A robust control-theory-based exploration strategy in deep reinforcement learning for virtual network embedding

Ghina Dandachi [a],[*], Sophie Cerf [b], Yassine Hadjadj-Aoul [a], Abdelkader Outtagarts [c], Eric Rutten [d]

[a] *INRIA, Univ Rennes, CNRS, IRISA, Rennes, France*
[b] *Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 CRIStAL, Lille, France*
[c] *Nokia-Bell Labs, France*
[d] *Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, Grenoble, France*

ARTICLE INFO

ABSTRACT

Network slice management and, more generally, resource orchestration should be fully automated in 6G networks, as envisioned by the ETSI ENI. In this context, artificial intelligence (AI) and context-aware policies are certainly major options to move in this direction and to adapt service delivery to changing user needs, environmental conditions and business objectives. In this paper, we step towards this objective by addressing the problem of optimal placement of dynamic virtual networks through a self-adaptive learning-based strategy. These constantly evolving networks present, however, several challenges, mainly due to their stochastic nature, and the high dimensionality of the state and the action spaces. This curse of dimensionality requires, indeed, a broader exploration, which is not always compatible with a real-time execution in an operational network. Thus, we propose DQMC, a new strategy for virtual network embedding in mobile networks combining a Deep Reinforcement Learning (DRL) strategy, namely a Deep Q-Network (DQN), and Monte Carlo (MC). As learning is costly in time and computing resources, and sensitive to changes in the network, we suggest a control-theory-based techniques to dynamically leverage exploration in DQMC. This leads to fast, efficient, and sober learning compared to a Monte Carlo-based strategy. This also ensures a reliable solution even in the case of a change in the requests' sizes or a node's failure, showing promising perspectives for solutions combining control-theory and machine learning.

1. Introduction

Network slicing has been introduced by the NGMN 5G paper [1], where logical networks are enabled to use a common physical substrate infrastructure. This contributes to create a flexible stakeholder ecosystem and a programmable software-defined network environment by integrating technical and business innovation sustained by network and cloud resources. Network slicing is accomplished via the definition of network slice instances (NSIs), a.k.a. Virtual Network Requests (VNRs), composed of Virtual Network Functions (VNF). Different VNRs can be deployed within the same physical network to support a wide variety of services with different requirements, such as enhanced mobile broadband (eMBB), massive machine type communication (mMTC), or ultra-reliable low latency communication (uRLLC).

In this paper, we consider the virtual network slice request placement problem also known in the literature as Virtual Network Embedding (VNE) or VNF Forwarding Graph Embedding (VNF-FGE). It is generally common to model this problem as an integer linear program,

with an objective of maximizing the revenue-to-the-cost (R2C), thus using the lowest amount of physical resources for placing a VNR [2]. The VNE problem being NP-hard, with very long convergence time, even for small network instances, several heuristics have been proposed to solve it [3–5]. More recently, DRL techniques have emerged and allow achieving qualitatively better results [3].

One of the main difficulties in using DRL-based strategies is the time required to learn the good neural network weights so that the strategy converges. On the other hand, in slices' placement problems with dynamic arrival and departure of slice requests, it is mandatory to adapt to changes in the network and to be resilient to the stochastic nature of the requests arrival.

The combined use of Deep Reinforcement Learning (DRL) and control theory benefit from increasing interest in the last years for its perspectives of safe, efficient, robust, scalable and explainable decision-making [6]. In this work, a combination of a DRL strategy, namely

---

* Corresponding author.
E-mail addresses: Ghina.Dandachi@inria.fr (G. Dandachi), sophie.cerf@inria.fr (S. Cerf), yassine.hadjadj-aoul@irisa.fr (Y. Hadjadj-Aoul), abdelkader.outtagarts@nokia-bell-labs.com (A. Outtagarts), eric.rutten@inria.fr (E. Rutten).

Table 1
Summary of Notations.

Notation	Description
b_{l^s}	Available bandwidth at substrate link l^s
b_{l^v}	Bandwidth request at VL l^v
$b_{l^v}^s$	Allocated bandwidth for VL l^v at substrate link l^s
c_{n^s}	Available CPU at substrate node n^s
c_{n^v}	CPU request at VNF n^v
$c_{n^v}^s$	Allocated CPU for VNF n^v at substrate node n^s
$\mathcal{G}_s, \mathcal{G}_v$	The substrate network and VNR graphs
L_s, L_v	Set of substrate links and links
N	Number of MC iterations
N_s, N_v	Set of substrate nodes and VNFs
R	Revenue to the cost
λ	VNRs arrival rate

Deep Q-Network (DQN), and control theory is proposed by creating a feedback loop controlling the DQN's exploration strategy. The control-based dynamic exploration strategy, presented in this paper, improves the exploration efficiency, while positively contributing on the placements' performance, by enhancing the convergence speed and increasing fastness and accuracy of the VNR placements, as well as soberness in computing resources. Besides the increased robustness regarding the VNR load variations, it provides for more safety and trustability by avoiding erratic decisions for new situations, in which the data used for learning may no longer match the current situation.

The next sections are organized as follows: Section 2 introduces the virtual network placement problem and the DQN placement solution. Section 3 introduces the state of the art. Section 4 presents the dynamic exploration control strategies, the main contribution of this paper. Section 5 presents the experimental evaluation and Section 6 concludes the paper.

2. Virtual network embedding using reinforcement learning

2.1. General description

In the following, the general problem of dynamic virtual network embedding is considered, in which the services' arrival follows a stochastic process (i.e., online service placement). Without loss of generality, we consider the consecutive arrival of requests for the placement of virtual services on top of an underlying network, the substrate network (Fig. 1).

Substrate network. We model the substrate network as an undirected graph $\mathcal{G}_s = (N_s, L_s)$, where N_s and L_s denote the substrate nodes and links respectively. The CPU capacity of a substrate node is denoted by c_{n^s} with $n^s \in N_s$, and the bandwidth capacity of a link by b_{l^s} with $l^s \in L_s$.

Virtual network request (VNR). Similarly, we model an arriving VNR as an undirected graph $\mathcal{G}_v = (N_v, L_v)$, where N_v and L_v denote the substrate nodes and links respectively. The CPU demand of a VNF is denoted by c_{n^v} with $n^v \in N_v$, and the bandwidth demand of a link by b_{l^v} with $l^v \in L_v$ (see Table 1).

We assume that the VNRs arrive dynamically according to a Poisson process with an arrival rate λ, each with a different CPU and bandwidth request, and that each Virtual Network (VN) has an associated lifetime measured in time units, following an exponential distribution.

2.2. VNE problem

We assume that a VNR is successfully deployed if the Virtual Nodes Mapping (VNM) and Virtual Links Mapping (VLM) to the substrate network meets its CPU and bandwidth requirements.
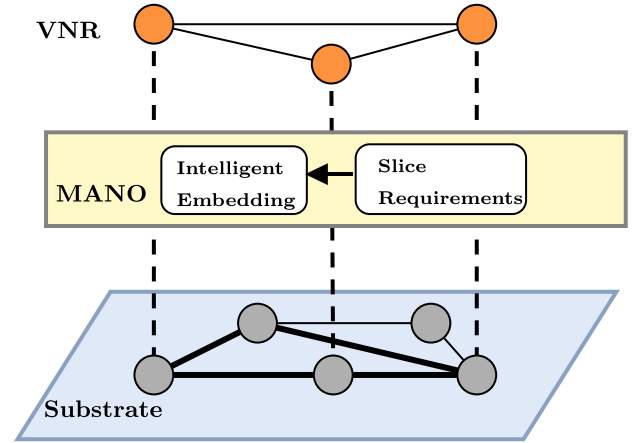
$$f_{VNM} : N^v \rightarrow N^S \tag{1}$$

Fig. 1. System model.

$$f_{VLM} : L^v \rightarrow L^S \tag{2}$$

We consider that the VNM function f_{VNM} is injective, that is, two VNFs of the same VNR cannot be hosted by the same substrate node. The following equations ensure that the resource constraints are met for both CPU and bandwidth resources, respectively:

$$c_{n^v} \leq c_{f_{VNM}(n^v)}, \forall n^v \in N^v \tag{3}$$

$$b_{l_v} \leq \min_{l^s \in f_{VLM}(l^v)} b_{f_{VML}(l^v)} \tag{4}$$

The management and orchestration of this placement is controlled by an intelligent framework compatible with the latest 5GPPP [7]. We consider that the intelligent embedding function in Fig. 1 is guided by the ETSI-ENI Network Function Virtualization Orchestrator (NFVO).

2.3. MDP model

Markov Decision Process (MDP) is a discrete time stochastic control process that provides a formalism for reinforcement learning algorithms. It can be defined as a tuple $\langle S, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where S, $\mathcal{A}$, $\mathcal{P}$ and $\mathcal{R}$, represent respectively the state space, the action space, the state transition probability matrix, and the reward function.

The MDP for the VNE problem can be described as follows:

States. The state of the system is defined as:

$$S = C_s \times B_s \times D_s \times C_v \times B_v \times D_v$$

where C_s is a vector representing the available CPU at each substrate node, B_s the available bandwidth at each substrate link, D_s the vector representing the substrate nodes' degree in order to give information about the nodes connectivity. Similarly, C_v, B_v, and D_v, are the CPU request, bandwidth request and the node degree of each virtual node of the VNR.

It is important to note that in MDP problems the current system's state is independent of previous and future states.

Actions. The output of the DQN is the probability distribution for the substrate nodes' selection. A random Gaussian noise is, then, added for a better exploration of the action space. Subsequently, a filtering is applied to keep only feasible nodes' placement solution, for a faster convergence of the learning process. Finally, the feasible node with the highest probability is chosen for the placement of the virtual node.

Reward. The learning reward $\mathcal{R}$ is set to the revenue to the cost metric (R2C), which is defined in Eq. (9). The reward $R_{s_t}^{a_t}$ at time t is obtained after selecting the action a_t at state s_t. The discounted reward $\hat{R}_t$ is:

$$\hat{R}_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{5}$$

Transition probability. The transition probability matrix $\mathcal{P}$ represents the dynamics of the system in terms of the state transition probabilities. It gives the probability of landing in a particular state knowing that one was in state and took a particular action. The state transition probability can be given by:

$$P_{s_{t+1}, s_t} = P[s_{t+1} | s_t, a_t], \tag{6}$$

where s_t and a_t are the state and the action at time t. In MDP, there are two value functions that can be used to find the best policy π: the state-value function $V_\pi(s)$ and the action-value function $q_\pi(s, a)$.

The usage of value-based algorithms becomes, unfortunately, impossible in the case of continuous state space. Hence, a DRL-based strategy was adopted, since it allows to optimize directly the policy, without prior knowledge of the model. Thus, the agent will have to learn through the experiences (i.e., exploration and exploitation) to take the right actions.

In this paper, a DQN-based strategy is considered, which is a value-based strategy: it predicts the value Q (Q-value) of taking an action a_t in a particular state s_t.

The Q-value update equation, which is derived from the Bellman equation, can be given as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \, \text{Loss} \tag{7}$$

where α is the learning rate. With γ being the discount factor, and $r(t)$ being the instantaneous reward value from Eq. (9), the Loss is given by:

$$\text{Loss} = (r(t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))^2 \tag{8}$$

2.4. Embedding strategy: DQMC

In this paper, we propose the usage of the DQN strategy, which is preceded by a features extraction layer composed of two fully connected three-layers neural networks (NN). DQN is a critic learning policy that uses a target network and experience replay to stabilize its results. This strategy performance is improved by adding a MC-based selection of a VNR placement: the best solution obtained from N iterations is chosen, we call this strategy: DQMC [8]. The DQMC algorithm is described in Alg. 1. The placement quality is evaluated using the R2C metric defined as:

$$R = \frac{\sum_{n^v} c_{n^v} + \sum_{l^v} b_{l^v}}{\sum_{n^v} \sum_{n^s} c_{n^v}^{n^s} + \sum_{l^v} \sum_{l^s} b_{l^v}^{l^s}} \tag{9}$$

Our aim is to maximize the R2C metric while reaching stability faster. For this reason, N_{iter} possible embeddings are explored using the DQN strategy, and the R2C is evaluated for each embedding solution. However, only the set of actions that achieves the highest R2C is stored in the replay memory. This set of actions consists of a series of DRL steps to place all virtual nodes onto the substrate nodes, and must offer feasible nodes and links' mapping.

2.5. Need for a dynamic exploration

Two DRL-related problems are addressed in this section, namely the fact that it requires expensive exploration in order to get its best performance and that it is sensitive to changes in the environment, which makes its decisions less safe.

Fig. 2 introduces the performance of the VNE implemented with a simple DQN strategy, i.e. without the control of the exploration and without the combination with Monte Carlo. DQN's performance is compared to that of MC strategy with a fixed number of iterations ($N = 8$). DQN approaches MC's performance with an average R2C of $R = 0.64$ only after a long learning phase that takes about $300k$ timesteps to converge.

To tackle such a problem, we propose to augment the DRL strategy with a control loop adapting the exploration in reaction to changes

---

Algorithm 1: DQMC algorithm

Data: N_{iter}, current state s_t, number of VNFs to place K
Result: VNR nodes' mapping to substrate network
while $n < N_{iter}$ **do**
    for $k \in K$ **do**
        Get current state;
        Action:
          • Calculate the nodes distribution as a function of the current state;
          • Adding a random Gaussian noise to the output;
          • Select feasible substrate nodes (sufficient resources, and a node not selected previously);
          • Select substrate node with the highest probability;
    end
    Test virtual node mapping;
    Test virtual link mapping;
    Evaluate the R2C R from Eq. (9);
end
Select VNR placement with the highest R2C ;
Place the VNR;
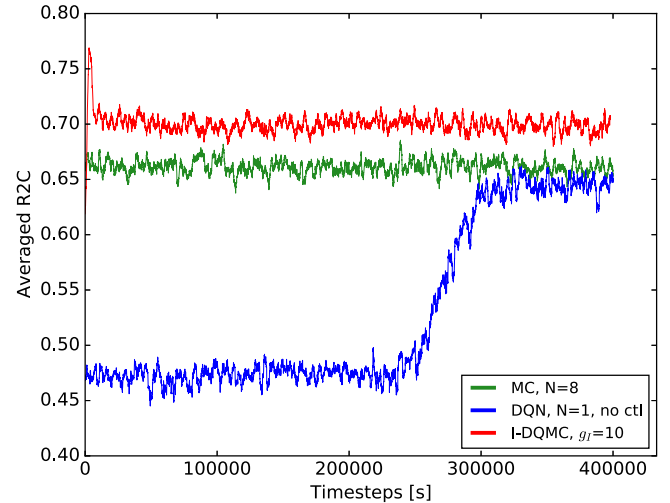Add state-action-state-reward to the replay memory;

Fig. 2. Dynamic exploration motivation: R2C averaged over 100 time steps for a DQN strategy without MC and without control; comparison with our solution I−DQMC.

in the environment, in order to sustain performance and safety, while reducing exploration cost. This control loop enables the dynamic adjustment of the number of MC iterations N. Indeed, the number of MC iterations impacts the learning speed of the DQN, as well as the learning quality. However, in the case of DQMC, N is fixed to a given number even after the system reaches stability. Besides, the learning quality varies when the incoming VNRs varies (i.e., the VNRs topology, the amount of requested resources, changes in the VNRs distribution ...).

We propose, in this paper, the adjustment of the number of MC iterations based on the obtained R2C value. The objective is to guarantee a better performance than the MC-based strategy and to improve the robustness of DQMC. The Integral control-based DQMC solution (I−DQMC) presented in this paper, that will be detailed in Section 4.2.2, converges faster and reaches a higher revenue-to-the-cost level than the other approaches, see Fig. 2.

3. Related works

3.1. Network slicing techniques

Network slicing mainly consists in placing services on a substrate network. Since services are represented by graphs (with system and network constraints), network slicing generalizes the "bin packing" problem, which involves placing only monolithic items (services). This generalization implies a much higher combinatorial complexity, categorizing network slicing as an NP-hard problem [9].

To tackle this combinatorial problem, there are typically three categories of approaches in the literature: exact methods [10], approximate and heuristic methods [4] and meta-heuristics [3]. Since exact problem-solving only works for fairly small problem instances, it is often necessary to use approximations or heuristics that allow converging very rapidly to a local minimum [4]. Meta-heuristics allow finding qualitatively more interesting solutions, but generally require a lot of computation to properly explore the solution space. Moreover, when using meta-heuristics, the majority of solutions are not viable, and therefore require a fix mechanism in order to transform a non-viable solution into a viable one, which increases the computational complexity even more, though it also presents the risk of converging into a local minima [5].

More recently, the service placement problem has also been addressed using machine learning techniques, the proposed strategies typically employ reinforcement learning [3,11]. Authors in [11] introduced the application of neural networks for a dynamic allocation of physical network resources to the virtual networks. This algorithm proposes an autonomous system that improves the resource utilization by acting on nodes' mapping. Authors in [12] proposed the use of Temporal-difference to learn the embedding solution that maximizes the long-term revenue. In Ref. [13], the Q-learning algorithm is used to allocate time slot and modulation coding scheme for a data transmission with transmitted data size being the reward function. Authors in [14] propose to use the Policy Gradient algorithm to gradually learn the optimal mapping mechanism. The algorithm applies the Policy Gradient method to the VNE domain and mainly to the nodes' embedding steps. This model learns how to strike a balance between exploring better solutions and developing existing models.

There exist also approaches that combine RL and heuristics. In [15], a Monte Carlo Tree Search (MCTS) strategy is proposed. It allows to find a sub-optimal solution to the placement problem, whereas the cost of a new research remains substantial since there is no learning. In [3], the authors proposed to combine a DRL strategy with a heuristic, to make the placement safer at the cost, however, of effectiveness.

Slicing was also addressed in virtual radio access network. Authors in [16] proposed a dynamic resource reservation and deep reinforcement learning-based autonomous virtual resource slicing framework. They introduced the possibility of optimization of resource allocation dynamically using a DQN. This work was extended in [17], in which the authors added a mixed traffic description, the system considers versatile users' quality of service. The results showed a good balance between the satisfaction and the resource utilization metrics.

The utilization of reinforcement learning techniques is very practical when labels are not available, which is the case in service placement problems. Moreover, reinforcement learning has been shown to be more effective in solving combinatorial problems, even when optimal labels are available [18].

When the model of the environment is known, it is possible to deduce the optimal policy, using dynamic programming [19]. However, in the general case it is difficult or even impossible to obtain it. This is particularly true for service placement in 5G and post-5G networks, since the state of the network is not known all the time and the placement impact on latency and loss metrics is complex to determine [20].

3.2. Exploration/exploitation trade-off in reinforcement learning

Despite the qualitative results achieved with DRL strategies for VNE problems, one of the major problems when using reinforcement learning in such a context is the trade-off existing between exploration and exploitation [19]. Exploration allows to better explore the space of solutions in order to increase the chances of finding better ones, with the risk of exploring less interesting or even useless regions. On the other hand, exploitation allows to use the knowledge accumulated to find new solutions, with the risk of converging to a local minima, if the exploration has not been sufficient. Some of the existing exploration/exploitation methods are: ϵ-Greedy [19], thompson sampling [21], bayesian techniques [22], and risk seeking utility function [23].

ϵ-Greedy is certainly one of the simplest approaches for exploring the action space [19]. The idea consists in exploring a random action with a probability equal to ϵ. It is quite practical to reduce the value of this parameter as the experiment progresses, but this makes the agent less efficient as the environment changes (i.e. change of service type, topological change, ...), which may happen in practical use cases. Thompson sampling, whose idea is to select an action (i.e. an arm) according to its probability of being the best, allows to obtain a result close to the optimum [21]. However, this type of approaches is only applicable in stationary systems and when the number of actions is very limited. This is clearly not the case for the services placement problems, as the environment is not stationary and the number of actions can be quasi-unlimited or even unlimited.

The authors, in [22], review recent work dealing with Bayesian techniques for reinforcement learning and which allow naturally a good compromise between exploration and exploitation. More recently, the author, in [23], derived an efficient exploration policy relying on propagating the certainty value corresponding to an epistemic risk-seeking utility function in the MDP, which led to a more efficient exploration. The latter strategies allow a better exploration of the action space, but present the disadvantage of not adjusting to very changing environments, unlike the strategy developed in this paper.

Recent years have seen the emergence of promising solutions combining machine learning with control theory. In [24], the exploration/exploitation trade-off is managed automatically using control theory, e.g. through controlling the ϵ hyper-parameter in the ϵ-greedy exploration technique [24]. In [25], the authors proposed Value Difference Based Exploration (VDBE), an approach that dynamically computes a state-dependent exploration probability, increasing the parameter robustness. However, to the best of our knowledge, there is no control approach for regulation of the exploration process that use Monte Carlo iterations as a control knob.

Note that although the proposed strategy is applied to service placement problems, it can be used to solve the exploration–exploitation dilemma in any combinatorial problem.

4. Dynamic control of the exploration in DRL

In short, a dynamical exploration strategy is proposed based on control theory. Exploration is leveraged by the number of Monte Carlo iterations trying out N output in the DQMC algorithm. The dynamic adaptation is based on an error, i.e. the difference between the current R2C and a reference R2C value. In this case, the DQMC is trained in competition with a simple Monte Carlo algorithm, with fixed number of iterations. Comparison with a reference algorithm enables the DQMC to take into account the task complexity, and thus to deal with the varying VNR patterns.

The core contribution of this paper is twofold. First, the exploration is formulated as a dynamical problem based on performance comparison with a reference followed by decision making. Second, this feedback adaptation is backed up on algorithms from control theory, and corresponding analysis tools of precision and rapidity. To highlight
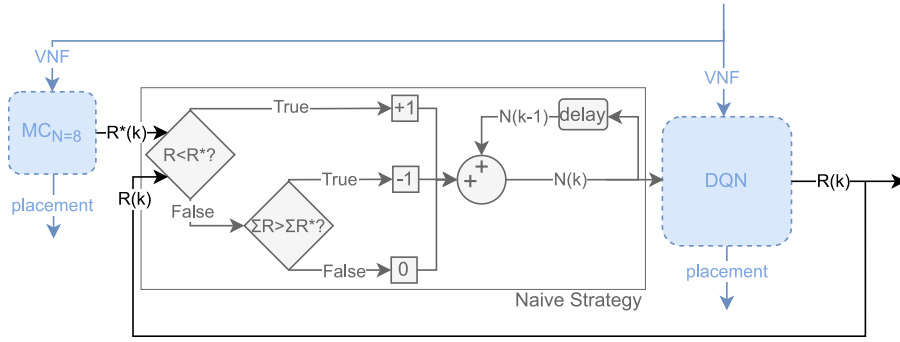
Fig. 3. Threshold-based exploration strategy. R2C comparison between DQMC and reference placement strategy followed by iteration increment decision. VNF placement tasks are depicted in blue, the threshold-based strategy in gray. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the additive benefits of those two points, two dynamic exploration laws are provided: a threshold-based strategy (Section 4.1), i.e. adaptation without control tools, and a control-based strategy (Section 4.2).

The dynamic exploration strategies have two steps, detailed hereafter for each one:

1. performance evaluation and comparison with a reference,
2. decision on the exploration depth.

4.1. Threshold-based dynamic exploration

As a first pedagogical approach to dynamically leverage RL exploration, we present the *threshold-based* strategy. In this naive approach, the following two steps are applied at each timestep. First, the R2C of the DQMC, denoted as $R(k)$ with k the timestep index, is compared with the R2C of the Monte Carlo-based strategy with fixed number of iterations $R^*(k)$, taken as a baseline. Then the exploration level – i.e. the number of iterations $N(k) > 1$ of the Monte Carlo-based strategy trying out DQN output – is incremented or decremented depending on the former condition. Eq. (10) formalize the threshold-based strategy while illustration is given by Fig. 3.

$$N(k) = \begin{cases} N(k-1) + 1 & \text{if } R(k) \leq R^*(k) \\ N(k-1) - 1 & \text{if } \sum_{i=k-H}^{k} R(i) > \sum_{i=k-H}^{k} R^*(i) \\ N(k-1) & \text{otherwise} \end{cases} \tag{10}$$

When the DQMC performance is lower than the reference value, exploration is increased. Conversely when DQMC significantly outperforms the reference, exploration is reduced. This second condition reduces N and thus computational costs when non necessary, i.e. when the placement task is performed accurately and no exploration is needed. The averaging on a horizon H for exploration aims at avoiding oscillations. Indeed with $H = 1$ and as R converges towards R^*, small variations on R could make the exploration N constantly oscillate between two values. Conversely a very large horizon H slows the process of exploration reduction. Note that the N update law here is a simple constant increment of ± 1. No tuning is needed, at the cost of a slow evolution and poor reactivity.

Emerges here the notions of stability and rapidity, and their contradictory nature. Dedicated tools are needed to properly address this trade-off.

4.2. Control-based exploration strategies

To overcome the expected limitations of the threshold-based approach – slow and possibly oscillatory – we adopt a control theory-based approach. Control relies on the feedback principle, taking action in accordance to the distance to the desired state, refer to [26] for an introduction of its use on computing systems. In our case, the

comparison condition of the threshold-based strategy (Section 4.1) is substituted by the quantification of the error Δ_R:

$$\Delta_R(k) = R^*(k) - R(k) \tag{11}$$

We propose three different control-based strategies: a proportional controller, an integral one and a controller combining proportional and integral action. This classification depicts the treatment done to the error $\Delta_R(k)$ in the computation of the exploration level $N(k)$ [27]. While the proportional P controller reacts to the present learning accuracy, the integral I control realizes adjustments with a memory of the past. All controllers manage differently the stability, rapidity and precision trade-off.

4.2.1. Proportional control P–DQMC

The proportional controller dynamically computes the exploration iterations $N(k)$ at each timestep depending of the R2C difference, as given in Eq. (12).

$$N(k) = g_P \, \Delta_R(k) \tag{12}$$

The parameter g_P, called controller gain, leverages the adaptation behavior: learning speed, performance achieved and cost awareness [27]. Sound choices for this gain will be discussed in Section 5.2.2.

4.2.2. Integral control I–DQMC

The second proposed control-based exploration strategy is *updated* proportionally to the error:

$$N(k) = N(k-1) + g_I \, \Delta_R(k) \tag{13}$$

The former law is called integral controller as Eq. (13) can be reformulated as:

$$N(k) = N(0) + g_I \sum_{i=0}^{k} \Delta_R(i),$$

highlighting that the current exploration level is proportional to the *integral* of the errors. The integral controller has by definition a memory of the past, which accounts for a higher precision in reaching the objective accuracy level. The gain g_I can be tuned to obtained the best trade-off between fast and accurate learning, see Section 5.2.2.
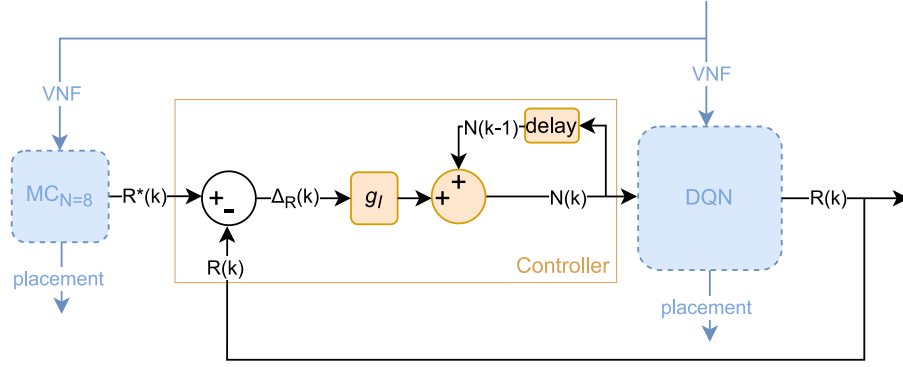
The integral control law is illustrated in Fig. 4. Note that the proportional control can be obtained from this representation by removing the addition of the delayed N signal.

#### 4.2.3. Proportional-integral control PI–DQMC

Advantages of both proportional and integral controller are combined in the PI-DQMC strategy. The exploration law is given is Eq. (14) [26].

$$N(k) = N(k-1) + (g_p + g_I) \, \Delta_R(k) - g_p \, \Delta_R(k-1) \tag{14}$$

There are two parameters in this formulation: g_p and g_I. This allows for a finer tuning of the learning characteristic in term of rapidity, precision and computation cost.

Fig. 4. Control-based dynamic exploration strategy for `I-DQMC`. Increment on iterations is computed from R2C difference following the integral control law. VNF placement tasks are depicted in blue, the integral controller in orange. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
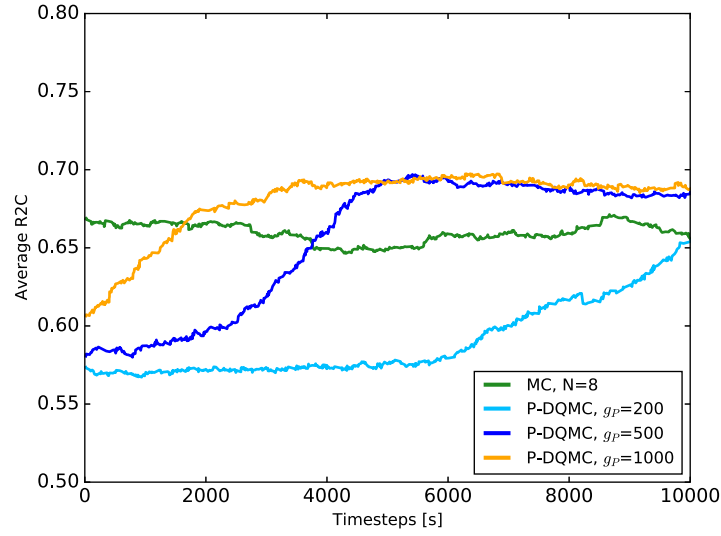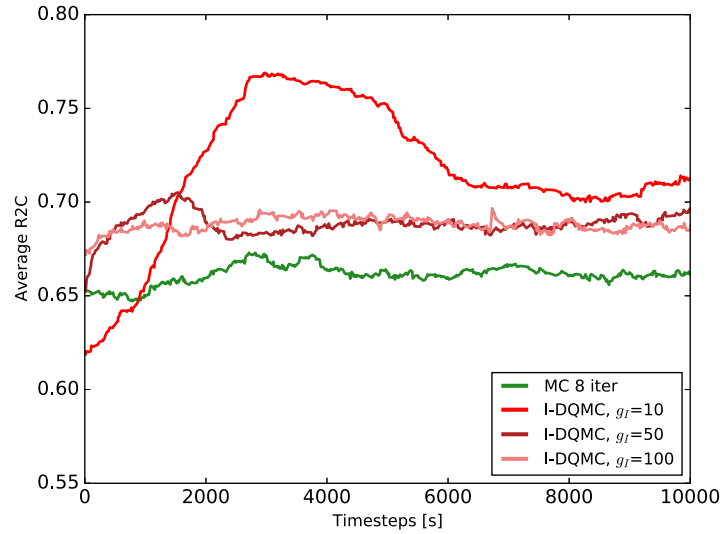

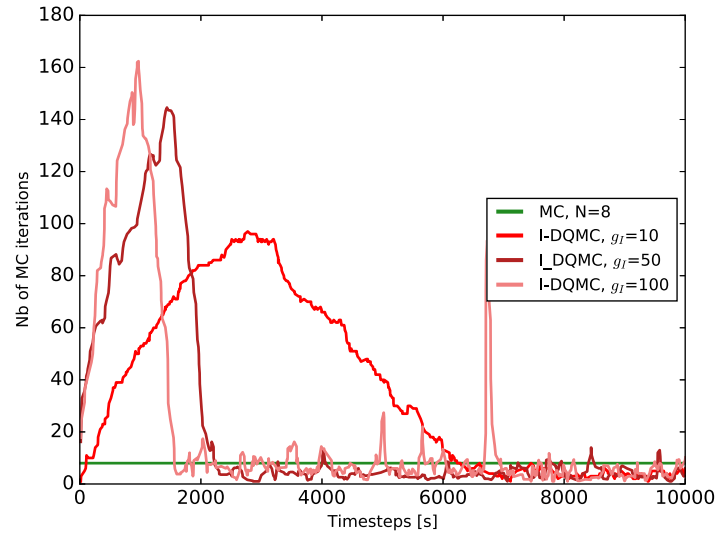(a) Revenue to the cost, averaged over 100 timesteps.
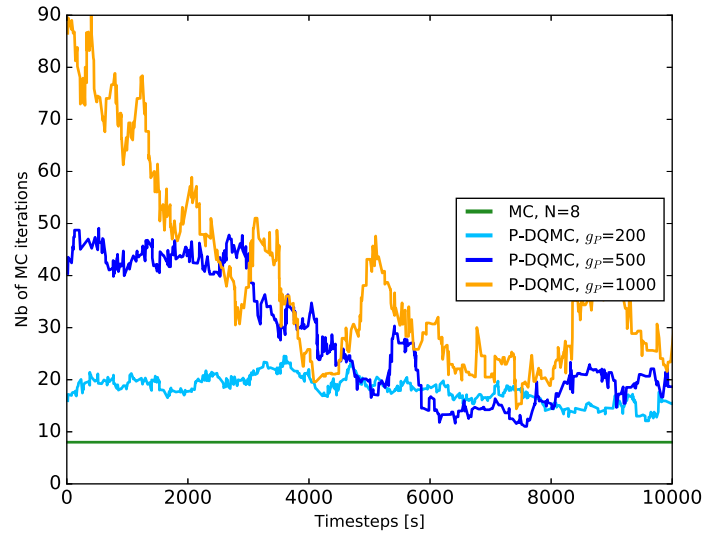
(b) Number of iterations, averaged over 30 timesteps.

Fig. 5. Evaluation of proportional control for dynamic DQMC exploration. Evaluation for various g_p parameters.

(a) Revenue to the cost, averaged over 100 timesteps.

(b) Number of iterations, averaged over 30 timesteps.

Fig. 6. `I-DQMC`: Integral control of dynamic DQMC exploration evaluation. Evaluation for various g_I parameters.

5. Evaluation

In this section, we evaluate our dynamic exploration strategies for various configurations and compare them with static (i.e., non-adaptive) approaches. Robustness analysis in case of node failure and VNR load variation is also provided. First, the simulation setup is presented.

5.1. Evaluation setup

To evaluate the performance of the proposed approach, the following setup is considered:

- The substrate network follows the btEurope topology [28] with 24 nodes. The capacity of the substrate nodes and links is drawn uniformly from the interval [50, 100].
- To generate the virtual requests, we use the Erdős–Rényi model [29]. In this model the generated graph is defined by the number of nodes n and the probability p of creating an edge between the nodes. With, for instance, $p = 2\frac{\ln n}{n}$, the generated graph is in

general connected (more precisely, this probability value goes to 1 as $n \to \infty$). The requested resources (CPU and bandwidth) of Virtual Network Requests are drawn randomly following a uniform distribution from the interval [5, 10]. The system operates in a dynamic manner: during each timestep, VNRs arrive to the system – with a mean time between arrival MTBA $\in \{1, 5, 10, 20, 40\}$ in seconds – and once a VNR is processed, the next one arrives. A VNR stays in the system between 5 and 10 timesteps.

- The model architecture is implemented in Python with the Pytorch library 2, and the DGL library 3. The neural network architecture is constructed with the following hyperparameters. Two fully connected 3-layers neural networks are used: one for substrate network features extraction, and the second for VNRs features extraction. Then, a fully connected layer is used, this layer takes as an entry the concatenation of both features extraction results and gives as an output the value network with 24 nodes. The learning rate is set to 5×10^{-4} and the discount factor γ is set to 0.95. To train the model, the Adam optimizer was used [30].
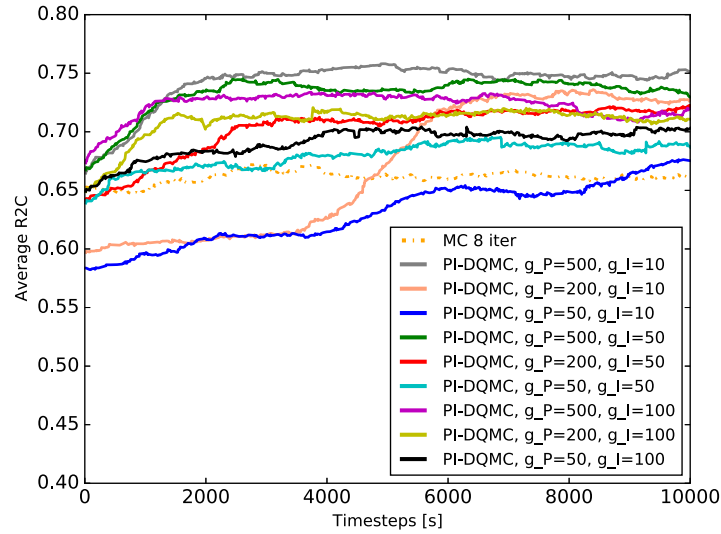
(a) Revenue to the cost, averaged over 100 timesteps.
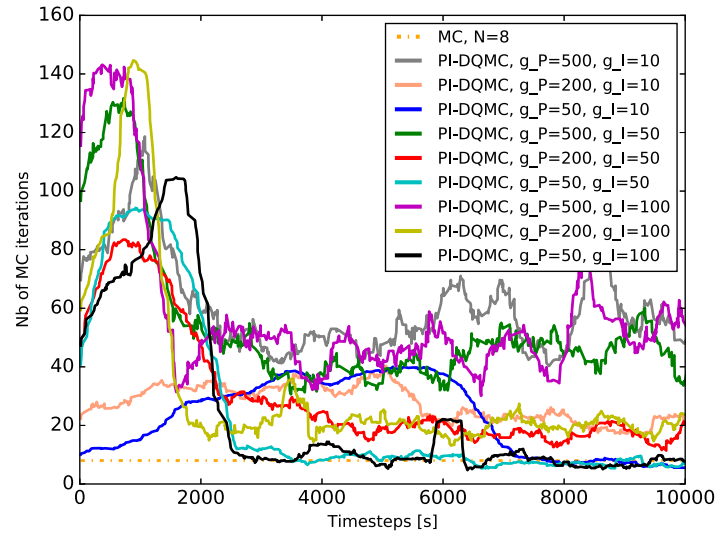
(b) Number of iterations, averaged over 30 timesteps.

Fig. 7. PI-DQMC: Proportional Integral control of dynamic DQMC exploration. Evaluation for various g_P, g_I parameters.

- The control reference value R^* is taken as the R2C of the Monte Carlo-based strategy only with fixed number of iteration $\text{MC}|_{N=8}$.

For better results visualization, figures represent the average R2C over 100 timesteps.

5.2. Behavior and performance of control strategies

We evaluate the performance of our control-based exploration strategies on a DQMC, where the number of iterations is set either by the Proportional (P), Integral (I) or Proportional-Integral (PI) law. Different values of the controllers' parameters are used. Evaluation criteria are the properties of stability, rapidity and precision of the learning, as well as the computational cost. First, we illustrate and explain the behavior of the controllers.

5.2.1. Proportional controller

Fig. 5(a) presents the measured revenue to the cost R through time obtained by the P-DQMC strategy, for various values of the parameter g_P. The reference value R^* given by the reference $\text{MC}|_{N=8}$ algorithm is

also reported. Fig. 5(b) shows the corresponding number of iterations computed by the controllers.

Choice of the controller parameter. N varies in $[1; 100]$, the error Δ_R is roughly lower than 0.1 and significant above 0.01 (given R variability, see Fig. 5(a) for instance). Given the relation on orders of magnitude in Eq. (12), g_P is chosen in $[100; 1000]$. Three values were used: $\{200; 500; 1000\}$.

Let us give insights on the P controller behavior. In the first timesteps of P-DQMC, R is significantly lower than R^*. The error Δ_R is, thus, large, which causes large values of N. Note, however, that a large number of iteration speeds up the learning and starts increasing the revenue to the cost R. While R stays lower than R^*, N keeps being large, however decreases as R reaches R^*. It is important to note that we ensure that the minimum number of MC iterations is equal to 1, so that at least one MC iteration is done, even when the Δ_R gives negative values of N. Then, the error Δ_R starts being negative and N converges. The revenue to the cost eventually converges to a same level no matter the configuration g_P, at a higher value than the reference R^*. The larger g_P, the larger N in the first timesteps, and the faster the learning, however at the cost of numerous iterations.
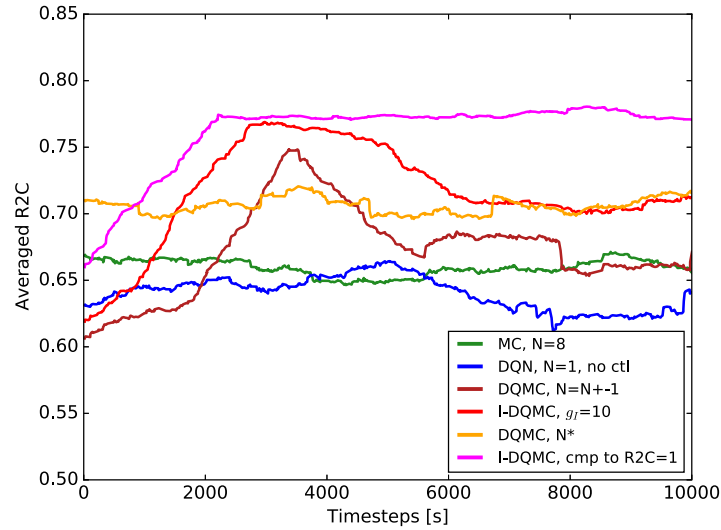
(a) Revenue to the cost, averaged over 100 timesteps.
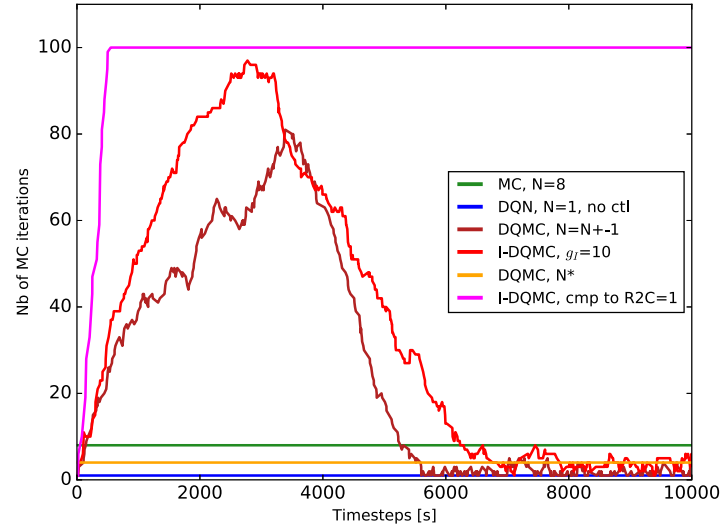
(b) Number of iterations, averaged over 30 timesteps.

Fig. 8. Comparison with competitive exploration strategies.

We observe that P-DQMC with $g_P = 500$ achieves a reasonable trade-off between R rapid increase and sober number of iterations.

5.2.2. Integral controller

We now evaluate the Integral strategy I-DQMC. Figs. 6(a) and 6(b) respectively show the average R and the number of iterations, for different values of the gain g_I.

Choice of the controller parameter. By considering a reasonable N update order of magnitude of about ± 1 at each time step and using Δ_R previous estimation, Eq. (13) gives rough bounds for $g_I \in [10; 100]$. Three values were used: $\{10, 50, 100\}$.

At the first timesteps, R is significantly lower than R^*: N increases at each timestep. By increasing the number of iterations N, the learning is improved, and eventually R meets R^*. Then, Δ_R becomes negative and N starts decreasing. However, as N stays large, R keeps increasing. When decreasing N becomes impossible as it would reach negative values, the controller settles N to 1. R slightly decreases as the number of MC iterations decrease, and finally converges to a value that is higher than R^*. Note that this behavior with a R peak reveals a poor learning: R is large because N is large, e.g. the number of iterations computed is such that even with a poor quality upstream DQN, R is

large. Keeping a high R value with low N is the real indicator of a learning improvement.

The larger g_I, the faster N increases but also decreases. Less timesteps with large N means less exploration. We observe that with a gain of $g_I = 10$, the I-DQMC strategy achieves the highest revenue to the cost R, with convergence after 7000 timesteps. With $g_I = 50$, I-DQMC reaches stability faster, at the expense of lower final R. This shows the trade-off between rapidity and final performance permitted by the integral controller parameterization. We also observe an overshoot — a peak in R value before convergence to a lower value. The peak value can be sustained only at the cost of very large number of iterations N, an undesirable situation in practice due to the excessive duration of a timestep it implies. The control smartly boosts learning in the initial moments to allow for high R level in steady state with very few iterations. With an average of only $N = 2.86$ after convergence ($k > 5,000$), the control-based I-DQMC reaches better R than MC$|_{N=8}$ in average (reps. 0.69 and 0.66, e.g. increase of +4.5%).

5.2.3. Proportional-integral controller

The benefits of the proportional and integral controllers – fast, efficient and sober learning – are combined in the PI-DQMC dynamic
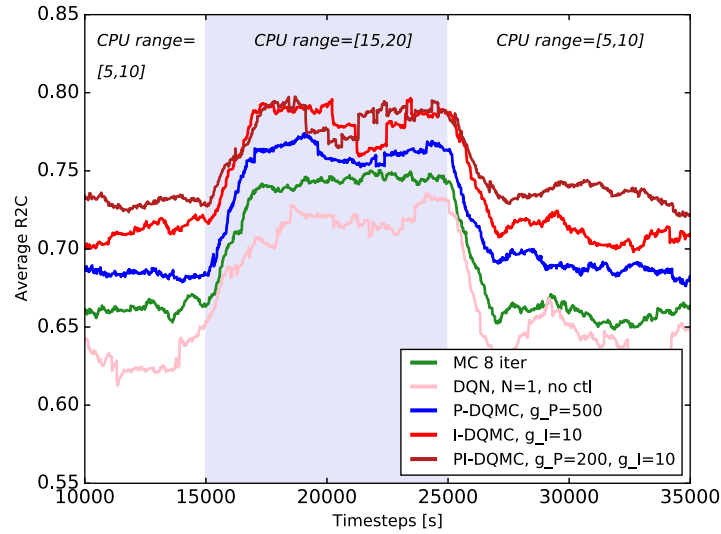
(a) R2C, averaged over 100 timesteps.
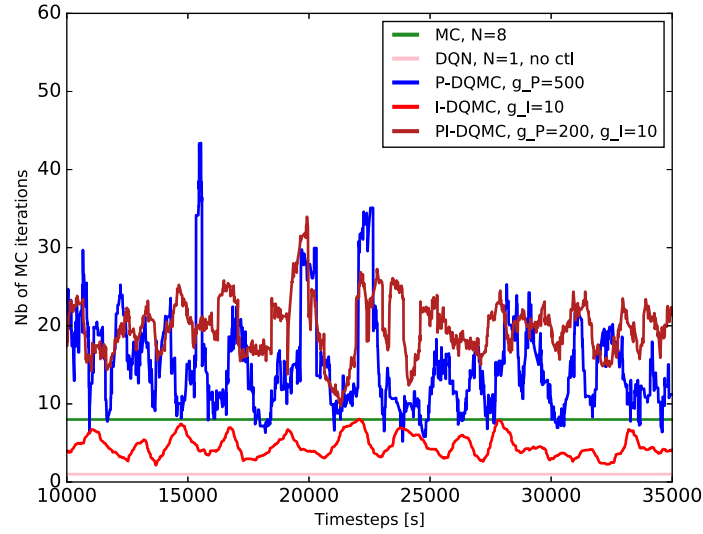
(b) Number of iterations, averaged over 30 timesteps.

Fig. 9. Robustness analysis: change of VNR size at $k = 15,000$ and $k = 25,000$. Initial convergence part is omitted.

exploration strategy. Figs. 7(a) and 7(b) present the average R and the number of iterations N through time in comparison with the reference R^*.

Choice of the controller parameters. we use the range of parameters of the proportional and integral controllers.

The first thing to note is that higher values of revenue over cost are achieved than with previous controllers, and in a short time. With large g_P, R increases fast and reaches higher values, at the cost of high N in average after stabilization (i.e. significant computing time). The larger is g_I, the faster is the learning, at the cost of large N peak. $g_P = 200$ and $g_I = 10$ achieves a fair compromise with high R2C and low number of MC iteration, despite a relative cold start.

To sum-up, our PI-DQMC allows to leverage between efficient (high R value), fast (rapid R increase), and sober (low N value at convergence) learning.

5.3. Comparison with competitive strategies

After evaluating the performance of our control approach, we compare to different exploration strategies. We first select static (i.e. non-dynamic, N fixed) competitive exploration strategies to illustrate the benefit of the dynamic perspective. The comparison with a naive threshold-based dynamic strategy highlights the relevance of the control theory. In detail, competitive strategies are:

- Monte Carlo-based with fixed number of iteration ($N = 8$), without DQN.
- DQN without control and only one iteration ($N = 1$). We show the system performance after it reaches convergence, after at least 300,000 iterations, see Fig. 2.
- DQMC with threshold-based dynamic exploration ($N = N \pm 1$).
- DQMC with N constant ($N^* = 4$, i.e. average number of iterations obtained using the integral strategy with $g_I = 10$). Again, we show the obtained R after the system converges and learns to find optimal solutions, which takes around $70k$ timesteps.
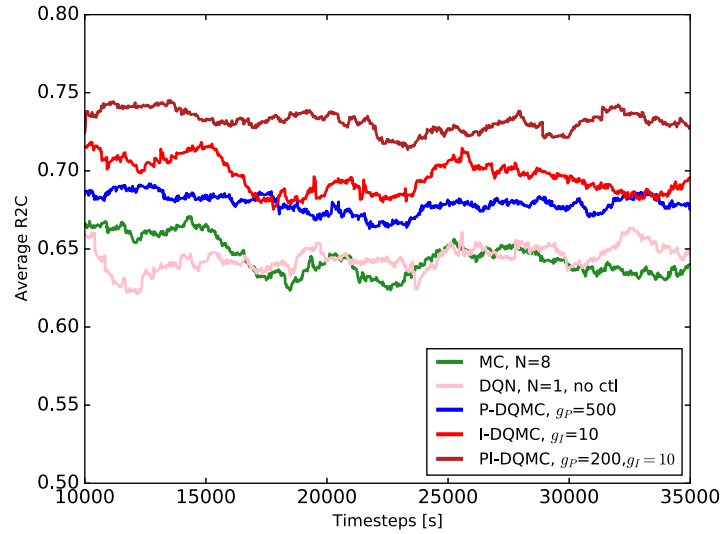
(a) R2C, averaged over 100 timesteps.
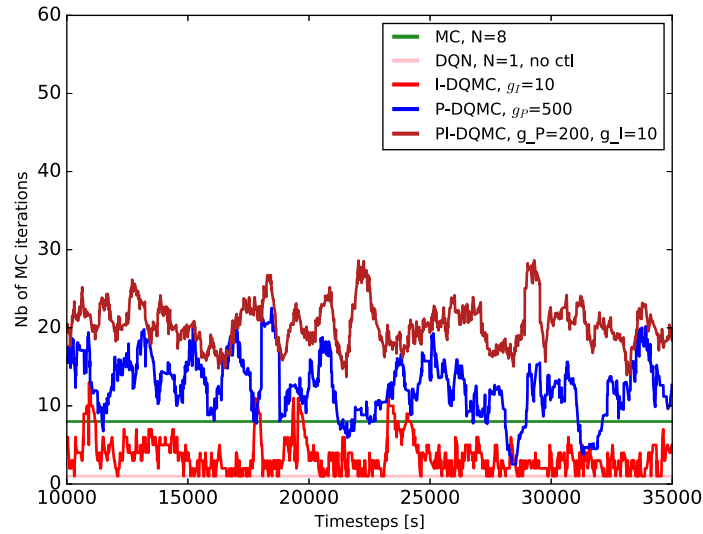
(b) Number of iterations, averaged over 30 timesteps.

Fig. 10. Robustness analysis: the case of Node failure.

- I-DQMC with $g_I = 10$, where the ideal reference is not based on Monte Carlo but fixed at $R^* = 1$

We select the I-DQMC strategy with $g_I = 10$ as our control baseline, as it achieves outstanding trade-off between learning efficiency and rapidity, and reasonable number of iteration. Revenue to the cost R and number of iterations N for each strategy are presented respectively in Figs. 8(a) and 8(b). In comparison with static strategies ($N = 1$, $MC|_{N=8}$), I-DQMC reaches higher revenue to the cost. Similar results are achieved with N^* strategy, which is expected as N^* is the average of I-DQMC's N after convergence. Our control based strategy however converges in only $6k$ timesteps against $70k$ for N^*, that is a reduction of 86%.

When comparing to the naive dynamic strategy ($N = \pm 1$), I-DQMC again allows for higher R and even reaches high R values faster. Even if the dynamic N variation look similar, the control-based strategy has better performances: this mathematically founded dynamic exploration law makes the difference.

The best performance of all is achieved by the control strategy with $R^* = 1$. However, the number of iteration saturates at its upper bound $N = 100$. Indeed as R never reaches 1, N is never large enough

and thus the controller keep increasing it. Such high values of N are not sustainable in practice, as it represents very long and costly computations. Using $R^* = MC|_{N=8}$ then reveals all its usefulness: it allows for more efficient and faster learning than all other competitive strategies, with number of iterations practically realizable.

5.4. Robustness analysis

Three aspects of robustness are evaluated hereafter: robustness coming from a fast convergence of the placement to high performance, robustness to a change in the incoming distribution of the VNRs' size, and robustness to a node failure in the substrate network.

5.4.1. Robustness from rapid convergence

Rapid convergence implies robustness in the sense that transitory sub-optimal decisions' phases are shorter with the control-based DQMC, potential detrimental decision are avoided. This method ensures the automation of Virtual network embedding in 6G Zero Touch Network.

5.4.2. Change in the VNRs' size

We consider a variation of VNRs CPU requests happening at 15k timesteps that ends at 25k timesteps. More precisely, we consider a CPU requests range of $[5, 10]$ from 0 to 15k, then it increases to $[15, 20]$ between 15k and 25k, and eventually returns to $[5, 10]$. The evaluated R through time of the different strategies are reported in Fig. 9(a), and the corresponding variations of the number of iterations in Fig. 9(b). The initial convergence part is omitted. Control-based strategies require around 8k timesteps to converge, while DQN require around 350k timesteps to converge. The timesteps axis between 10k and 35k corresponds actually to 310k and 335k for the DQN strategy, this presentation was adopted in order to have comparable plots. The parameter chosen for each of the three controllers are the parameters that offer the optimal R/Number of iterations trade-off.

Even when the task changes, all control-based DQMC strategies are able to adapt and maintain a higher R than R^*. The I and PI controllers have similar performance in terms of R, while I controllers have a lower number of MC iterations. After increasing the CPU request at 15k, we observe an increase in R. This is related to the fact that when increasing the size of CPU requests, the difference between the revenue and the cost decreases and thus R increases.

To sum-up, the advantage of control-based strategies are: (1) rapidity of convergence around 6k timesteps compared to 350k timesteps for DQN, (2) robustness of the solution by staying less time in the suboptimal region and by reactively adapting with system changes when the task changes.

5.4.3. Substrate node failure

We now consider a different type of perturbation that can be managed by the controllers: substrate node failure. We show the control behavior (R in Fig. 10(a) and N in Fig. 10(b)) of the Integral controller ($g_I = 10$), the Proportional controller ($g_P = 500$), and the PI controller ($g_P = 200$, $g_I = 10$) with a node failure happening at $t = 15k$ [s]. The proportional and the PI controllers are able to keep a stable performance contrarily to $MC|_{N=8}$ and the integral controller, for which a drop in the R performance is visible. Thus, the revenue to the cost of the PI-DQMC is the highest at the expense of a higher number of iterations N. I-DQMC shows however a better trade-off between the revenue-to-the-cost R and the number of iterations N.

6. Conclusion

In this paper, we proposed a combination between Deep Reinforcement Learning and Control strategies for robust 6G slices embedding and resource management. In short, we combine a Deep Q-Network with Monte Carlo (DQMC), with a control-theory-based exploration strategy. The proposed approach allows increasing the exploration of the system in a short time compared to the long-time required for DQN to converge in a system without control. We compared different aspects for VNE including robustness, especially in the case of change in the pattern of VNR requests and studied the system performance in this case. We also considered the case of node failure, and showed the advantage of the Control-DQMC system in this case both in terms of performance and robustness.

An improvement of the control strategy is to be considered in the future work by integration a derivation action in the controller in addition to the current proportional and integral control, as well as, exploring the potential of control more advanced than PID, especially in the perspective of obtaining guarantees on the exploration safety. While presented and validated on service placement problems, we believe that this strategy can be broadly used to solve the exploration–exploitation dilemma in learning problems.

CRediT authorship contribution statement

Ghina Dandachi: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing, Visualization. **Sophie Cerf:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Visualization. **Yassine Hadjadj-Aoul:** Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing, Project administration. **Abdelkader Outtagarts:** Conceptualization. **Eric Rutten:** Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

References

[1] NGMN Alliance, Description of Network Slicing Concept, Tech. rep., Version 1.0, 2016.

[2] A. Rkhami, Y. Hadjadj-Aoul, A. Outtagarts, Learn to improve: A novel deep reinforcement learning approach for beyond 5G network slicing, in: 2021 IEEE 18th Annual Consumer Communications Networking Conference, CCNC, 2021, pp. 1–6, http://dx.doi.org/10.1109/CCNC49032.2021.9369463.

[3] P.T.A. Quang, Y. Hadjadj-Aoul, A. Outtagarts, A deep reinforcement learning approach for VNF forwarding graph embedding, IEEE Trans. Netw. Serv. Manag. 16 (4) (2019) 1318–1331, http://dx.doi.org/10.1109/TNSM.2019.2947905.

[4] Y. Carlinet, E. Gourdin, N. Perrot, The offline virtual network function packing problem, in: 2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops, ICIN, 2021, pp. 151–158, http://dx.doi.org/10.1109/ICIN51074.2021.9385554.

[5] T.A.Q. Pham, J.-M. Sanner, C. Morin, Y. Hadjadj-Aoul, Virtual network function–forwarding graph embedding: A genetic algorithm approach, Int. J. Commun. Syst. 33 (10) (2020) e4098, http://dx.doi.org/10.1002/dac.4098, e4098 0.1002/dac.4098 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/dac.4098.

[6] B. Recht, A tour of reinforcement learning: The view from continuous control, Ann. Rev. Control Robot. Auton. Syst. 2 (2019) 253–279.

[7] G.P.A.W. Group, View on 5G Architecture, Tech. rep., 5GPPP, 2020, http://dx.doi.org/10.5281/zenodo.3265031.

[8] G. Dandachi, A. Rkhami, Y. Hadjadj-Aoul, A. Outtagarts, A robust Monte-Carlo-based deep learning strategy for virtual network embedding, in: the 47th IEEE Conference on Local Computer Networks, LCN, Edmonton, Canada, 2022, URL https://hal.inria.fr/hal-03727967.

[9] S. Vassilaras, L. Gkatzikis, N. Liakopoulos, I.N. Stiakogiannakis, M. Qi, L. Shi, L. Liu, M. Debbah, G.S. Paschos, The algorithmic aspects of network slicing, IEEE Commun. Mag. 55 (8) (2017) 112–119, http://dx.doi.org/10.1109/MCOM.2017.1600939.

[10] A. Benhamiche, A.R. Mahjoub, N. Perrot, E. Uchoa, Capacitated multi-layer network design with unsplittable demands: Polyhedra and branch-and-cut, Discrete Optim. 35 (2020) 100555, http://dx.doi.org/10.1016/j.disopt.2019.100555, URL https://www.sciencedirect.com/science/article/pii/S1572528619301550.

[11] R. Mijumbi, J.-L. Gorricho, J. Serrat, M. Claeys, F. De Turck, S. Latré, Design and evaluation of learning algorithms for dynamic resource management in virtual networks, in: 2014 IEEE Network Operations and Management Symposium, NOMS, 2014, pp. 1–9, http://dx.doi.org/10.1109/NOMS.2014.6838258.

[12] S. Wang, J. Bi, J. Wu, A.V. Vasilakos, Q. Fan, VNE-TD: A virtual network embedding algorithm based on temporal-difference learning, Comput. Netw. 161 (2019) 251–263, http://dx.doi.org/10.1016/j.comnet.2019.05.004, URL http://www.sciencedirect.com/science/article/pii/S138912861830584X.

[13] Y. Kawamoto, H. Takagi, H. Nishiyama, N. Kato, Efficient resource allocation utilizing Q-learning in multiple UA communications, IEEE Trans. Netw. Sci. Eng. 6 (3) (2018) 293–302.

[14] H. Yao, X. Chen, M. Li, P. Zhang, L. Wang, A novel reinforcement learning algorithm for virtual network embedding, Neurocomputing 284 (2018) 1–9.

[15] S. Haeri, L. Trajković, Virtual network embedding via Monte Carlo tree search, IEEE Trans. Cybern. 48 (2) (2018) 510–521, http://dx.doi.org/10.1109/TCYB.2016.2645123.

[16] G. Sun, Z.T. Gebrekidan, G.O. Boateng, D. Ayepah-Mensah, W. Jiang, Dynamic reservation and deep reinforcement learning based autonomous resource slicing for virtualized radio access networks, IEEE Access 7 (2019) 45758–45772, http://dx.doi.org/10.1109/ACCESS.2019.2909670.

[17] G. Sun, K. Xiong, G.O. Boateng, D. Ayepah-Mensah, G. Liu, W. Jiang, Autonomous resource provisioning and resource customization for mixed traffics in virtualized radio access network, IEEE Syst. J. 13 (3) (2019) 2454–2465, http://dx.doi.org/10.1109/JSYST.2019.2918005.

[18] I. Bello, H. Pham, Q.V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning, 2016, arXiv:1611.09940.

[19] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.

[20] Z. Yan, J. Ge, Y. Wu, L. Li, T. Li, Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks, IEEE J. Sel. Areas Commun. 38 (6) (2020) 1040–1057, http://dx.doi.org/10.1109/JSAC.2020.2986662.

[21] S. Agrawal, N. Goyal, Analysis of thompson sampling for the multi-armed bandit problem, in: S. Mannor, N. Srebro, R.C. Williamson (Eds.), Proceedings of the 25th Annual Conference on Learning Theory, Vol. 23, in: Proceedings of Machine Learning Research, JMLR Workshop and Conference Proceedings, Edinburgh, Scotland, 2012, pp. 39.1–39.26, URL http://proceedings.mlr.press/v23/agrawal12.html.

[22] N. Vlassis, M. Ghavamzadeh, S. Mannor, P. Poupart, Bayesian reinforcement learning, in: M. Wiering, M. van Otterlo (Eds.), Reinforcement Learning: State-of-the-Art, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 359–386, http://dx.doi.org/10.1007/978-3-642-27645-3_11.

[23] B. O'Donoghue, Variational Bayesian reinforcement learning with regret bounds, 2018, arXiv:1807.09647.

[24] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, I. Mordatch, Plan online, learn offline: Efficient learning and exploration via model-based control, 2018, arXiv preprint arXiv:1811.01848.

[25] M. Tokic, Adaptive ϵ-greedy exploration in reinforcement learning based on value differences, in: Annual Conference on Artificial Intelligence, Springer, 2010, pp. 203–210.

[26] J.L. Hellerstein, Y. Diao, S. Parekh, D.M. Tilbury, Feedback control of computing systems, Wiley Online Library, 2004.

[27] K.J. Åström, T. Hägglund, PID Controllers: Theory, Design, and Tuning, Vol. 2, Instrument society of America Research Triangle Park, NC, 1995.

[28] The internet topology zoo, 2021, http://www.topology-zoo.org/dataset.html (Accessed: 25 June 2021).

[29] P. Erdős, A. Rényi, On the evolution of random graphs, Publ. Math. Inst. Hung. Acad. Sci. 5 (1) (1960) 17–60.

[30] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

Sophie Cerf is junior researcher at the INRIA and CRIStAL in Lille (France). Before that, she was a postdoc fellow on a joint project with Argonne National Labs (USA) and LIG (France). She received her Ph.D. degree in 2019 from the University of Grenoble Alps (GIPSA-lab) in collaboration with the LIRIS (Lyon). Her research interest are control theory and software systems self-adaptation.

Yassine Hadjadj-Aoul is currently working as full professor at the University of Rennes 1, France, since 2022, where he is also a member of the IRISA Laboratory and the INRIA project-team Ermine. He received a B.Sc. In computer engineering with high honours from Mohamed Boudiaf University, Oran, Algeria, in 1999. Dr. Hadjadj received his Master's and Ph.D. degrees in computer science from the University of Versailles, France, in 2002 and 2007, respectively. He was an assistant professor at the University of Versailles from 2005 to 2007. He was also a post-doctoral fellow at the University of Lille 1 and a research fellow, under the EUFP6 EIF Marie Curie Action, at the National University of Dublin (UCD). Then, he joined the University of Rennes 1 as an associate professor. His main research interests concern the fields of wireless networking, multimedia streaming architectures and protocols, congestion control protocols and QoS provisioning, and satellite and space communications.

Abdelkader Outtagarts IEEE senior member is a senior researcher and technical leader in NFV and SDN orchestration, machine learning and automation in Nokia Bell Labs. Abdelkader received the M.Sc. and Ph.D. degrees from the INSA de Lyon in France, in 1990 and 1994 respectively, on automation of energy systems. In 1999, he receives a M.Sc. on software engineering from Ecole de Technologie Supérieure of Montreal in Canada. His professional experience of over 20 years, on information systems and telecommunications, energy efficiency, software engineering, cloud computing, data mining, NFV micro services and SDN orchestration and automation, is mainly acquired in France and Canada in R&D teams (Alcatel-Lucent, Nextenso, Hydro-Quebec, UTILICASE and SCII Technology), in research laboratories in Lyon (INSA), Montreal (ETS, CRIM) and Nozay (Nokia Bell labs).

Ghina Dandachi received the M.S. degree in telecommunication system technologies and the M.E. degree in computer science and telecommunications in 2013 from the Lebanese University, Beirut, Lebanon. She received the Ph.D. degree jointly at Telecom SudParis, Paris, France, the University of Paris VI, Paris, and the Lebanese University in 2017. She has worked as a research engineer at CEA LETI between 2017 and 2018. Since 2019, she has worked as a research engineer at INRIA Rennes, France. Her research interests include artificial intelligence mechanisms for dimensioning and performance optimization in wireless networks, 5G slicing and resource orchestration and Deep Reinforcement Learning.

Eric Rutten (Ph.D. 90, Habil. 99 at U. Rennes, France) is head of the Ctrl-A team at INRIA in Grenoble, France. His research interests are in the field of reactive systems, applied to embedded systems and autonomic systems. His present activities are on model-based control of adaptive and reconfigurable computing systems, at the levels of hardware, operating system, middleware and software components. He is using discrete control techniques, integrated in the compilation of a reactive programming language.