Symbolic Reinforcement Learning for Safe RAN Control

Demonstration Track

Alexandros Nikou
Ericsson Research

Anusha Mujumdar
Ericsson Research

Marin Orlić
Ericsson Research

Aneta Vulgarakis Feljan
Ericsson Research

ABSTRACT

In this paper, we demonstrate a Symbolic Reinforcement Learning (SRL) architecture for safe control in Radio Access Network (RAN) applications. In our automated tool, a user can select a high-level safety specifications expressed in Linear Temporal Logic (LTL) to shield an RL agent running in a given cellular network with aim of optimizing network performance, as measured through certain Key Performance Indicators (KPIs). In the proposed architecture, network safety shielding is ensured through model-checking techniques over combined discrete system models (automata) that are abstracted through reinforcement learning. We demonstrate the user interface (UI) helping the user set intent specifications to the architecture and inspect the difference in allowed and blocked actions.

ACM Reference Format:
Alexandros Nikou, Anusha Mujumdar, Marin Orlić, and Aneta Vulgarakis Feljan. 2021. Symbolic Reinforcement Learning for Safe RAN Control: Demonstration Track. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 3 pages.

1 INTRODUCTION AND MOTIVATION

Due to the growing complexity of modern cellular networks, network optimization and control constitutes one of the main challenges. It is desirable by the Mobile Network Operators (MNOs) that the configuration is adjusted automatically in order to ensure acceptable Quality of Service (QoS) to each user connected to the network. In such application, the goal is to optimize a set of network KPIs such as *coverage*, *quality* and *capacity* and to guarantee that certain bounds of these KPIs are not violated (safety specifications). This optimization is performed by adjusting the vertical electrical tilt of each of the antennas of the given network, known in the literature as remote electrical tilt (RET) optimization problem [4, 5, 9, 14].

Reinforcement learning (RL) [3, 7, 11, 15] has become a powerful solution for dealing with the problem of optimal decision making for agents interacting with uncertain environments. However, it is known that the large-scale exploration performed by RL algorithms can sometimes take the system to unsafe states [7]. In the problem of RET optimization, RL has been proven to be an effective framework for KPI optimization due to its self-learning capabilities and adaptivity to potential environment changes [16]. For addressing the safety problem (i.e., to guarantee that the desired KPIs remain in specified bounds) authors in [16] have proposed a statistical approach to empirically evaluate the RET optimization in different baseline policies and in different worst-case scenarios.

The aforementeioned statistical approach focus on ensuring the reward value remains above a desired baseline. However, more widely accepted notions of safety are expressed in terms of safe states, defined according to a (formal) intent specification [6]. The approach in [6] decouples the notion of safety from that of reward. Intuitively, safety intents define the boundaries within which the RL agent may be free to explore. Motivated by the abovementioned, in this work, we demonstrate a novel approach for guaranteeing safety in RET optimization problem by using model-checking techniques and in parallel, we seek to generalize the problem in order to facilitate richer specifications than safety. In order to express desired specifications to the network into consideration, LTL is used (see [2, 10, 12, 13]), due to the fact that it provides a powerful mathematical formalism for such purpose. Our proposed demonstration exhibits the following attributes: (1) a general automatic framework from LTL specification user input to the derivation of the policy that fulfills it; at the same time, blocking the control actions that violate the specification; (2) novel system dynamics abstraction to companions Markov Decision Processes (MDP) which is computationally efficient; (3) UI development that allows the user to graphically access all the steps of the proposed approach.

Related work. Authors in [1] propose a safe RL approach through shielding. However, they assume that the system dynamics abstraction into an MDP is given, which is challenging in network applications that this demonstration refers to. As mentioned previously, authors in [16] address the safe RET optimization problem, but this approach relies on statistical guarantees and it cannot handle general LTL specifications that we treat with this manuscript.

2 DEMONSTRATION

Our key contribution is a proposed architecture which allows for intent specifications in RL, demonstrated in a real-world example. Here we focus on task specifications given in LTL. The syntax of LTL (see [2]) over a set of atomic propositions Σ is defined by the grammar $\varphi := \top \mid \varsigma \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 \,\mathfrak{U}\, \varphi_2 \mid \Diamond\varphi \mid \Box\varphi$, where $\varsigma \in \Sigma$ and $\bigcirc$, $\mathfrak{U}$, $\Diamond$, $\Box$ operators stand for the next, until, eventually and always operators, respectively; $\neg$ and $\wedge$ are the negation and conjunction operator respectively. Every LTL formula can be translated to a Büchi Automaton (BA) that models all the system traces satisfying the formula [8].

Consider a geographic area covered by Radio Base Stations (RBS) and cells that serve a set of UEs uniformly distributed in that area. The RET optimization problem has goal to maximize network capacity and coverage while minimizing interference between the antennas. The RET control strategy handles the antenna tilt of each of the cells (agents), and is executed independently for each cell. The environment of the RL agents is a simulated mobile network as it can be seen in Fig. 2. The system dynamical model is captured through an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where: $\mathcal{S}$ are the states consisting of values for down-tilt and KPIs (coverage, capacity and quality); actions $\mathcal{A} = \{\text{downtilt}, 0, \text{uptilt}\}$; transition probability matrix $\mathcal{P}$ which describes the state evolution given the current and the executed by the action state; rewards $\mathcal{R}$; and, discount factor γ. The

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

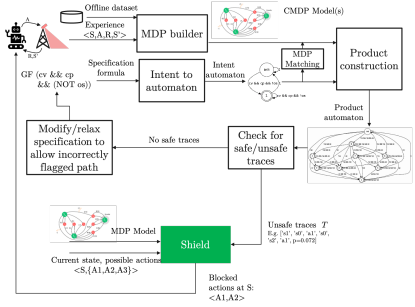

Figure 1: A graphical illustration of the proposed architecture.

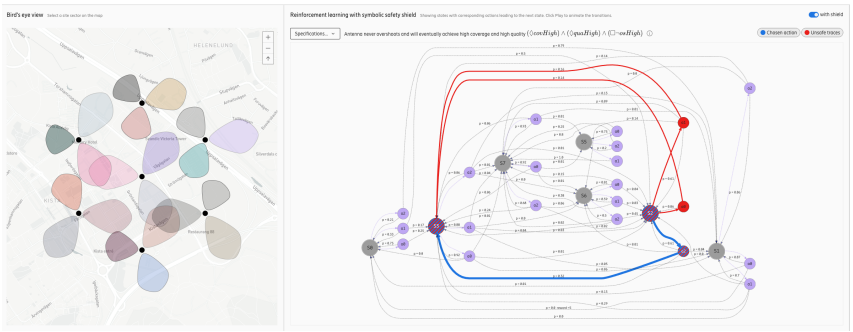

Figure 2: A graphical illustration of the UI for the demonstration of the approach. The user can chose: the desired LTL formula, the resulting BA; and the evolution of the RL agent training and the actions blocked by the safety shield.

RL agent's policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ is a function that maps the states to actions that define the agent's strategy.

Our solution takes a sequence of steps to match the LTL specification with the RL agent as it is depicted in Fig. 1 and block the actions that lead to unsafe states. Initially, the desired user specification is translated into LTL logic and subsequently into a BA. Then, by gathering the experience data tuples from the RL agent which is trained to a simulation environment with state-of-the-art model-free RL algorithms (DQN, Q-learning, SARSA [4, 5, 9, 14]) we construct the system dynamics modelled as an MDP. In this solution, we have a novel structure known as Companion MDPs (CMDPs). CMDPs encode the state transitions only in terms of the subset of features specified in the intent, not the full set of state features. This reduces the state space complexity, and keeps only the relevant features depending on the intent. An MDP matching component matches the intent to the relevant CMDP (depending on the features mentioned in the intent).

The experience data tuples that are generated during training are in the form (s, a, r, s') where s indicates the current state, a the executed action, r the received reward that the agent receives after applying action a at state s; and s' the state the agent transitions to after executing action a at state s. In order to match the BA from the given LTL specification and the MDP, the states of the MDP are labelled according to the atomic propositions from the LTL specification. Then, by computing the product of the MDP with the specification, we construct: an automaton $\mathcal{A}_\varphi$ that models all the system behaviours over the given specification; an automaton $\mathcal{A}_{\neg\varphi}$ that models all the traces violating the specification. Then, by utilizing graph techniques and model checkers, we are able to find all the system traces violating the specification (w.r.t the trained MDP); if no safe traces are found from all states in the MDP, the user specification cannot be satisfied, which means that the LTL has to be modified (or relaxed). If there exist some unsafe and some safe traces, then the process moves to a shield strategy that blocks the actions that lead to unsafe traces. This process is depicted more formally in Fig. 1 and Algorithm 1.

3 DISCUSSIONS

Interaction with the UI [1]. The UI is designed to be used by a network operations engineer who can specify safety intents, monitor tilts and their impact, and supervise the RL agent's operation. The initial screen of the UI depicts a geographic area with the available radio sites and cells. By clicking to one of the cells, a new screen appears with the KPI values depicted on the left. On the right part of the page, one can see: 1) the MDP system model; 2) a list of available LTL intents; 3) BAs representing each of the intents; 4) the button "Run safe RL" to run the simulation; and 5) the switch "with/without shield" for enabling the safety shield. The chosen actions on the MDP are depicted in blue, while the blocked actions by the shield are depicted in red. The user can view the training process and the optimal choice of actions that guarantee the satisfaction of given input as well as the block of unsafe actions. The current high level of detail in the UI is meant to illustrate the technology, it can be imagined that a production UI would instead show a summary of selected and blocked actions instead of large MDP models. The impact of the shield may also be viewed, and it is seen that the shield blocks a proportion of unsafe states (leading to 639 unsafe states instead of 994 without the shield). Interestingly, the shield also leads to a 68% improvement in the reward values.

Algorithm 1

Input: User specification Φ
1: **Gather** experience replay (s, a, r, s') from data;
2: **Discretize** states into N_b. State space size is $|\mathcal{S}|^{N_b}$;
3: **Construct** the MDP dynamics $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$;
4: **Translate** the LTL formula Φ to a BA $\mathcal{A}_\varphi$;
5: **Compute** the product $\mathcal{T} = MDP \otimes \mathcal{A}_\varphi$ and pass it to model checker;
6: **Model checking** returns traces that violate φ;
7: **If** no safe traces found **Modify/Relax** φ
8: **Else** Block unsafe actions by function *Shield*(MDP, $\mathcal{T}$).

Applicability to other domains. The proposed architecture is general and it can be applied to any framework in which the under consideration dynamical system is abstracted into an MDP, while LTL specifications need to be fulfilled. For example, in a robot planning applications, the states are locations of the environment that the robot can move, and atomic propositions are the goal state and the obstacles. The LTL formula of such application would include reachability and safety tasks.

Conclusions and future work. In this paper, we have demonstrated an architecture for network KPIs optimization guided by user-defined intent specifications given in LTL. Our solution consists of MDP system dynamics abstraction, automata construction and products and model-checking techniques to block undesired actions that violate the specification. Future research directions will be devoted towards applying the proposed framework in other telecom use cases as well as in robotics (motion planning).

[1] The video accompanying this paper can be found in: https://www.ericsson.com/en/reports-and-papers/research-papers/safe-ran-control.

The authors thank Ezeddin Al Hakim, Jaeseong Jeong, Maxime Bouton, Swarup Mohalik, Pooja Kashyap and Athanasios Karapantelakis for the fruitful discussion in topics related to this work and support in the simulation environment. Moreover, special thanks to Ericsson Research for supporting and funding our work.

REFERENCES

[1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe reinforcement learning via shielding. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (2018).

[2] Christofer Baier, Joost Pieter Katoen, and Kim G. Larsen. April 2008. Principles of Model Checking. *MIT Press* (April 2008).

[3] Maxime Bouton, Jana Tumova, and Mykel J Kochenderfer. 2020. Point-Based Methods for Model Checking in Partially Observable Markov Decision Processes. *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), 10061–10068.

[4] Víctor Buenestado, Matías Toril, Salvador Luna-Ramírez, José María Ruiz-Avilés, and Adriano Mendo. 2016. Self-tuning of remote electrical tilts based on call traces for coverage and capacity optimization in LTE. *IEEE Transactions on Vehicular Technology* 66, 5 (2016), 4315–4326.

[5] Shaoshuai Fan, Hui Tian, and Cigdem Sengul. 2014. Self-optimization of coverage and capacity based on a fuzzy neural network with cooperative reinforcement learning. *EURASIP Journal on Wireless Communications and Networking* 2014, 1 (2014), 57.

[6] Nathan Fulton and André Platzer. 2018. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. 32, 1 (2018).

[7] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

[8] Paul Gastin and Denis Oddoux. 2001. Fast LTL to Büchi automata translation. *International Conference on Computer Aided Verification* (2001), 53–65.

[9] Weisi Guo, Siyi Wang, Yue Wu, Jonathan Rigelsford, Xiaoli Chu, and Tim O'Farrell. 2013. Spectral-and energy-efficient antenna tilting in a HetNet using reinforcement learning. *2013 IEEE Wireless Communications and Networking Conference (WCNC)* (2013), 767–772.

[10] Savas Loizou and Kostas J. Kyriakopoulos. Los Angeles, California, USA, December 2004. Automatic Synthesis of Multi-Agent Motion Tasks Based on LTL Specifications. *43rd IEEE Conference on Decision and Control (CDC)* 1 (Los Angeles, California, USA, December 2004), 153–158.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[12] Alexandros Nikou. 2019. Robust and Decentralized Control of Multi-agent Systems under High-level Tasks. *Doctoral Thesis, KTH Royal Institute of Technology, Link: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-263712* (2019).

[13] Alexandros Nikou, Dimitris Boskos, Jana Tumova, and Dimos V. Dimarogonas. November 2018. On the Timed Temporal Logic Planning of Coupled Multi-Agent Systems. *Automatica* 97 (November 2018), 339–345.

[14] Rouzbeh Razavi, Siegfried Klein, and Holger Claussen. 2010. A fuzzy reinforcement learning approach for self-optimization of coverage in LTE networks. *Bell Labs Technical Journal* 15, 3 (2010), 153–175.

[15] Richard S Sutton and Andrew G Barto. 2018. Reinforcement learning: An introduction. (2018).

[16] Filippo Vannella, Jaeseong Jeong, and Alexandre Proutiere. 2020. Off-policy Learning for Remote Electrical Tilt Optimization. *arXiv preprint, arXiv:2005.10577* (2020).