfili, M., & Pietrabissa, A. (2013). A lexicographic approach to constrained MDP Admission Control. In *21st Mediterranean Conference on Control and Automation* (pp. 1428–1433). Chania, GR: IEEE. doi:10.1109/MED.2013.6608908

Pietrabissa, A. (2008a). Admission Control in UMTS Networks based on Approximate Dynamic Programming. *European Journal of Control*, *14*(1), 62–75. doi:10.3166/ejc.14.62-75

Pietrabissa, A. (2008b). An Alternative LP Formulation of the Admission Control Problem in Multiclass Networks. *IEEE Transactions on Automatic Control*, *53*(3), 839–845. doi:10.1109/TAC.2008.919516

Pietrabissa, A. (2009a). A policy approximation method for the UMTS connection admission control problem modelled as an MDP. *International Journal of Control*, *82*(10), 1814–1827. doi:10.1080/00207170902774233

Pietrabissa, A. (2009b). Optimal Call Admission and Call Dropping Control in Links with Variable Capacity. *European Journal of Control, 15*(1), 56–67. doi:10.3166/ejc.15.56-67

Pietrabissa, A. (2011). A Reinforcement Learning Approach to Call Admission and Call Dropping Control in Links with Variable Capacity. *European Journal of Control*, *17*(1), 89–103. doi:10.3166/ejc.17.89-103

Pillai, A., Hu, Y. F., & Halliwell, R. (2013). An Adaptive Connection Admission Control Algorithm for UMTS Based Satellite System with Variable Capacity Supporting Multimedia Services. In *4th International ICST Conference (PSATS 2012)* (pp. 9–16). Bradford, UK. doi:10.1007/978-3-642-36787-8_2

Puterman, M. L. (1994). *Markov Decision Processes*. (M. L. Puterman, Ed.). New York: John Wiley & Sons, Inc. doi:10.1002/9780470316887

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: an introduction*. MIT Press, Cambridge, MA.

Szita, I., & Szepesvári, C. (2010). Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27 th International Conference on Machine Learning (ICML-10)* (pp. 1031–1038). Haifa, Israel.

Xu, X., Zuo, L., & Huang, Z. (2014). Reinforcement learning algorithms with function approximation: Recent advances and applications. *Information Sciences*, *261*, 1–31. doi:10.1016/j.ins.2013.08.037

Zhou, J., Sun, L., Liu, T., Zhang, T., & Xiao, F. (2014). An admission control scheme based on the game theory for LEO satellite networks. In *The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014)* (pp. 525–530). doi:10.1109/ICSAI.2014.7009343