mobile-env: An Open Platform for Reinforcement Learning in Wireless Mobile Networks

Stefan Schneider, Stefan Werner
Paderborn University, Germany
{stschn, stwerner}@mail.upb.de

Ramin Khalili, Artur Hecker
Huawei Technologies, Germany
{ramin.khalili, artur.hecker}@huawei.com

Holger Karl
Hasso Plattner Institute,
University of Potsdam, Germany
holger.karl@hpi.de

***Abstract*—Recent reinforcement learning approaches for continuous control in wireless mobile networks have shown impressive results. But due to the lack of open and compatible simulators, authors typically create their own simulation environments for training and evaluation. This is cumbersome and time-consuming for authors and limits reproducibility and comparability, ultimately impeding progress in the field.**

To this end, we propose *mobile-env*, a simple and open platform for training, evaluating, and comparing reinforcement learning and conventional approaches for continuous control in mobile wireless networks. *mobile-env* is lightweight and implements the common OpenAI Gym interface and additional wrappers, which allows connecting virtually any single-agent or multi-agent reinforcement learning framework to the environment. While *mobile-env* provides sensible default values and can be used out of the box, it also has many configuration options and is easy to extend. We therefore believe *mobile-env* to be a valuable platform for driving meaningful progress in autonomous coordination of wireless mobile networks.

***Index Terms*—wireless mobile networks, network management, continuous control, autonomous coordination, reinforcement learning, gym environment, simulation, open source**

I. Introduction

There is an ongoing trend towards self-learning and self-adapting approaches for continuous control and coordination in networking [1]. Examples are self-learning approaches for management in wired networks [2] but also for cell selection or power control in wireless mobile networks [3], [5]. In contrast to conventional approaches (e.g., heuristics or mixed-integer linear programs), such self-learning approaches use reinforcement learning or contextual bandits, trained on simulation environments, to adapt to various scenarios autonomously without requiring expert or a priori knowledge [1].

While there are existing network simulators (Sec. III), they are often not directly compatible with typical reinforcement learning frameworks requiring the OpenAI Gym interface [6]. They are also rather cumbersome to set up and configure, preventing fast prototyping of new self-learning approaches. Therefore, many authors currently resort to creating their own environments when designing and publishing a new reinforcement learning approach. Creating a suitable simulation environment is time-consuming and error-prone, slowing down authors in their research. It is also more difficult for others to interpret and compare published approaches if they each rely on different simulation environments. Often, the corresponding environment is not publicly available, making reproduction and reuse of the published approach impossible. These limitations severely hinder comparability and meaningful progress in the field of autonomous control for wireless mobile networks.

To mitigate these problems, we propose *mobile-env*, which is a simple and open platform for training, evaluating, and comparing coordination approaches, particularly suitable for wireless mobile networks. The *mobile-env* platform is lightweight and fast, simple to install and use, written purely in Python, and compatible to existing reinforcement learning frameworks by complying with the OpenAI Gym interface. Besides novel self-learning approaches, *mobile-env* also supports existing conventional approaches, facilitating their evaluation and comparison. To encourage adoption by the community, *mobile-env* is open source [7], available on the official Python package index (PyPI), comes with sensible default values yet extensive configuration options, and is well documented. Thanks to its modular architecture, it is also easy to adjust and extend (Sec. II). With *mobile-env*, we hope to provide a useful tool for prototyping, training, and evaluating new approaches as well as for benchmarking and comparing existing approaches.

II. The *mobile-env* Platform

mobile-env is a minimalist and lightweight simulation platform that can serve as training and evaluation environment for reinforcement learning and other control algorithms in wireless mobile networks. Sec. II-A outlines our design goals. Sec. II-B describes the simulation procedure and interaction with an agent through the OpenAI Gym interface, pointing out common challenges and our approaches to address them. Sec. II-C presents the resulting architecture.

A. Design Goals

Current simulation environments are typically either tailored to a specific use case [8], [9] or versatile, generic network simulation frameworks [10], [11] (detailed in Sec. III). Custom-tailored simulation environments are ideal to quickly prototype new approaches in a specific use case but are difficult to adjust and extend to other scenarios. Generic network simulators provide a wealth of features and configuration options, allowing fine-grained simulation of many different scenarios but also requiring significant overhead for modeling each new scenario and additional effort for integrating self-learning approaches.

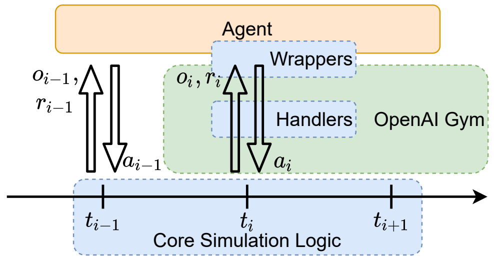

Fig. 1: *mobile-env* simulation procedure.

With *mobile-env* we aim to fill the gap between these two sides of the spectrum. Our goal is to keep *mobile-env* simple and lightweight with sensible default values and predefined example scenarios, enabling easy adoption and fast prototyping of new approaches. At the same time, we aim at configurability and extensibility, allowing users to customize all aspects of the simulation environment with low overhead to experiment with different scenarios and address their specific use cases. Core to this configurability and extensibility are the many provided configuration options and the modular architecture (Sec. II-C).

B. Simulation Procedure

When training and evaluating self-learning approaches, the control agent repeatedly interacts with the simulation environment, typically through the standardized OpenAI Gym interface [6]. Fig. 1 illustrates this simulation procedure, where the agent obtains an observation o_{i-1} of the environment's current state at time t_{i-1} and selects an action a_{i-1} to control the environment (e.g., cell selection, power control, scheduling, ...). This action is applied to the environment, affects its state, and leads to a new observation o_i and a reward r_i at time t_i.

Authors wanting to devise and prototype new self-learning approaches often face multiple challenges: How do they quickly set up the environment's simulation logic? How do they integrate the learning agent with the simulation loop? How do they customize the environment to experiment with different scenarios and with varying observations, actions, and reward, which are crucial to reinforcement learning? How do they deal with multi-agent scenarios and with different available reinforcement learning frameworks?

mobile-env addresses these challenges in different ways: The core simulation logic is inside the environment's `step` function (as defined by OpenAI Gym), which takes an action, applies it to the environment, progresses simulation time, and returns the next observation and reward. We call `step` periodically rather than event-based to let the agent decide itself for each time step whether to take an action or to choose a no-op. An exemplary use case of *mobile-env* is multi-cell selection for scenarios with coordinated multipoint (CoMP), where users can connect to more than one serving cell simultaneously for higher data rates or improved coverage. Here, the action defines which cells to (dis-)connect, and the `step` function applies these actions and updates users' connections, positions, data rates, etc. To this end, it leverages realistic realistic models for path loss (Okumura-Hata [12]), user mobility (improved random waypoint [13]), scheduling (e.g., proportional-fair), etc., which are included in *mobile-env* and can be replaced, customized, and extended freely. For other use cases, e.g., power control or scheduling, these models can be replaced by the agent's actions. To facilitate quick prototyping, *mobile-env* comes with predefined example scenarios and all configuration options have sensible defaults.

In addition to providing the core simulation logic, *mobile-env* also introduces "handlers", which allow customizing observations, actions, and reward easily and "wrappers" to align the environment to interfaces other than OpenAI Gym, e.g., for multi-agent learning (detailed in Sec. II-C).

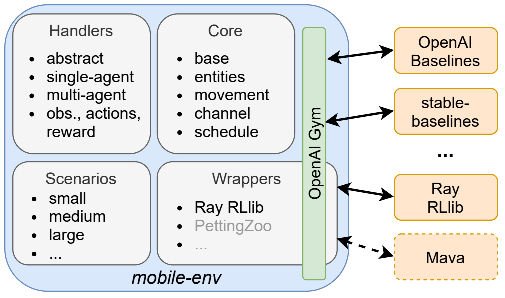

Fig. 2: *mobile-env* modular architecture.

C. Modular Architecture

We built *mobile-env* in a modular, object-oriented fashion. Fig. 2 illustrates the architecture consisting of four main packages: core, scenarios, handlers, and wrappers. The *core* package contains all main functionality and simulation logic, e.g., modeling properties of users and base stations (entities) or implementing different algorithms for scheduling resources to connected users (scheduling). It also implements rendering functionality to visualize learned policies, e.g., when running inside a Jupyter notebook. Each of these core modules contains classes that can be extended easily through inheritance and by overwriting the desired attributes or methods. The *scenarios* package contains predefined configurations, currently for a small, medium, and large scenario, i.e., for cell selection with different numbers of users and cells.. Over time, we plan to include more and more predefined scenarios, depending on feedback and contributions by the community. Each scenario is versioned and registered as Gym environment such that it can be used out of the box after importing *mobile-env* (e.g., `gym.make("mobile-medium-central-v0")`).

As described in Sec. II-B, *mobile-env* implements the common OpenAI Gym interface with its `step` function [6]. Rather than hard-wiring specific observations, actions, and rewards inside `step`, they are configurable through *handlers*. A handler class has full access to the environment state, defines the observation and action space, and is called by `step` to process actions, calculate rewards, and return new observations. *mobile-env* already provides example handlers for both single-agent and multi-agent approaches, which can be sub-classed, adjusted, and extended.

Some reinforcement learning frameworks deviate from the standard OpenAI Gym interface, e.g., for multi-agent approaches. Here, *wrapper* classes wrap the Gym interface and transform observations, actions, or reward into the desired format but, unlike handlers, do not change their meaning. We provide a wrapper for multi-agent learning with the popular Ray RLlib framework [14], which, for example, expects the observation space to be defined per agent, not combined for all agents. Through the OpenAI Gym interface and these wrappers, researchers can connect their approaches based on virtually any framework to *mobile-env* and leverage it for fast and simple training and evaluation.

III. Comparison Against Other Platforms

mobile-env is inspired by `highway-env` [9], which is also a simple and lightweight environment for reinforcement learning but considers autonomous driving instead of wireless mobile networks. Brunori et al. [8] propose a related platform for multi-agent reinforcement learning in wireless networks but focuses on autonomous unmanned aerial vehicles (UAVs). Besides, their platform has little documentation and examples and is limited by an overly restrictive, custom license, denying open-source use, modification, or redistribution. In contrast, *mobile-env* is more flexible, better documented, and licensed by the standard, permissive MIT license, allowing private and commercial use, modification, and distribution.

On the other side of the spectrum, there are established, generic network simulators like ns-3 [10] or OMNeT++ [11], which are very flexible and model networking in detail. These simulators provide far more features than *mobile-env*, but their flexibility comes at the cost of increased complexity and overhead when testing many different scenarios and approaches. *mobile-env* can be installed quickly with a single command (`pip install mobile-env`; 25 kB) and can train or evaluate a reinforcement learning agent in less than 10 lines of Python code. In contrast, OMNeT++ requires defining the structure of the desired simulation environment in NED language within a custom IDE, then implementing its logic in various C++ classes, then parametrizing it in a separate `omnetpp.ini` file, and finally building and compiling it, before the simulation is ready to run. At this point, the simulation still lacks the interaction with the reinforcement learning agent through the OpenAI Gym interface (Python)—a considerable, often prohibitive overhead for researchers. While there is an OpenAI Gym implementation for ns-3 [15], the complexity and overhead are still comparable to OMNeT++.

In comparison, *mobile-env* represents an intermediate solution with reasonable features and flexibility, built-in single- and multi-agent support via OpenAI Gym and wrappers, as well as fast and simple installation and usage.

IV. Demonstration

We demonstrate *mobile-env*, we provide an interactive Jupyter notebook, which others can experiment with at their own pace.[1] In this demonstration, we show how to set up *mobile-env*, illustrate the predefined default scenarios, and options for configuration and extension (e.g., regarding the number and movement of simulated users and the observation or action space). We also demonstrate how to connect different reinforcement learning frameworks to *mobile-env* for training and visualizing single-agent and multi-agent approaches.

V. Conclusion

With *mobile-env* we propose an open and simple yet configurable platform for reinforcement learning in wireless mobile networks [7]. We look forward to others using and extending *mobile-env* for their research and welcome their contributions!

Acknowledgments

This work has received funding from the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (SFB 901) and the open6GLab (German BMBF funding; grant 16KISK011).

References

[1] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[2] S. Schneider, A. Manzoor, H. Qarawlus, R. Schellenberg, H. Karl, R. Khalili, and A. Hecker, "Self-driving network and service coordination using deep reinforcement learning," in *International Conference on Network and Service Management (CNSM)*. IEEE, 2020, pp. 1–9.

[3] W. Liu, L. Wang, E. Wang, Y. Yang, D. Zeghlache, and D. Zhang, "Reinforcement learning-based cell selection in sparse mobile crowdsensing," *Computer Networks*, vol. 161, pp. 102–114, 2019.

[4] S. Vimal, M. Khari, N. Dey, R. G. Crespo, and Y. H. Robinson, "Enhanced resource allocation in mobile edge computing using reinforcement learning based MOACO algorithm for IIOT," *Computer Communications*, vol. 151, pp. 355–364, 2020.

[5] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, 2019.

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[7] S. Schneider and S. Werner, "*mobile-env* GitHub repository," https://github.com/stefanbschneider/mobile-env, 2021.

[8] D. Brunori, S. Colonnese, F. Cuomo, and L. Iocchi, "A reinforcement learning environment for multi-service UAV-enabled wireless systems," in *IEEE PerCom Workshops*. IEEE, 2021, pp. 251–256.

[9] Leurent, "highway-env," https://github.com/eleurent/highway-env, 2018.

[10] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM Demonstration*, vol. 14, no. 14, p. 527, 2008.

[11] OpenSim, "OMNeT++ website," https://omnetpp.org/, 2021.

[12] A. Medeisis and A. Kajackas, "On the use of the universal okumura-hata propagation prediction model in rural areas," in *IEEE Vehicular Technology Conference (VTC)*, vol. 3. IEEE, 2000, pp. 1815–1818.

[13] J. Yoon, M. Liu, and B. Noble, "Random waypoint considered harmful," in *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2003, pp. 1312–1321.

[14] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 3053–3062.

[15] P. Gawłowicz and A. Zubow, "ns-3 meets OpenAI Gym: The playground for machine learning in networking research," in *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. ACM, 2019.

[1] Interactive Jupyter notebook: https://bit.ly/3FDHKEk (Jan. 18, 2022)