

UNIVERSIDADE FEDERAL DO PARÁ  
INSTITUTO DE TECNOLOGIA  
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

**Hybrid CAVIAR Simulations and Reinforcement Learning Applied to 5G Systems:  
Experiments with Scheduling and Beam Selection**

**João Paulo Tavares Borges**

**DM: 04/2022**

UFPA / ITEC / PPGEE  
Campus Universitário do Guamá  
Belém-Pará-Brasil

2022



UNIVERSIDADE FEDERAL DO PARÁ  
INSTITUTO DE TECNOLOGIA  
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

**João Paulo Tavares Borges**

**Hybrid CAVIAR Simulations and Reinforcement Learning Applied to 5G Systems:  
Experiments with Scheduling and Beam Selection**

**DM: 04/2022**

UFPA / ITEC / PPGEE  
Campus Universitário do Guamá  
Belém-Pará-Brasil  
2022

UNIVERSIDADE FEDERAL DO PARÁ  
INSTITUTO DE TECNOLOGIA  
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

**João Paulo Tavares Borges**

**Hybrid CAVIAR Simulations and Reinforcement Learning Applied to 5G Systems:  
Experiments with Scheduling and Beam Selection**

A dissertation submitted to the examination committee in the graduate department of Electrical Engineering at the Federal University of Pará in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with emphasis in Computational Intelligence.

UFPA / ITEC / PPGEE  
Campus Universitário do Guamá  
Belém-Pará-Brasil

2022

**Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD  
Sistema de Bibliotecas da Universidade Federal do Pará  
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)**

---

B732h Borges, João Paulo Tavares.  
Hybrid CAVIAR Simulations and Reinforcement Learning  
Applied to 5G Systems: Experiments with Scheduling and Beam  
Selection / João Paulo Tavares Borges. — 2022.  
73 f. : il. color.

Orientador(a): Prof. Dr. Aldebaro Barreto da Rocha Klautau  
Júnior  
Dissertação (Mestrado) - Universidade Federal do Pará,  
Instituto de Tecnologia, Programa de Pós-Graduação em  
Engenharia Elétrica, Belém, 2022.

1. Aprendizado por reforço. 2. 5G. 3. Simuladores  
híbridos. 4. MIMO. 5. Alocação de recursos. I. Título.

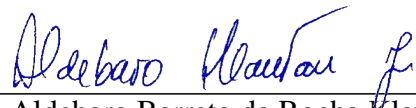
CDD 621.3

---

**Hybrid CAVIAR Simulations and Reinforcement Learning Applied to 5G Systems:  
Experiments with Scheduling and Beam Selection**

A dissertation submitted to the examination committee in the graduate department of Electrical Engineering at the Federal University of Pará in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering with emphasis in Computational Intelligence.

Approved in 28 / 01 / 2022



---

Prof. Dr. Aldebaro Barreto da Rocha Klautau Júnior

ADVISOR



---

Prof. Dr. Carlos Renato Lisboa Francês

MEMBER OF THE EXAMINATION COMMITTEE



---

Prof. Dr. Ilan Sousa Correa

MEMBER OF THE EXAMINATION COMMITTEE

---

Prof. Dr. Carlos Tavares da Costa Júnior

DIRECTOR OF THE GRADUATE  
DEPARTMENT OF ELECTRICAL ENGINEERING

# Acknowledgments

First, I would like to express my gratitude to God for giving me the opportunity to progress my professional career, and to work alongside several talented and helpful colleagues throughout the past few years at LASSE, with special thanks to Aldebaro Klautau, Ailton Oliveira, Cleverson Nahum, Felipe Bastos, Pedro Batista, Luan Gonçalves, Daniel Suzuki, Emerson Oliveira, Ingrid Ariel and Lucas Matni for their valuable help in this work. Many thanks to my family and friends for all their continuous support along the years.

I also give my thanks to the sponsors of this work, which was supported in part by the Innovation Center, Ericsson Telecomunicações S.A., Brazil, CNPq and the Capes Foundation. Finally, some of the CAVIAR concepts and motivation, were initially presented by the author in the “Connecting physical and virtual worlds” ITU Kaleidoscope 2021 academic conference from which I am also thankful.

João Paulo Tavares Borges

January 2022

*If I have seen further it is by standing  
on the shoulders of giants.*

*Isaac Newton*



# List of Acronyms

**5G** 5th Generation

**A2C** Advantage Actor Critic

**6G** 6th Generation

**RL** Reinforcement Learning

**DRL** Deep Reinforcement Learning

**UAV** Unmanned Aerial Vehicle

**BS** Base Station

**AI** Artificial Intelligence

**API** Application Programming Interface

**PHY** Physical Layer

**LoS** Line-of-Sight

**ULA** Uniform Linear Array

**CAVIAR** Communication Networks, Artificial Intelligence and Computer Vision with 3D  
Computer-Generated Imagery

**UPA** Uniform Planar Array

**3GPP** 3rd Generation Partnership Project

**DFT** Discrete Fourier Transform

**DL** Deep Learning

**GPS** Global Positioning System

**ITU** International Telecommunication Union

**KPI** Key Performance Indicator

**MDP** Markov Decision Process

**MIMO** Multiple-Input Multiple-Output

**ML** Machine Learning

**mmWave** millimeter wave

**MPC** Multipath Component

**NR** New Radio

**OFDM** Orthogonal Frequency-Division Multiplexing

**PHY** Physical Layer

**QoS** Quality of Service

**RT** Ray Tracing

**UE** User Equipment

**V2X** Vehicle-to-Everything

**CSV** Comma-Separated Values

**LiDAR** Light Detection and Ranging

**ViWi** Vision-Wireless

**NYU** New York University

**LIM** Light Intensity Model

**SITL** Software-in-the-Loop

**HITL** Hardware-in-the-Loop

**HLA** High Level Architecture

**OpenGL** Open Graphics Library

**PPO** Proximal Policy Optimization

**TD** Temporal Difference

**TRPO** Trust Region Policy Optimization

**ReLU** Rectified Linear Unit

**GPU** Graphics Processing Unit

**VISTA** Virtual Image Synthesis and Transformation for Autonomy

**CARLA** Car Learning to Act

**KL** Kullback–Leibler

# List of Figures

1.1	CAVIAR simulation scenario, depicting the radiation pattern (in light green) corresponding to the chosen beamforming codebook index to serve a drone (at the right). . . . .	3
2.1	The agent-environment dynamic in reinforcement learning. . . . .	7
2.2	A given sequence of states and rewards, composing an episode. . . . .	10
2.3	Monte Carlo estimation of the state-values. . . . .	11
2.4	Actor-critic estimation of the state-values. . . . .	11
3.1	Definition of digital twins sub-categories in terms of their level of integration with their physical counterparts . . . . .	17
4.1	Representation of a possible CAVIAR simulation, integrating the three subsystems. . . . .	19
4.2	Out-loop CAVIAR data generation. . . . .	21
4.3	Block diagram explaining how hardware-in-the-loop includes the hardware of the actual drone inside the simulation loop for increased realism. . . . .	24
5.1	Out-loop CAVIAR simulation flow used in the experiments. . . . .	27
5.2	Example of radiation pattern for specific beam index with a $8 \times 8$ UPA. . . . .	28
5.3	Distribution of the channel throughput when using always the best beam index $\hat{i}$ and a simple scheduling strategy, that chooses users sequentially in a round-robin fashion (uav->car->pedestrian->uav->car->pedestrian...), over 200 episodes. . . . .	32
6.1	Performance of the agents and baseline approaches over 25 test episodes. . . . .	34
6.2	Cumulative reward of the agents and baseline approaches over 25 test episodes. . . . .	35

6.3	Cumulative reward of the scheduling agent and baseline approaches over 25 test episodes. . . . .	36
7.1	Overall messages needed by the proposed digital twin solution, together with the main processing steps. . . . .	39
A.1	Example of the proposed MIMO beam selection environment. . . . .	52
A.2	Performance of a PPO agent in the task of beam selection inside the Multiple-Input Multiple-Output (MIMO) MiniGrid environment. . . . .	53
A.3	Performance of a PPO agent in a slightly more complex task, with the introduction of one additional user to the dynamic of the MIMO MiniGrid environment. . . . .	54

# List of Tables

4.1	Content of an episode file generated for the out-loop CAVIAR mode . . . . .	22
4.2	Content of an output file generated from a CAVIAR simulation . . . . .	23
5.1	Network load information for light and heavy scenarios . . . . .	30
A.1	Comparison of the methods performance on 1000 steps on a scenario with one user . . . . .	54
A.2	Comparison of the methods performance on 1000 steps on a scenario with two users . . . . .	55

# Contents

<b>Acknowledgment</b>	<b>vi</b>
<b>List of Acronyms</b>	<b>viii</b>
<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xiii</b>
<b>Contents</b>	<b>xiv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Context . . . . .	1
1.2 Proposal . . . . .	3
1.3 Dissertation Outline . . . . .	4
<b>2 Background Concepts</b>	<b>6</b>
2.1 MIMO Beam Selection . . . . .	6
2.2 Reinforcement Learning . . . . .	7
2.2.1 Objective of Reinforcement Learning Methods . . . . .	8
2.2.2 State-Value Estimation Approaches . . . . .	9
<b>3 Related Works</b>	<b>13</b>
3.1 5G MIMO Data Generation . . . . .	13
3.2 Hybrid Simulators for Communication Networks . . . . .	15
3.3 Improving Simulations Realism to Close Reality Gap . . . . .	15
3.4 Digital Twins . . . . .	16

<b>4</b>	<b>CAVIAR Simulations</b>	<b>19</b>
4.1	Overall CAVIAR Description . . . . .	20
4.2	Out-loop and In-loop Simulation Modes . . . . .	21
4.3	Main Software Tools . . . . .	23
4.3.1	Unreal Engine . . . . .	23
4.3.2	AirSim . . . . .	24
<b>5</b>	<b>Experimental Evaluation</b>	<b>26</b>
5.1	CAVIAR Simulation for User Scheduling and Beam Selection Problems . . . . .	26
5.2	Communication Model . . . . .	28
5.3	Traffic Model . . . . .	29
5.4	Possible Inputs to RL Agents . . . . .	30
5.5	Evaluation of the RL agent . . . . .	31
5.6	Characterizing the Communication Scenario with Baseline Approaches . . . . .	32
<b>6</b>	<b>Results and Discussion</b>	<b>33</b>
6.1	Results . . . . .	33
6.2	Discussion . . . . .	35
<b>7</b>	<b>Conclusion and Future Works</b>	<b>37</b>
7.1	Improvements on the Subsystems . . . . .	38
7.2	Using the Realistic Simulation Tools Described in this Work to Develop Simulations that Explore Concepts Related to Digital Twins . . . . .	39
	<b>Bibliography</b>	<b>42</b>
	<b>Appendices</b>	<b>50</b>
<b>A</b>	<b>Toy Environment for MIMO Beam Selection</b>	<b>51</b>
A.1	MiniGrid Environment . . . . .	51
A.2	MIMO Beam Selection MiniGrid Environment . . . . .	52
A.3	MIMO MiniGrid environment results . . . . .	53



## Abstract

Reinforcement Learning (RL) is a learning paradigm suitable for problems in which an agent has to maximize a given reward, while interacting with an ever-changing environment. This class of problem appears in several research topics of the 5th Generation (5G) and the 6th Generation (6G) of mobile networks. However, the lack of freely available data sets or environments to train and assess RL agents are a practical obstacle that delays the widespread adoption of RL in 5G and future networks. These environments must be able to close the so-called *reality gap*, where reinforcement learning agents, trained in virtual environments, are able to generalize their decisions when exposed to real, never before seen, situations. Therefore, this work describes a simulation methodology named CAVIAR, or *Communication Networks, Artificial Intelligence and Computer Vision with 3D Computer-Generated Imagery*, tailored for research on RL methods applied to the physical layer (PHY) of the wireless communications systems. In this work, this simulation methodology is used to generate an environment for the tasks of user scheduling and beam selection, where, at each time step, the RL agent needs to schedule a user and then choose the index of a fixed beamforming codebook to serve it. A key aspect of this proposal is that the simulation of the communication system and the artificial intelligence engine must be closely integrated, such that actions taken by the agent can reflect back on the simulation loop. This aspect makes the trade-off of processing time versus realism of the simulation, an element to be considered. This work also describes the modeling of the communication systems and RL agents used for experimentation, and presents statistics concerning the environment dynamics, such as data traffic, as well as results for baseline systems. Finally, it is discussed how the methods described in this work can be leveraged in the context of the development of digital twins.

**Keywords** — Reinforcement learning, 5G, hybrid simulators, MIMO, resource allocation

## Resumo

Aprendizado por reforço, do inglês Reinforcement Learning (RL), é um paradigma de aprendizagem adequado para problemas em que um agente tem que maximizar uma determinada recompensa, enquanto interage com um ambiente em constante mudança. Esta classe de problema aparece em diversos tópicos de pesquisa da 5ª Geração (5G) e da 6ª Geração (6G) das redes móveis. No entanto, a falta de conjuntos de dados ou ambientes disponíveis gratuitamente para treinar e avaliar os agentes de RL é um obstáculo prático que atrasa a adoção de RL em redes 5G e futuras. Esses ambientes devem ser capazes de fechar o chamado *reality gap*, onde os agentes de aprendizagem por reforço, treinados em ambientes virtuais, são capazes de generalizar suas decisões quando expostos a situações reais, nunca antes vistas. Portanto, este trabalho descreve uma metodologia de simulação denominada CAVIAR, ou *Communication Networks, Artificial Intelligence and Computer Vision with 3D Computer-Generated Imagery*, voltada para pesquisa sobre métodos de RL aplicados à camada física (PHY) dos sistemas de comunicações sem fio. Neste trabalho, essa metodologia de simulação é utilizada para gerar um ambiente para as tarefas de escalonamento de usuários e seleção de feixes, onde, a cada passo, o agente RL precisa escalonar um usuário e então escolher o índice de um codebook de beamforming para atendê-lo. Um aspecto fundamental desta proposta é que a simulação do sistema de comunicação e o software de inteligência artificial devem estar intimamente integrados, de modo que as ações realizadas pelo agente possam refletir de volta no loop de simulação. Esse aspecto torna a compensação de tempo de processamento versus realismo da simulação, um elemento a ser considerado. Este trabalho também descreve a modelagem dos sistemas de comunicação e agentes RL usados para experimentação, e apresenta estatísticas sobre a dinâmica do ambiente, como tráfego de dados, bem como resultados para sistemas baseline. Por fim, é discutido como os métodos descritos neste trabalho podem ser aproveitados no contexto do desenvolvimento de gêmeos digitais.

**Palavras-chave** — Aprendizado por reforço, 5G, simuladores híbridos, MIMO, alocação de recursos

# Chapter 1

## Introduction

### 1.1 Context

Reinforcement Learning (RL) is a learning paradigm suitable for problems in which an agent has to learn an optimal behavior that is previously unknown to the system, by maximizing a given reward via interactions with an environment prone to suffer changes as the time passes. This class of *learning through interaction* problem represented by RL, appears in several points of interest inside the 5th Generation (5G) and the 6th Generation (6G) of mobile networks, such as in: congestion control [1], network slicing [2], resource allocation [3], and the 5G Physical Layer (PHY) [4], to name a few.

To enable the use of Artificial Intelligence (AI) to address those challenging tasks, one could leverage the use of simulators in order generate labeled data to be used inside an Machine Learning (ML) pipeline, as it is common when dealing with supervised learning methods. In this regard, some works explore hybrid simulators to perform that task. These software integrate several other simulators to enable research on multidisciplinary areas, for example, in Vehicle-to-Everything (V2X) applications, and initially report experiments on use cases where the focus is on communications in general, without necessarily being data-driven and tailored for ML [5, 6, 7].

Building upon these ideas, works such as [8, 9] use this same kind of software orchestration, but explicitly for AI/ML applied to 5G/6G data set generation, however, for the best of the author knowledge, these approaches do not explicitly exposes interfaces where a reinforcement learning agent can directly interact with the environment within the simulation flow. That is important because, having the means of placing the AI "inside the loop", or in other words, en-

abling a model to receive information and take actions that reflect back on the simulation flow, is necessary in order to have simulations that can work as an RL environment.

Not only that, but besides including the AI in the loop, these hybrid simulations applied to RL must also mind its suitability to produce an environment able to train an agent that generalizes well for real, never before seen, situations. This concept is called *reality gap* [10] and is an element to be taken into account when considering the realism of the simulation. That is because having a small *reality gap* between virtual and real scenarios become increasingly crucial as the service requirements also become more stringent in the targeted use case. Examples of such situations can be found in the training of autonomous vehicles, where an agent does not have the freedom to train from scratch in a real road, without rising safety concerns [11], or in a communication system that cannot fail to deliver a good Quality of Service (QoS) for its users for the entirety of the time.

Another case where the *reality gap* must be minimized is when dealing with the so-called *digital twins*. In summary, digital twin is a paradigm in which a target, that can be a physical asset or a process, is monitored and have its states stored for purposes such as predicting future events or performing adjustments in the present time, for instance, preventing an incoming congestion on a network, or scheduling maintenances. The digital twin system synchronizes a virtual representation of the monitored target in a two-way fashion, reflecting changes in the physical counterpart back in the virtual one and vice-versa. So, if an RL agent wants to leverage a digital twin to produce a realistic environment where it can test a given hypothesis or improve its current performance based on the most recent feedback, without negative effects on the physical counterpart, the digital twin, and the set of tools that enables it, must be at the same time realistic and fast enough to allow the agent to adjust itself and act back on the environment while the data used in its training process is still relevant.

Therefore, leveraging the fact that 5G and beyond systems will benefit from rich contextual information to improve performance and reduce loss of radio resources to support its services [4, 12, 13], the key idea in this work is to use realistic representations of deployment sites, together with vehicle physics, sensor and communication network simulations, to generate a virtual representation that enables training and testing RL agents for tasks related to the PHY of the telecommunication system, such as beam selection and/or user scheduling.



**Figure 1.1:** CAVIAR simulation scenario, depicting the radiation pattern (in light green) corresponding to the chosen beamforming codebook index to serve a drone (at the right).

## 1.2 Proposal

This work proposes a simulation methodology named Communication Networks, Artificial Intelligence and Computer Vision with 3D Computer-Generated Imagery (CAVIAR), with a preliminary idea proposed in [14], as a methodology suitable to generate environments with varying *reality gap* and computational complexity, to train and test RL agents, mainly for problems related to the PHY of the 5G networks.

More specifically, in this work, the joint beam selection and user scheduling task is posed as a problem that must be solved with RL. The goal is to schedule and allocate resources to Unmanned Aerial Vehicles (UAVs), cars and pedestrians, composing a scenario with aerial and terrestrial User Equipment (UE), as exemplified in Figure 1.1.

The RL agent is executed at the Base Station (BS) and receives a reward based on the service provided to the users, by periodically taking actions guided by the information captured from the environment, which includes channel estimates, buffer status, and positions from a

Global Positioning System (GPS). The training occurs “offline”, without rendering the 3D scenes, but it is possible to render the output in a post-processing step and generate a video. The CAVIAR simulation integrates three components: the communications, the AI/ML, and finally the virtual world subsystems, with the last one being created with AirSim [15] and Unreal Engine [16].

As RL is a difficult learning paradigm, a simpler version of the same problem and environment is also proposed, mainly for educational purposes. This version reduces the complexity of the simulation by removing the computation of packet buffers, substituting it by predefined penalties in case of long periods of repeated allocation of the same user. It also removes the 3D scenarios and, in turn, uses a variant based on the widely known *minigrad* problem [17]. This one will be described in Appendix A.

Hence, the main contributions of this dissertation are summarized as follows:

- It provides two environments, with crescent complexity, for exploration of RL approaches to the problem of user scheduling and beam selection;
- It also outlines the current state-of-the-art research on the use of hybrid simulators for training and testing of RL agents;
- Finally, it formalizes the use of CAVIAR for digital twins in telecommunications.

### 1.3 Dissertation Outline

The rest of the work is organized as follows:

- in Chapter 2, some useful concepts to comprehend the proposed methodology are briefly introduced, focusing on the MIMO beam selection problem and an introduction to the theory of RL;
- in Chapter 3 we elaborate on the research being conducted to address the problem of simulating communication systems for reinforcement learning, and the work about *reality gap* and digital twins;
- Chapter 4 discusses CAVIAR simulations in general: the software used, explaining some features available, such as Hardware-in-the-Loop (HITL), also detailing simulation modes, and providing examples via block diagrams;

- Chapter 5 explores the specific RL problem addressed in this work, with an overall description of the experiment and the specific implementation of the communication and packet traffic models, described in Section 5.2 and Section 5.3, respectively, as well as RL model used and its design;
- Chapter 6 shows the results, followed by a brief discussion;
- Chapter 7 concludes the work and gives the view of the author regarding possible paths to be taken in order to improve the current status of the CAVIAR subsystems, as well as a proposal on how to build a digital twin, based on the same set of tools explored in this work.

# Chapter 2

## Background Concepts

In order to better comprehend the application area, as well as the reinforcement learning methods used, this chapter briefly introduces the concept of MIMO beam selection and reinforcement learning.

### 2.1 MIMO Beam Selection

5G and beyond technologies leverages new frequency bands to serve its users, such as the, once weakly explored, spectrum range of the millimeter waves (mmWaves) [18]. However, their use depends on massive MIMO techniques, such as beamforming, to produce directional beams to propagate the antenna reach more efficiently, as, given its use of higher frequencies, mmWave propagation is more prone to losses due to fading and blockage [18]. This increases the responsibility of the 5G BS, which needs to perform two tasks: to keep track of the valid beams and to choose the most appropriate one in a process called beam selection, that consists of selecting a beam-pair for BS and UE to achieve the best communication channel at any given time [19].

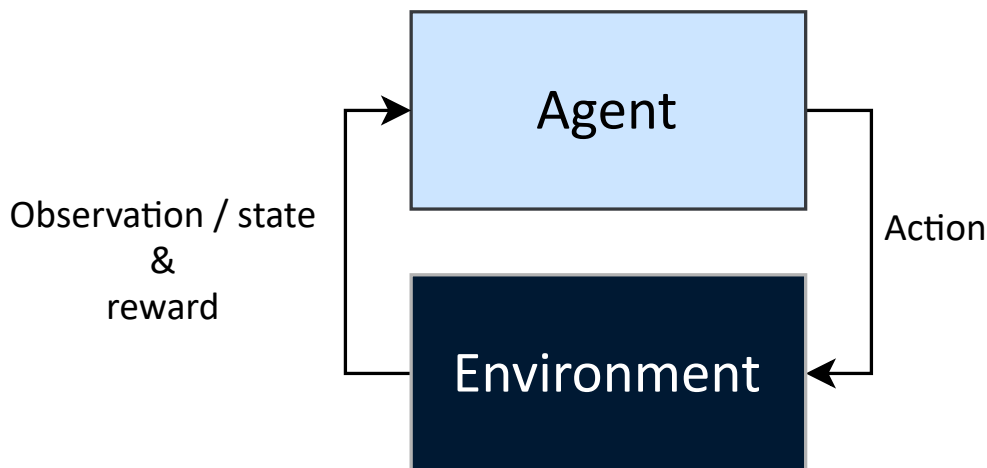
For that, the BS needs to be aware of its surroundings and the users' locations, which can be hard to achieve, specially given that 5G networks will enable not only traditional ground users, such as pedestrians, but also several distinct types of vehicles, like trucks, buses, cars, and even aerial vehicles, introducing new movement patterns [20]. So, in this context, the site covered by the base station becomes increasingly dynamic, and the process of beam selection gets proportionally more complex. This results in systems having to continuously send signals that do not carry information (overhead) [21], compromising a parcel of the channel capacity.



Therefore, decreasing this overhead with intelligent beam selection approaches is a fundamental problem that can enable systems to improve the usage of physical resources (e.g., with lower latency and higher bit rates) [22, 23, 24].

## 2.2 Reinforcement Learning

One possible tool to optimize this beam selection process is RL. As defined by *Sutton et al.* [25], RL is a computational approach to learn from interaction. In this paradigm, an agent makes an **observation**, which can be a partial or an integral representation of the environment that the agent is exposed to and based on it, the agent defines its **state**  $s$  and chooses an **action**  $a$ , derived from its **policy**  $\pi$ , which is, in simple terms, a probability distribution that maps which action to take given the agent current state. This action results in a change in the observed state and a **reward**  $r$ , which is sent to the agent as a feedback for its behavior and allows it to, not only improve its policy, but also improve its capacity to evaluate the environment state that it is currently at. This dynamic can be represented by Figure 2.1.



**Figure 2.1:** The agent-environment dynamic in reinforcement learning.

This terminology is introduced when posing the RL problem as a Markov Decision Process (MDP), which is an idealized way to formalize sequential decision making, computing the effects of actions in terms of rewards and new states, allowing a more accurate mathematical analysis of the problem.

## 2.2.1 Objective of Reinforcement Learning Methods

Defining concepts and a mathematical framework to solve this class of problems is important, as reinforcement learning is a different paradigm from both, supervised and unsupervised learning. Differently from the former, it does not rely on predefined labels, and also, unlike the latter, it does not actively search for underlying structures on the data that it is being exposed to, instead, RL methods focuses on optimizing a value named **return** ( $G$ ), which is the sum of all the accumulated reward throughout a sequence of interactions with the environment.

To achieve this optimization objective, the RL agent faces a trade-off in which it needs to balance its actions on **exploration** of the environment, focusing on updating its knowledge of the dynamics it is inserted into, or on **exploitation**, which consist on taking actions that are currently optimal, ensuring a steady return, but leading the agent to abide in a potentially suboptimal behavior. So, to ensure the exploration of the environment, some methods rely on strategies such as defining a probability  $\epsilon$  for exploratory actions. An example of it is the common  $\epsilon$ -greedy policy, in which the agent exploits the environment with probability  $1 - \epsilon$  and explores it with probability  $\epsilon$ .

Objectively speaking, when an agent explores the environment, it tries to improve its knowledge of the *state-value* function  $v(s)$ , which can also be slightly adapted to *action-value* functions  $q(s)$  as well. Both functions can be described respectively by the *Bellman equations* [25], shown in Equation 2.1, to calculate the value  $v(s)$  of a state  $s$

$$v_{\pi}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_{\pi}(s')], \quad (2.1)$$

and in Equation 2.2, to calculate the value  $q(s)$  of taking a certain action  $a$  given a state  $s$

$$q_{\pi}(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_a \pi(a|s)q_{\pi}(s', a')], \quad (2.2)$$

where  $\pi(a|s)$  is the probability of taking action  $a$  given state  $s$  while following policy  $\pi$ ,  $s'$  and  $a'$  are respectively the state and the action that come immediately after the current step,  $r$  is the reward of taking action  $a$  while being in state  $s$ ,  $p(s', r|s, a)$  is the probability of reaching state  $s'$  and receiving reward  $r$  while being in state  $s$  and taking action  $a$ . Finally,  $\gamma$  is the discount rate, which affects how much the reward received a long time ago influences the return: as  $\gamma$  increases, future rewards become more relevant. So, in other words, what Equation 2.1 and Equation 2.2 tells us is that one can describe  $v_{\pi}(s)$  as the expected return for an agent that starts in state  $s$  and follows the policy  $\pi$  until the end of the given sequence, while  $q_{\pi}(s, a)$  is

the expected return when an agent starts in state  $s$ , takes action  $a$  and, after that, only follows actions derived from its policy  $\pi$ .

There are several approaches to implement RL agents to optimize both  $v$  and  $q$ , as well as the policy  $\pi$ . The next subsections will explore some approaches that rely on Deep Reinforcement Learning (DRL) to perform these predictions.

## 2.2.2 State-Value Estimation Approaches

Note that  $v_\pi$  and  $q_\pi$  evaluate a certain policy " $\pi$ ", which can be the optimal policy "\*" or not. For the case where the optimal policy "\*" is assumed, the state-value equations can be rewritten as:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')], \quad (2.3)$$

while the action-value function, becomes

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')]. \quad (2.4)$$

So, posing RL problems as MDPs gives one the mathematical framework that allows the optimal solutions to be derived theoretically. However, note that the *Bellman optimality equations* presented in Equation 2.3 and Equation 2.4 assume previous knowledge of the environment, mainly in the form of the transition probabilities  $p(s', r|s, a)$ . In the rare case that someone is dealing with an environment where these probabilities are known, the optimal  $v_\pi$  can be obtained via a *model-based* method, such as Dynamic Programming [25], but, for the vast majority of situations, these environment dynamics are unknown, which lead to the need of using *model-free* methods. Such methods update their knowledge of the environment via interactions with it, following the dynamic shown in Figure 2.1. One of the simpler *model-free* algorithms is the Monte Carlo method, which will be briefly described in the next paragraph, as it provides good support for the concepts explored by the state-of-the-art approaches, such as Advantage Actor Critic (A2C) and Proximal Policy Optimization (PPO).

Monte Carlo has some variations, such as *first-visit* or *every-visit* where it only considers the return received by the first encounter with a certain state, or it computes the experience of every visit on a state in its state-value update, respectively. So, as every mode rely on improvements on the estimation that occurs only at the end of an episode or trajectory, the *first-visit* Monte Carlo method is the one that is going to be described, as it gives enough insight into

how the other modes work. The only difference from *first-visit* or *every-visit*, can be considered the modification of the computation of the return  $G(s)$  for state  $s$ , which is made for each episode  $e$ , in case of the *first-visit*, and for every time it reaches state  $s$  during a given episode, in *every-visit*. Monte Carlo *first-visit* improves its  $v(s)$  estimation as described in Algorithm 1.

---

**Algorithm 1:** Monte Carlo *first-visit* method.

---

**foreach** episode  $e$  in Episodes **do**

**while**  $t \leq Timesteps$  **do**

$G_e(s) \leftarrow G_e(s) + r_t;$

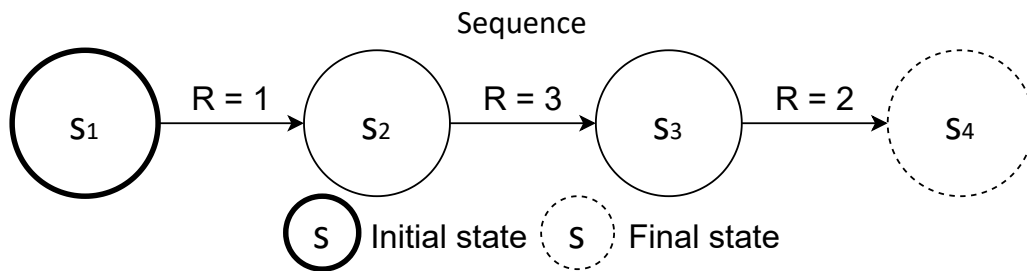
**end**

$v(s) = \frac{\sum_e G_e(s)}{N_e}$

**end**

---

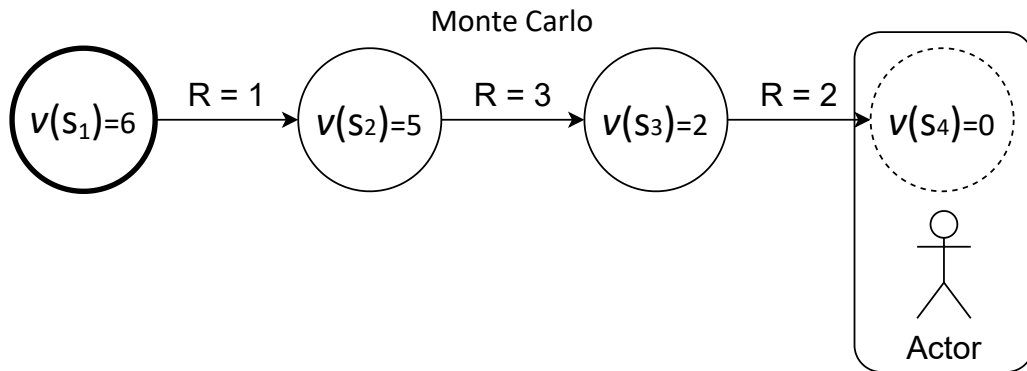
So, given the sequence of state, action, reward and new state proposed in Figure 2.2, the Monte Carlo approach solves the problem of calculating the state-value  $v(s)$  by completing an entire trajectory, starting from the initial state, on to the final state, and only then, based on the received rewards, it calculates  $v(s)$ . In fact, it improves its estimate as the number of new iterations through the environment also grow, by taking a mean of the return values divided by the number  $N_e$  of episodes.



**Figure 2.2:** A given sequence of states and rewards, composing an episode.

This can be exemplified by Figure 2.3, where the Monte Carlo agent waits until the end of the given episode in order to compute the  $v(s)$  of each state, based on the already known rewards.

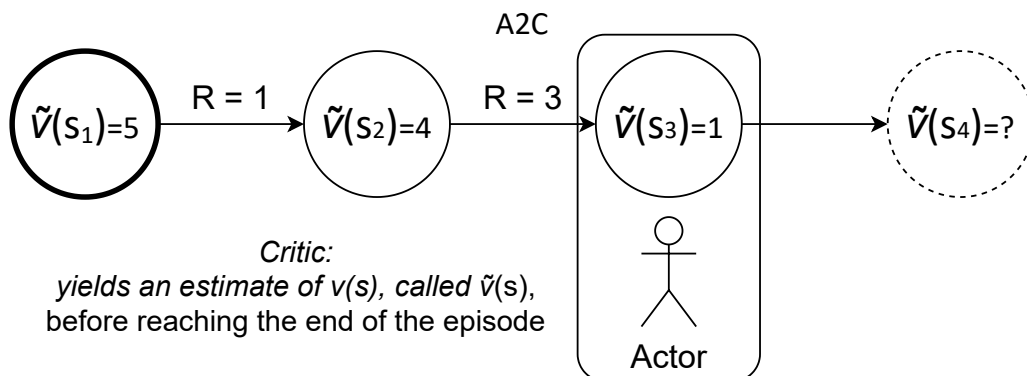
This allows one to estimate state-values that can become very close to the optimal state-value, however, this type of calculation is also very slow, as it needs several runs to yield a good estimate. Therefore, many other approaches were proposed, such as Temporal Difference



**Figure 2.3:** Monte Carlo estimation of the state-values.

(TD) and n-step bootstrapping methods, until the growing complexity of state and action spaces eventually led to the need of using functions to define  $v(s)$  and  $q(s)$ , leading to the advent of DRL. In this context actor-critic methods such as A2C ended up serving as the foundation for more sophisticated approaches [25].

An A2C agent relies on neural networks to perform the roles of critic and actor, the first is responsible for receiving the state observation and estimating the  $v(s)$ , and the second one is responsible for defining the probability distribution that will influence the action choice. Besides that, differently from Monte Carlo methods, for example, A2C do not need to wait until the end of an episode to generate state-value estimates.



**Figure 2.4:** Actor-critic estimation of the state-values.

As Figure 2.4 points out, the A2C agent is able to leverage its critic to estimate future returns before reaching the end of an episode. It uses the difference between the  $v(s)$  predicted by the critic ( $\tilde{v}(s)$ ), and the calculated estimations, to evaluate its predictive capability, as well as the error between predicted state-value and the actual reward obtained by an action at that

state, named *advantage*, to further improve the actor actions choice.

Drawing from these advancements, this work uses the PPO method, which is characterized for being a sample efficient method, in other words, it tends to need less training episodes to converge to a good policy, when compared against other approaches in several baseline environments, as reported in its source paper [26]. It is a state-of-the-art DRL method based on actor-critic, that adopts policy gradient approaches to continually improve the reward obtained from its action choice. It draws ideas from a precursor named Trust Region Policy Optimization (TRPO) [27], and tries to prevent destructive policy updates by defining a threshold value  $\epsilon$  that keeps its gradient descent steps inside a safe region, but without the extra computation present in TRPO, that needs to also calculate a Kullback–Leibler (KL) divergence factor.

# Chapter 3

## Related Works

Having known more about MIMO beam selection and reinforcement learning, to better contextualize CAVIAR, this chapter explores some efforts regarding simulations applied to telecommunications, starting with works related to the generation of synthetic data, by leveraging the integration of different simulators, mainly for applications in Deep Learning (DL). Then, we proceed to investigate different approaches, such as the hybrid simulators, in particular a hybrid traffic-network simulator for vehicular networks, which not only integrate simulators for data set generation, but also allows visualization and greater parameter configuration. After that, in Section 3.3 we address concepts such as *reality gap* and studies that explore the generalization capacity of reinforcement learning agents trained in virtual environments when exposed to real, never before seen, situations. Finally, Section 3.4 presents the concept of digital twins, its broad scope, its adoption by several companies, and how it presents itself as the potential evolution of the current efforts to generate synthetic data and environments for research in reinforcement learning agent training in general and also for communication systems.

### 3.1 5G MIMO Data Generation

The 5G and future networks presents many use cases with challenging service requirements. One of the main strategies to reach these stringent needs, is to use frequencies in the unoccupied bandwidth available in the mmWaves spectrum [20]. However, prior to deploying networks that use this technology, there is a need to correctly assess the environment in which the propagation will take place, what is usually accomplished via the execution of measurements campaigns, that are both, financially expensive and very laborious [28]. Considering

this context, generating propagation channel data through simulations is a reasonable way to alleviate data scarcity.

Several works aim to address the obstacle of the lack of data to apply AI for future networks. Among those, some notable ones are Raymobtime [4], DeepMIMO [9], and Vision-Wireless (ViWi) [29]. The first one proposes a methodology that combines a vehicle traffic simulator with a Ray Tracing (RT) simulator to generate channel realizations representing 5G scenarios with transceivers and objects mobility. It has the advantage of using realistic channel models due to the utilization of ray tracing, but lacks in two aspects: the first one is realism of vehicle movements when considering situations outside pre-established mobility patterns, such as during collisions, because it works in discrete manner by using SUMO traffic simulator [30], which does not account for vehicle geometry; and the second one is fluidness, as it is not a real-time simulation, having the need to obtain a channel realization at each step, by repositioning the vehicles according to the vehicle traffic simulator and invoking the propagation simulator afterwards. This ends up leading to the need of using it beforehand to generate data and, only then, utilize it in an ML workflow.

DeepMIMO also suffers from the very same limitations, as it builds upon the same ideas. However, it tries to differentiate itself from Raymobtime by proposing to use a generic simulation scenario, allowing the users to adjust a set of system and channel parameters, such as the number of antennas, the number of Orthogonal Frequency-Division Multiplexing (OFDM) subcarriers, and the number of channel paths, etc.

Finally, the most recent versions of the Raymobtime data set also introduced multimodal data, extracted from simulating other sources of information, for example, Light Detection and Ranging (LiDAR) and camera images, to benefit from the increasing variety of sensor signals to help reduce the overhead associated with link configuration in mmWave communication systems [12, 31]. In counterpart, as an evolution of the DeepMIMO, there is ViWi[32], which relies on the same principles regarding the addition of multimodal data seen in Raymobtime, and, as its predecessor, distincts itself by the assuming that it allows a more generic simulation in terms of adjustable parameters.



## 3.2 Hybrid Simulators for Communication Networks

If from one side we have data-driven simulations, that have the sole purpose of generating data that will later be used in a AI/ML workflow, from the other we have simulators that seek realism by integrating several expertises, but without proposing a clear integration with ML models inside the simulation loop and much less providing examples where these simulators could be used as reinforcement learning environments. Among those, some notable works are LIMoSim [6] and Veneris[5].

LIMoSim presents an extension of the ns-3 set of tools by proposing a module that enables the joint simulation of hybrid ground-based and aerial communication networks, extending ns-3 with mobility models for cars and drones, and a simple 3D visualization feature obtained with Open Graphics Library (OpenGL). They do this using a *shared codebase* coupling method instead of the more common High Level Architecture (HLA) approach. In the latter, one must synchronize/orchestrate several simulators and, besides being a traditional method, its complexity in terms of maintenance and usability is tackled by the former, where they extend the ns-3 simulator by inheriting from parent classes and leveraging the event queue present within the same codebase of the original software. Nonetheless, LIMoSim still and event-based simulator, such as ns-3 on which it is based.

Another hybrid simulators is Veneris, that integrates Unity3D, a multipurpose engine [33], OMNeT++, a network simulator, and Opal, a ray-launching propagation simulator that uses Graphics Processing Unit (GPU), developed by their team and validated against a custom ray tracing software implemented in MATLAB. It focuses only on car mobility and generates its own vehicle systems models, without implementing features that allow integrating with the real hardware, such as in the concept of HITL, explored in Subsection 4.3.2. Also, Veneris does not explicitly expose experiments where it interacts with a reinforcement learning agent to execute actions that would influence the state, and therefore, the next simulation step.

## 3.3 Improving Simulations Realism to Close Reality Gap

The difficulty of transferring simulated experience into the real-world is called *reality gap*, and, in order to close it, several approaches are taken, such as leveraging computer vision and the use of simulators containing features such as photorealism, physics and sensors models. In [10], to improve the generalization capacity of autonomous vehicles policies trained

in simulated environments, a simulator based on Unreal Engine 4, named Car Learning to Act (CARLA), and the Unreal Engine plug-in, named AirSim, are used to recreate original traffic scenarios using prior information of a given scene, adding realism to the virtual scenes based on real ones. Similarly, in [34] a real vehicle was able to drive through 3 kilometers on a real traffic lane using a DL model trained on synthetic data from a virtual world also built on Unreal Engine 4. In [11] it is demonstrated that, using virtual simulations with enough realistic elements, it is possible to learn policies that not only perform well in the virtual scenario where it was originally trained, but also shows good generalization for previously unseen real-world scenarios. They focus on the problem of obtaining policies for autonomous driving vehicles and use a simulation engine named Virtual Image Synthesis and Transformation for Autonomy (VISTA). Their methodology consists on synthesizing photorealistic and semantically accurate local viewpoints based on a repository of sparsely sampled, human collected, trajectories. For example, given a state  $s_t$ , composed by an image representing the view from the driver seat during a real cruise present in the repository, the next state  $s_{t+1}$ , after the agent takes an action, will be an artificially generated image produced by VISTA, by taking the closest next image in that cruise and adaptating it using computer vision techniques. By using this technique they were also able to deploy the policy on a real vehicle.

Therefore, these experiences from the use of RL in the autonomous driving field, show that using hybrid simulators capable of providing realism in diverse aspects, such as photorealism, mobility patterns, real hardware integration, and other relevant characteristics for the investigated problem, is beneficial to close the reality gap and improve the use of RL as a tool to deal with problems of interest in 5G and future networks, specially on tasks with stringent service requirements and highly dynamic scenarios.

### 3.4 Digital Twins

In order to reduce the reality gap, a virtual environment that hopes to allow training an agent with the expectation that it can perform equally well on the real world, must also strive to be in sync with any changes that affects the status of the environment it is trying to simulate. To address that synchronization between the real world and its virtual representation, a paradigm named digital twin emerged.

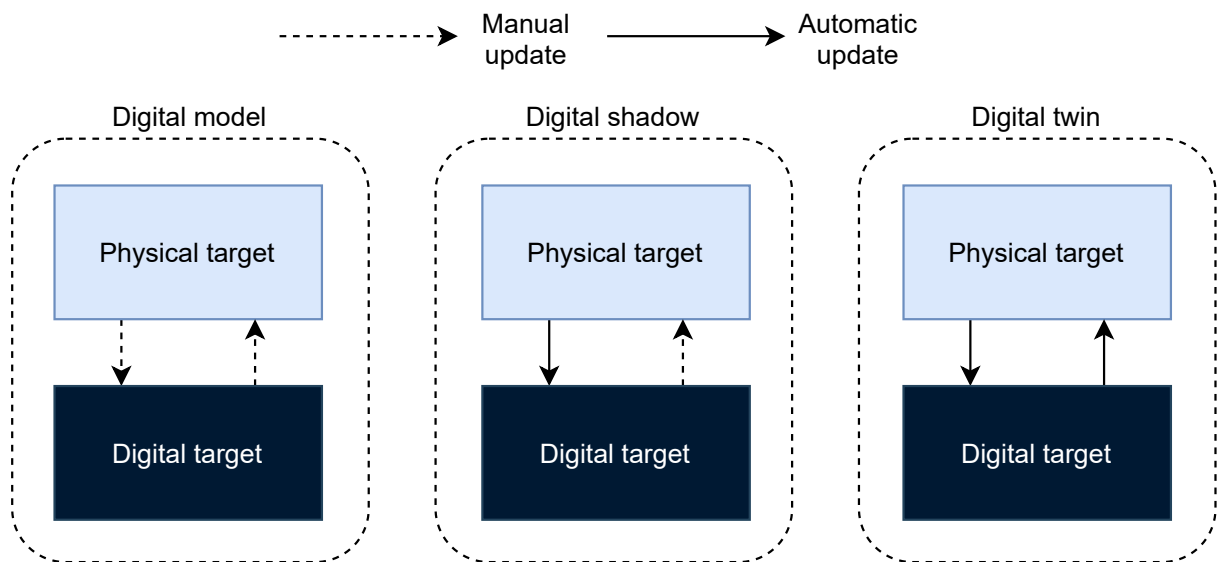
Digital twins are already commercially explored in several areas such as manufacturing

and aviation [35, 36, 37, 38] and now, big tech companies such as Ericsson, Microsoft, Siemens and IBM [39, 40, 41, 42, 43, 44, 45], also, some other small companies [46, 47], and open-source initiatives [48, 49] started to explore this concept in their respective areas as well. Moreover, although it is still in its infancy regarding its use on wireless systems, digital twins are already envisioned as an important use case of 6G networks, with increasing amount of works relating them to communication systems, such as [50, 51, 52, 53].

Digital twins can be used to monitor assets and processes, appearing in diverse manners, depending on the application. In the literature, digital twins can be sub-categorized by their level of integration with their physical counterpart:

1. **digital model**, which is a virtualization, without any automatic interaction or update;
2. **digital shadow**, is the same as 1, with the addition of automatic state update of the digital target;
3. **digital twin**, that not only updates its states automatically, but also adds the possibility to interact with the monitored counterpart.

Figure 3.1 illustrates this digital twin sub-categorization.



**Figure 3.1:** Definition of digital twins sub-categories in terms of their level of integration with their physical counterparts (modified figure from Kritzing, Werner et al. [36]).

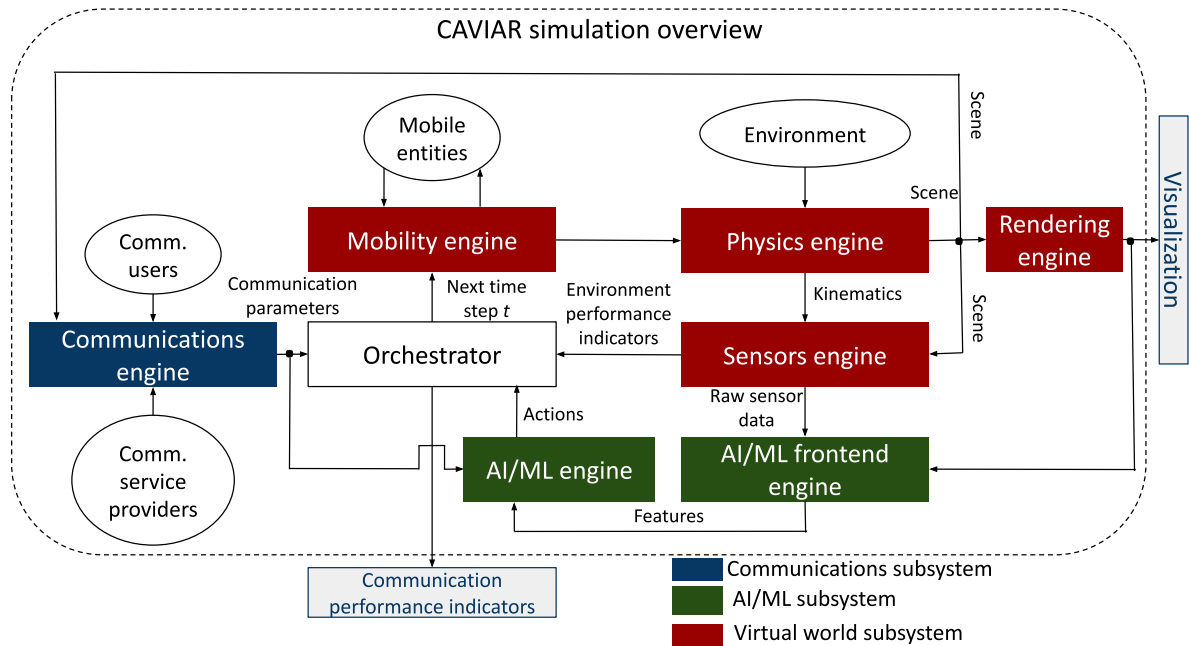
A hybrid simulator capable of capturing the Key Performance Indicators (KPIs) of a given process is a viable option to serve as basis to a digital twin, providing an up-to-date virtual

environment for the training of a model. In this work, that possibility is still not completely explored, however, Section 7.2 provides an overview of an implementation that leverages tools and experience obtained from this effort.

# Chapter 4

## CAVIAR Simulations

Considering the possibilities that hybrid simulations provide to the training and testing of RL agents, the CAVIAR methodology is proposed. As initially discussed in [8], the CAVIAR methodology incorporates three main components: communications, AI/ML, and virtual world subsystems. In the next sections, we describe the method, focusing on the overall description, simulation modes, and software used. After that, Chapter 5 elaborates on how the simulation was realized for the user scheduling and beam selection environment.



**Figure 4.1:** Representation of a possible CAVIAR simulation, integrating the three subsystems.

## 4.1 Overall CAVIAR Description

CAVIAR main idea is to allow a simulation to integrate the three previously mentioned subsystems. Figure 4.1 displays the main components in a CAVIAR simulation, being brought together by an orchestrator script. This figure represents the current implementation, where three different pieces of software are used to realize the simulation. The next paragraphs describes in general the blocks that encompass the proposed simulation strategy.

Starting from the left, the *Communications engine* block, in blue, is the one that handles all information regarding the communication aspect of the simulation, such as data traffic, buffers, wireless channel generation and propagation of radio signals. It can be realized either by a set of scripts, a single software, or even an integration of them. For example, the packet traffic can be handled by a specific program such as ns-3<sup>1</sup> allied with the orchestrator, while the communication channels can be modeled with the help of a propagation software such as Remcom Wireless InSite<sup>2</sup>, or Altair WinProp<sup>3</sup>, given that the computational needs are met.

The 3D assets used in the *Environment* and as *Mobile entities*, such as UAVs, cars, buildings, etc, are either created or obtained online, using the same methods described in Raymobtime [4], in other words, leveraging OpenStreetMap<sup>4</sup> and free 3D models libraries. They compose the simulation environment as fixed and/or mobile objects, whose eventual movements and interactions are managed by the *Mobility engine* and by the *Physics engine* of the virtual world subsystem, respectively. The *Sensors engine* output can be composed of any sensor related information, ranging from camera images to GPS data, and constitutes one of the possible inputs to the *AI/ML frontend engine*, which performs the necessary pre-processing steps before using this information in an ML pipeline.

The *AI/ML engine* receives pre-processed signals and communication parameters that were simulated in the virtual world environment and suggests actions that are then implemented by the *Orchestrator*, which also considers parameters from the *Communications engine* and the *Environment*.

---

<sup>1</sup><https://www.nsnam.org/>

<sup>2</sup><https://www.remcom.com/wireless-insite-em-propagation-software>

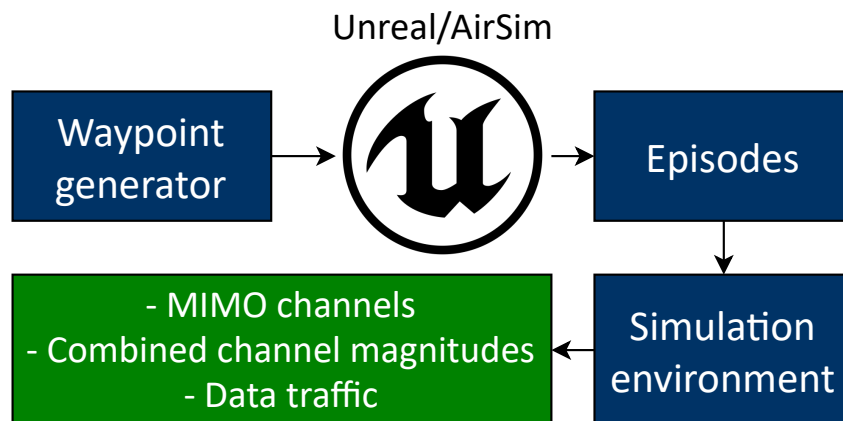
<sup>3</sup><https://www.altair.com/resource/altair-winprop-datasheet>

<sup>4</sup><https://www.openstreetmap.org>

## 4.2 Out-loop and In-loop Simulation Modes

As a simulation for an RL environment, CAVIAR benefits much more from keeping the AI inside the loop, however, to also act for the purpose of generating data sets to be used in an ML pipeline, besides having the common mode of operation, named *in-loop*, it is also allowed to run in a much more decoupled way, called *out-loop*. These two modes of operation refer to whether the AI/ML engine is used within or outside of the simulation loop, respectively.

An in-loop simulation represents the usual mode of operation depicted in Figure 4.1. It allows the subsystems to interact with one another in an orchestrated manner, including the AI in the loop and generating data that is fed back to the system during the simulation. On the other hand, in an out-loop simulation, CAVIAR is used to generate data, similarly to the approach adopted by Raymobtime and DeepMIMO. This mode decouples the communication subsystem from the AI/ML and virtual world ones, having each one run in different moments. Figure 4.2 exemplifies the outloop CAVIAR data generation pipeline.



**Figure 4.2:** Out-loop CAVIAR data generation.

First, the virtual world subsystem is used to generate a set of Comma-Separated Values (CSV) files containing the trajectory data of all moving objects within a simulation, named *episode*. To generate it, an waypoint file, which is a text file with reference points for the mobile entities, must be executed by Airsim, and during its execution, the information from the mobile elements are stored in the *episode*. Each one of these *episodes* lasts about 3 minutes, with a sampling interval of 10 ms, and is composed by columns related to position and orientation for pedestrians and cars, with the addition of acceleration, linear, and angular velocities for UAVs as described in Table 4.1.

**Table 4.1:** Content of an episode file generated for the out-loop CAVIAR mode

Content	Description
timestamp	Moment when the info was gathered
obj	Name of the mobile entity
pos_x	Position in meters (north positive)
pos_y	Position in meters (east positive)
pos_z	Position in meters (down positive)
orien_x	Quaternion in degrees
orien_y	Rotation in degrees
orien_z	Rotation in degrees
orien_w	Rotation in degrees
linear_vel_x	Linear velocity in $m/s$
linear_vel_y	Linear velocity in $m/s$
linear_vel_z	Linear velocity in $m/s$
angular_vel_x	Angular velocity in $m/s$
angular_vel_y	Angular velocity in $m/s$
angular_vel_z	Angular velocity in $m/s$
linear_acc_x	Linear acceleration in $m/s^2$
linear_acc_y	Linear acceleration in $m/s^2$
linear_acc_z	Linear acceleration in $m/s^2$
angular_acc_x	Angular acceleration in $m/s^2$
angular_acc_y	Angular acceleration in $m/s^2$
angular_acc_z	Angular acceleration in $m/s^2$

The second step is to use the *episode* files to obtain information from MIMO channels and data traffic. For that, one must execute the *episode* files within the CAVIAR Communication subsystem, pointing out the UE that will be used. This leads to an output CSV file with the contents disposed in Table 4.2, that stores the environment status during the given simulation step.



**Table 4.2:** Content of an output file generated from a CAVIAR simulation

Content	Description
chosen_ue	Name of the chosen UE
ue_index	Numeric identification of chosen UE
beam_index	Codebook index of the selected beam
x	UE position in meters on the x-axis (north positive)
y	UE position in meters on the y-axis (east positive)
z	UE position in meters on the z-axis (down positive)
pkts_dropped	Number of all dropped packets in the given step
pkts_transmitted	Number of all transmitted packets in the given step
pkts_buffered	Number of all buffered packets in the given step
bit_rate_gbps	Data rate in Gbps achieved during the given step
channel_mag	Magnitude of the combined channel
reward	Reward obtained during the given step

## 4.3 Main Software Tools

To realize the previously mentioned subsystems, CAVIAR leverages the features of Unreal Engine and AirSim, that provides photorealism; physics, automotive, and sensor modeling; and an Application Programming Interface (API), that facilitates the interaction between external scripts and the simulation. The next subsections provide more details about these capabilities.

### 4.3.1 Unreal Engine

The advancements on multipurpose engines led them to transcend their initial objective of being tools for game development and extended their audience to areas such as simulated training, architectural visualization, digital twins, etc. This expansion also had effects on the AI research field, that leverages the sophisticated lighting, collision and 3D modeling available in these tools to provide synthetic data for their models.

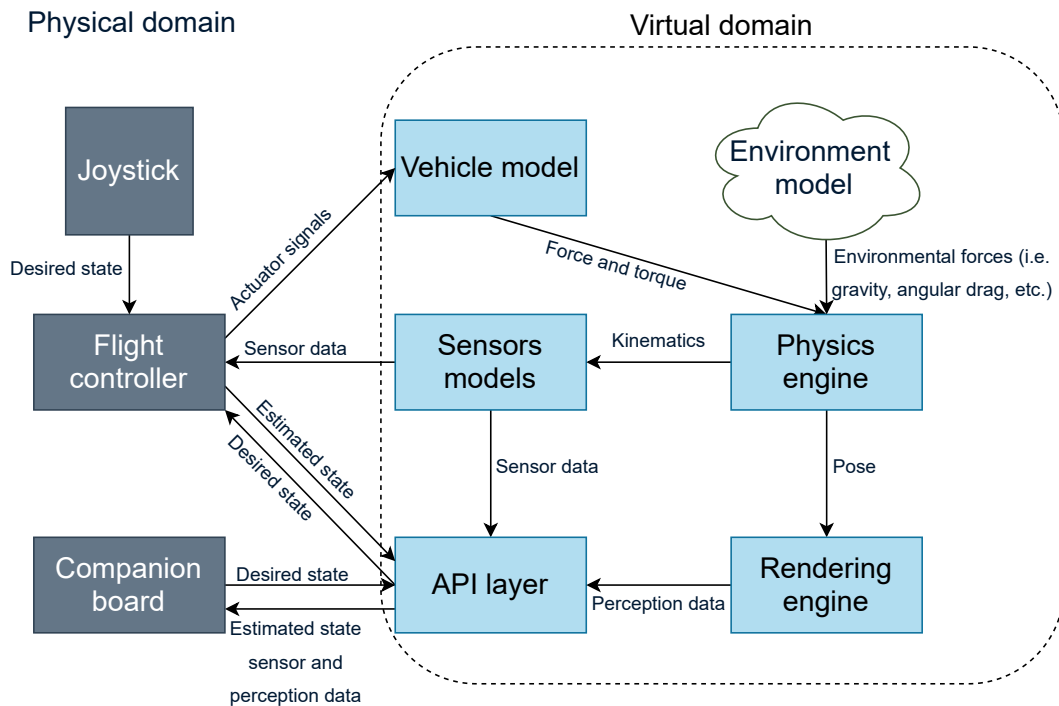
Besides providing photorealistic rendering of 3D scenarios, Unreal also provide a physics engine with useful features for V2X scenarios, such as linear and angular drag, acceleration,

collision detection, and the possibility to extend it as it is an open-source software.

Due to the availability of the previously mentioned features, and also the possibility to integrate more plug-ins such as AirSim, CAVIAR uses Unreal Engine as the platform to implement the virtual world subsystem.

### 4.3.2 AirSim

AirSim [15] is an autonomous vehicle simulator built as a plug-in for Unreal Engine and primarily designed for research involving RL and computer vision for the navigation of UAVs and cars. It provides several capabilities for a simulation ran in Unreal Engine, such as the modeling of vehicles control systems, mainly for the case of drones, environmental physics with regards to gravity, magnetic field, and air pressure and density. It also features sensor simulations, for example, barometer, gyroscope, accelerometer, magnetometer, GPS, RGB and LiDAR cameras. Finally, to increase the realism of the UAV movements, it allows the use of Software-in-the-Loop (SITL) and HITL, both approaches allow the use of real flight controller firmware such as PX4, ROSFlight, etc, to control the drone movement. Figure 4.3 explains how the HITL takes place in AirSim:



**Figure 4.3:** Block diagram explaining how hardware-in-the-loop includes the hardware of the actual drone inside the simulation loop for increased realism (modified figure from Shah, Shital et al. [15]).

During a flight mission, the UAV needs to be connected to a flight controller. This element is the one that will receive a command, given by an autonomous navigation intelligence in a companion board or via a joystick, and translate it into movement. This command is named the *desired state*, or, in other words, where the pilot wants the UAV to be. Once the flight controller receives this command, it interprets signals from the environment in the form of sensor data and, based on that, calculates the actuators signal in terms of force and torque to be applied in the rotors. The calculation of these actuators signals depends on the control strategy implemented by the flight controller.

Hence, these features allows a real hardware, or its firmware, to be connected to the simulation via HITL or SITL, respectively, which increases the realism of the simulation, and also the possibility that the same approach also works in a real life scenario, contributing to the reduction of the reality gap.

# Chapter 5

## Experimental Evaluation

### 5.1 CAVIAR Simulation for User Scheduling and Beam Selection Problems

In 2021, CAVIAR was used in the context of the AI/ML in 5G Challenge, held by the International Telecommunication Union (ITU)<sup>1 2</sup>, serving as the methodology where the *problem statement* PS-006 was implemented. In this challenge, students and professionals from several countries affiliated to the ITU, competed for global recognition and prizes. Since the participants were to undergo evaluations processes, there was a need to improve the reproducibility of training and test procedures, so, the out-loop operation mode proposed in Section 4.2 was adopted, due to providing a set of predefined trajectories for the UE to follow, favoring the predictability of the experiments in general.

Therefore, for this set of experiments, the realization of CAVIAR methodology as an environment for the user scheduling and beam selection problem is based on the one used at the challenge. For it, a set of software tools and scripts were prepared, where the *Physics* and the *Mobility engines*, as well as the rest of the *Virtual World* subsystem, were handled by Unreal Engine and Airsim. Furthermore, for the *AI/ML engine*, an RL environment was developed using PyTorch<sup>3</sup>. Finally, regarding the *Communications engine* used by the CAVIAR simulations, it used a simpler procedure based on geometric MIMO channel modeling [54], instead of implementing methods based on RT. The reason for this choice is that we first want to evolve the

---

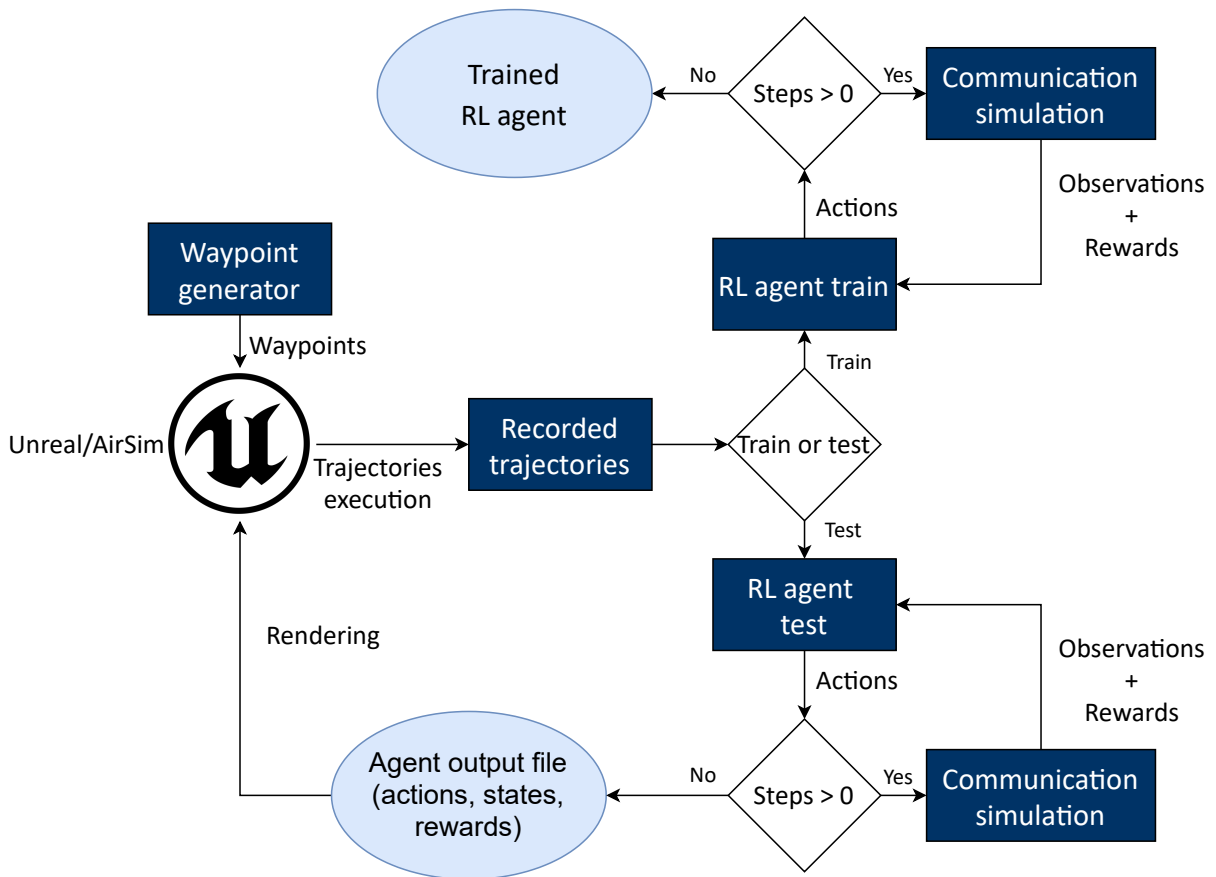
<sup>1</sup><https://aiforgood.itu.int/about/aiml-in-5g-challenge/>

<sup>2</sup><https://www.youtube.com/watch?v=6o6PUsbaWnA>

<sup>3</sup><https://pytorch.org/>

*Communications engine* such that newer versions allow calculating the communication channels in (near) real-time, along with the training of the RL agent.

So, using the virtual scenario provided by CAVIAR, three mobile entities: a pedestrian, a car, and a UAVs were simulated in order to generate a data set of urban mobility. These recorded trajectories are organized in *episodes*, containing spatial information (position, orientation, acceleration, etc.) of each mobile entity as indicated in Table 4.1.

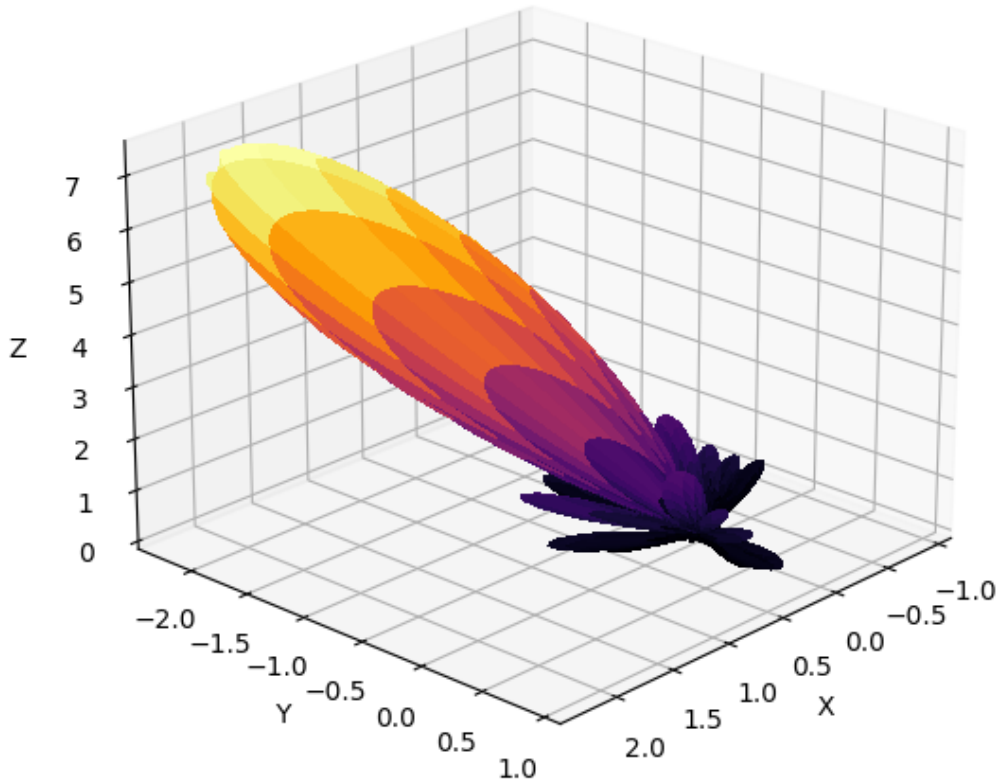


**Figure 5.1:** Out-loop CAVIAR simulation flow used in the experiments.

Then, as shown in Figure 5.1, the recorded trajectories are used as input for the CAVIAR communication simulation, which is responsible for computing the radio channels and others parameters related to the telecommunication system, such as buffer size, etc. This information generated by the communication simulation, along with the spatial data from the trajectories, are the input for an RL agent, that is trained to choose, in each time slot, a user to serve, and the beam that should be used.

## 5.2 Communication Model

For the communication system, it is assumed a downlink transmission using a carrier frequency  $f_c = 60$  GHz and a bandwidth of 100 MHz. The BS serves three distinct receivers, located at the UAV, at the car and used by the pedestrian. The MIMO system corresponds to an analog architecture using a Uniform Planar Array (UPA) with  $N_t$  antenna elements at the BS, and receivers with UPAs with  $N_r$  antennas. Therefore, the MIMO channel between the BS and a given user is represented by a  $N_r \times N_t$  matrix  $\mathbf{H}$ . The codebooks are obtained from Discrete Fourier Transform (DFT) matrices and denoted by  $\mathcal{C}_t = \{\bar{\mathbf{w}}_1, \dots, \bar{\mathbf{w}}_{N_t}\}$  and  $\mathcal{C}_r = \{\bar{\mathbf{f}}_1, \dots, \bar{\mathbf{f}}_{N_r}\}$ . They are used at the transmitter and the receiver sides, respectively. When modeling the beam selection, the chosen beam pair  $[p, q]$  is represented by a unique index  $i \in \{1, 2, \dots, M\}$ , where  $M \leq N_t N_r$ . Each index  $p$  (or  $q$ ) generates a specific radiation pattern, as depicted in Figure 5.2 for a  $N_t = 64$ .



**Figure 5.2:** Example of radiation pattern for specific beam index with a  $8 \times 8$  UPA.

For the  $i$ -th index, the *equivalent channel* can be calculated as:

$$y_i = \mathbf{w}_i^* \mathbf{H} \mathbf{f}_i, \quad (5.1)$$

and the *optimal* beam index  $\hat{i}$  is given by

$$\hat{i} = \arg \max_{i \in \{1, \dots, M\}} |y_i|. \quad (5.2)$$

Also, noise is not taken in account in order to isolate the impact of the beam selection procedure.

The simplified  $\mathbf{H}$  currently represents a Line-of-Sight (LoS) channel. A *narrowband* channel model [54] is used, but *wideband* models can be readily incorporated in case their extra computational cost is not an issue. For simplicity, the users have a single antenna ( $N_r = 1$ ) while the BS has a  $8 \times 8$  UPA ( $N_t = 64$ ).

The *geometric channel model* [54] is adopted with  $L = 2$  Multipath Components (MPCs):

$$\mathbf{H} = \sqrt{N_t N_r} \sum_{\ell=1}^L \alpha_\ell \mathbf{a}_r(\phi_\ell^A, \theta_\ell^A) \mathbf{a}_t^*(\phi_\ell^D, \theta_\ell^D). \quad (5.3)$$

The parameters in Eq. 5.3 are obtained as follows. The phase of the complex-gain  $\alpha_\ell$  is obtained from a uniform distribution with support  $[0, 2\pi]$ . For generating the magnitude  $|\alpha_\ell|$ , first the distance  $d$  between the BS and the given receiver is used to calculate the received power via the Friis equation [55]. The path loss is obtained from this equation and determines  $|\alpha_\ell|$ , which decreases with  $d$ . The elevation  $\phi_\ell$  and azimuth  $\theta_\ell$  angles, for departure (e.g.  $\phi_\ell^D$ ) and arrival (e.g.  $\phi_\ell^A$ ) are obtained from the orientation provided by the LoS path. The nominal LoS angles are slightly changed by adding to them Gaussian random variables with zero-mean and variance of 1 degree. These angles are used to compose the steering vectors  $\mathbf{a}_t$  and  $\mathbf{a}_r$ .

### 5.3 Traffic Model

The users data traffics are defined as Poisson processes with time-varying mean  $\lambda_u[t]$ , with each user  $u$  having a specific traffic intensity defined as a percentage of total throughput, in accordance with Table 5.1. This differentiation is used as a representation of the data requirements of distinct applications. Also, besides these throughput differences between users, two different network load scenarios were specified, representing light and heavy network traffics, with the simulation alternating between them at every 1000 steps.

The BS has an individual buffer with size of 1 Gigabit for each one of the receivers and is assumed that each packet has 400 bits. The incoming traffic for each user is buffered when

**Table 5.1:** Network load information for light and heavy scenarios

Network load	Total throughput	UAV (%)	Pedestrian (%)	Car (%)
Light	40 Gbps	50%	20%	30%
Heavy	90 Gbps	50%	20%	30%

there is buffer space available, otherwise the excess of packets are *tail-dropped*. The packets are also dropped when they occupy the buffer for more than 10 steps after arriving.

## 5.4 Possible Inputs to RL Agents

The inputs (also known as states or observations) can be selected from both, information provided in CSV files (position (x,y,z), velocities, etc.) or obtained from the environment, such as the buffer state and channel information for specific beam index previously chosen. More specifically, the RL agent can use: the UEs geographic position in X, Y, Z, with the origin of the coordinate system being on the BS RL agent. Also, the UEs orientation in the three rotation coordinates: the front and side roll angles, as well as its rotation over its own axis. Besides that, there is also dropped, transmitted and buffered packets. Finally, some other two available input features for the agent are the bit rate and the channel magnitude at each step of the simulation.

Note that we assume the BS (or equivalently, the RL agent) does not know the best index  $\hat{i}$ . In practice, this would require a full beam sweep, which is assumed to be unfeasible in our model due to stringent time requirements. Similarly, given that the RL agent chose user  $u$  and beam index  $j$  at time  $t$ , it learns only the magnitude  $|y_j|$  and the spectral efficiency  $S_{u,t,j}$  for this specific user and beam index at time  $t$ .

The channel throughput is obtained by multiplying the spectral efficiency  $S_{u,t,j}$  by the bandwidth BW, as shown in Equation 5.4

$$T_{u,t,j} = S_{u,t,j} \text{BW}, \quad (5.4)$$

and indicates the maximum amount of bits that can be transmitted. An empirical factor is used to adjust  $T_{u,t,j}$  in order to define the network load presented in Table 5.1, such that, for the given input traffic, some packets have to be dropped.

Algorithm 2 summarizes the steps for executing an already trained RL agent.



---

**Algorithm 2:** High-level algorithm of the RL-based scheduling and beam selection problem.

---

Initialization for a given episode  $e$ ;

**while**  $t \leq N_s^e$  **do**

- 1) Based on the number of bits in the buffers of the users and other input information, RL agent schedules user  $u$  and selects beam index  $i$ ;
- 2) Environment calculates combined channel magnitude  $|y_i|$  and corresponding throughput  $T_i$ ;
- 3) The number of transmitted bits is  $R_i = \min(T_{u,t,j}; \mathbf{b}_u)$  ;
- 4) Update buffers;
- 5) Receive new packets;
- 6) Eventually drops packets;
- 7) Environment calculates rewards and updates its state;
- 8) Update buffers again;

**end**

---

## 5.5 Evaluation of the RL agent

To evaluate the RL agent, the return  $G$  over the test episodes is used. The return  $G_e$  for episode  $e$  is

$$G_e = \sum_{t=1}^{N_s^e} r_e[t], \quad (5.5)$$

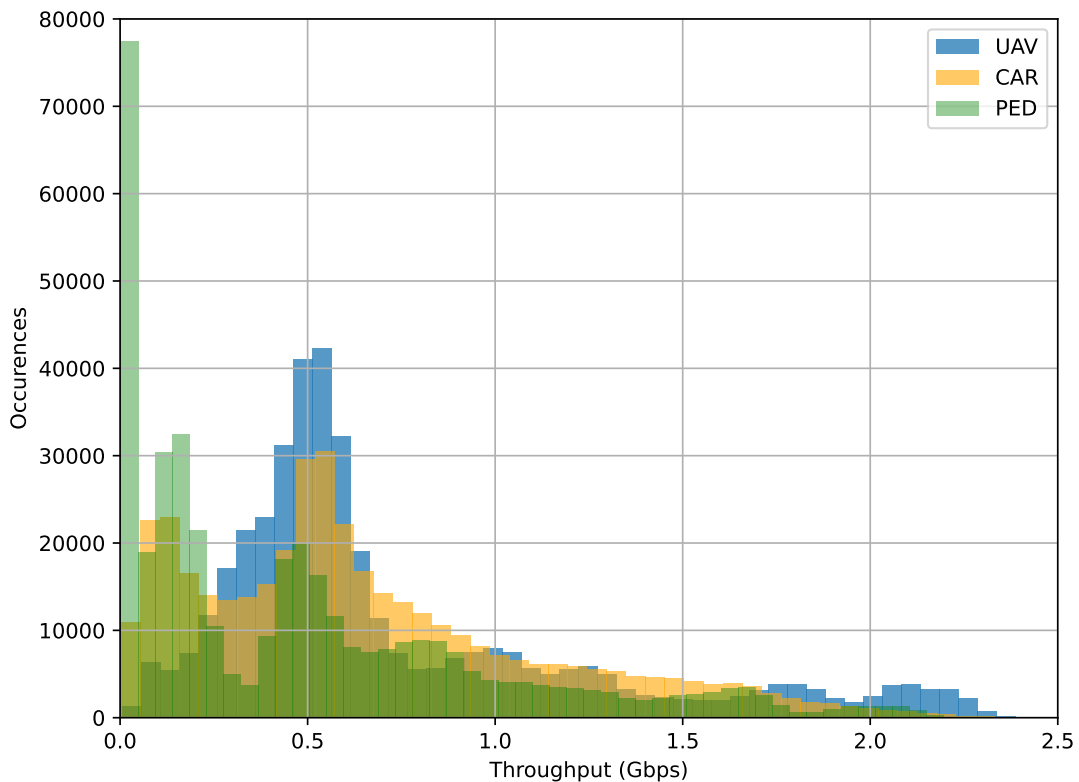
where  $N_s^e$  is the number of steps in episode  $e$ . At each time  $t$ , a single user can be served and the corresponding reward  $r_e[t]$  at discrete-time  $t$  is a ratio of transmitted and discarded packets given by

$$r_e[t] = \frac{P_{\text{tx}}[t] - P_d[t]}{P_{\text{tx}}[t] + P_d[t]}, \quad (5.6)$$

where  $P_{\text{tx}}[t]$  and  $P_d[t]$  correspond, respectively, to the total amount (summation for all users) of transmitted and dropped packets at time  $t$ . The reward  $r_e[t]$  is restricted to the range  $-1 \leq r_e[t] \leq 1$  if it does not receive any additional penalty besides dropped packets.

## 5.6 Characterizing the Communication Scenario with Baseline Approaches

To better evaluate the performance of the agent, some baseline approaches to the problem were also implemented. The first one is meant to characterize a bad result, by choosing the actions randomly, following a discrete uniform distribution. Another baseline, but now meant to characterize a good behavior, follows a *sequential* user scheduling pattern in a round-robin fashion (uav->car->pedestrian->uav->car->pedestrian...), and always uses the optimum beam index  $\hat{i}$  for the selected user, obtained via exhaustive search (oracle beam selection). In Figure 5.3 we exemplify the channel maximum throughput of this experiment when using this baseline in 200 episodes.



**Figure 5.3:** Distribution of the channel throughput when using always the best beam index  $\hat{i}$  and a simple scheduling strategy, that chooses users sequentially in a round-robin fashion (uav->car->pedestrian->uav->car->pedestrian...), over 200 episodes.

# Chapter 6

## Results and Discussion

### 6.1 Results

The following results are extracted from a PPO agent from the Stable-Baselines3 package [56], trained with default parameters during 400 episodes, around 3 million steps. The actor and critic neural networks have both, two hidden layers of 256 neurons, with Rectified Linear Unit (ReLU) as activation function.

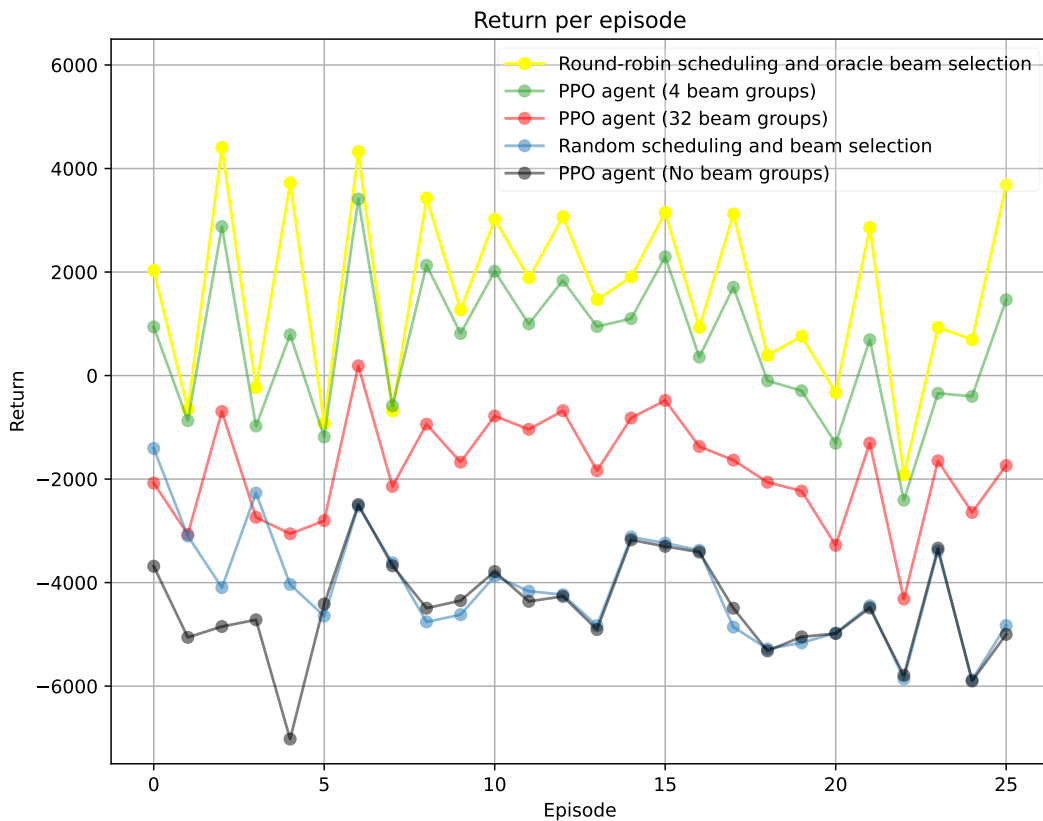
The agent observation is defined by 19 features, which are: each UE individual position in XYZ, their dropped, transmitted and buffered packets during the last time step, and the last scheduled user. The action space is composed by a vector with two integers: a numeric identification of the user being allocated at the specific time step  $t$ , that ranges between  $[0, 2]$ ; and a beam group, that represents a subset of the defined codebook and is within the range  $[0, N_{groups}]$ , where  $N_{groups}$  is the number of beam groups in which the codebook with 64 indexes was divided. Once the agent chooses one of those groups, the best beam inside the chosen subset is used to serve the selected user. This approach resembles the one used in [4, 12, 31], where the model had a *top-k* approach to beam selection, yielding as output a subset of beams to execute the beam sweeping, reducing the overall overhead of pilot tones on the communication channel.

Finally, the agent used the reward given in Equation 5.6. However, during the training process a penalty was also included in order to prevent the agent to schedule a single user in every step. This penalty consists of keeping a variable  $N_{seq}$  to count how many times a user was chosen in sequence, and subtracting  $2 \times N_{seq}$  from the reward. In case that  $N_{seq} \geq 10$ , the reward received a penalty of  $4 \times N_{seq}$ .

So, three agents were trained in total, providing a comparison between the performance of

the two baselines defined in Section 5.6 against an agent with different number of beam groups, namely 4, 32 and 64 beam groups, with the last one being considered an agent that does not choose a group, but a beam index, therefore being called "No beam groups".

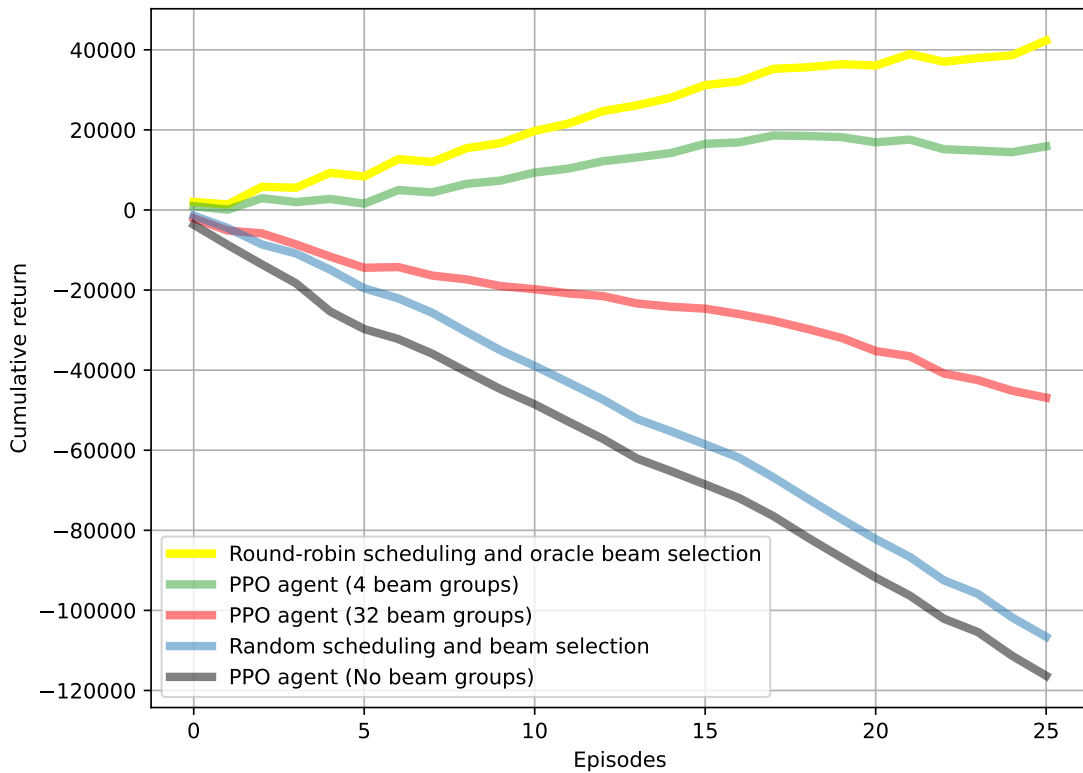
Figure 6.1 shows the agents performances over 25 test episodes, with Figure 6.2 providing another take on the same experiment, but now considering the cumulative return.



**Figure 6.1:** Performance of the agents and baseline approaches over 25 test episodes.

As expected, the baselines were able to provide a fine estimate of both good and bad behaviors, with the one that implements round-robin and always uses the best beam to serve the UE being the most performant (yellow line), and the one that always chooses its actions randomly being the second worst overall (blue line).

The performance of the RL agents improved as the number of beam groups decreased, given that this reduced the possible beam selection actions, thus suggesting that the beam selection procedure is the bottleneck of the current approach. To further investigate this hypothesis, another experiment was executed, this time to evaluate whether the environment was able to train



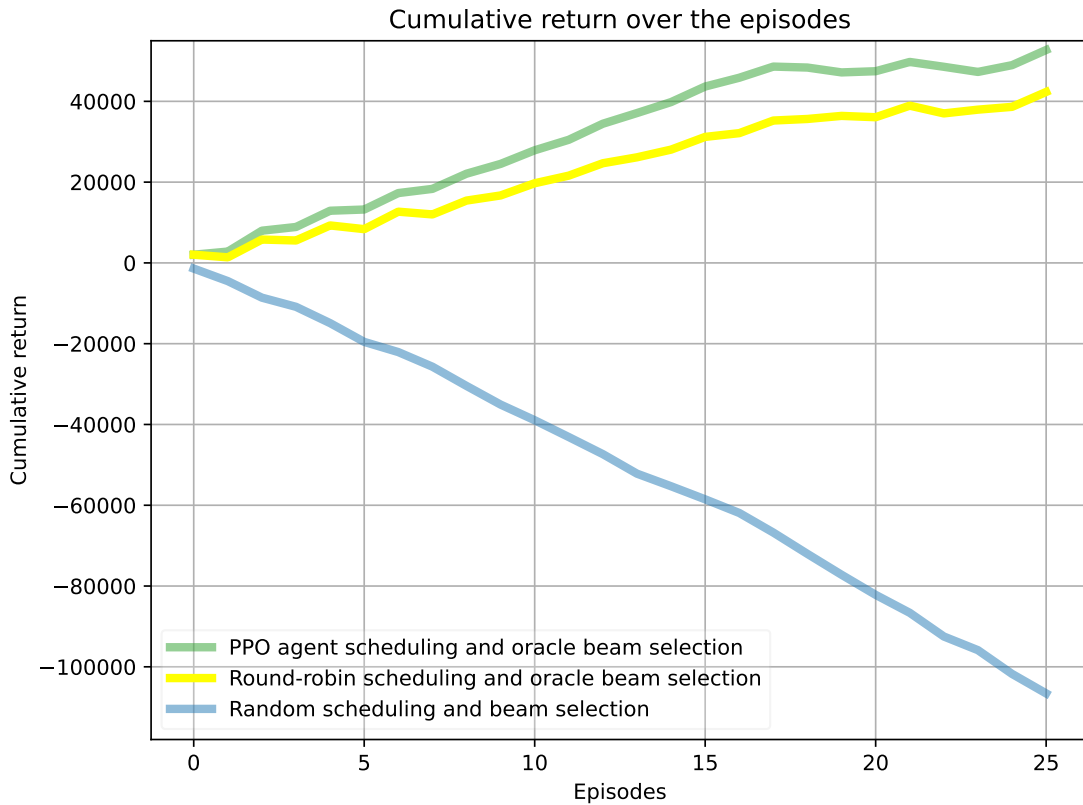
**Figure 6.2:** Cumulative reward of the agents and baseline approaches over 25 test episodes.

an agent that displayed reasonable scheduling performance. So, an agent with action space reduced to perform only user scheduling was trained over only 20 episodes (around 124 thousand steps). While this agent performed the scheduling, the beam selection procedure adopted was the same from the oracle baseline, in other words, after the agent scheduled a user, it was always served with the optimal beam.

From the results presented in Figure 6.3, it is possible to see that the agent is capable of very quickly learning a scheduling approach that surpasses the round-robin, hence, the beam selection procedure is the one that must receive the focus in efforts to improve the agent design.

## 6.2 Discussion

The results presented in Section 6.1 showed that the current realization of CAVIAR is capable of providing an environment to train and test RL agents for the joint problem of user scheduling and beam selection. However, although generalizing well for user scheduling, it still



**Figure 6.3:** Cumulative reward of the scheduling agent and baseline approaches over 25 test episodes.

lacks in performance for the beam selection task.

The literature shows that the problem of joint scheduling and beam selection is a challenging one and demands sophisticated modeling in order to obtain outstanding results [57, 58]. More specifically, the literature about beam selection also relies on the use of strategies such as *top-k* classification, where the objective goes from "choosing the singular best beam to serve a given user", to instead "choosing a subset that has a high probability of containing the optimal choice" [4, 31].

Besides that, RL is still a widely investigated tool in the context of 5G and future networks, however, it still has downsides that need to be addressed in order for it to achieve the same success of DL on a wide variety of problems. Some of the obstacles that hinder RL are the impact that the framing of the problem and the reward engineering [59] have on the agent's capacity to converge, also the low data efficiency, where even state-of-the-art methods commonly need to execute millions of steps in order to converge to a good policy [26].

# Chapter 7

## Conclusion and Future Works

This work presented a simulation methodology named CAVIAR, as a methodology suitable to generate environments with varying reality gap and computational complexity, to train and test RL agents for the problems related to the PHY of the 5G networks. Using a realization of it that served as the base for the PS-006 in the 2021 ITU AI/ML in 5G challenge, it was provided statistics of an experiment in which an RL agent faces the problems of user scheduling and beam selection. The experiments allowed to validate the designed environment for RL training and testing. The work presented here also participated in the 2021 ITU Kaleidoscope conference in the video and full-paper tracks, achieving the third place (ex aequo) in the best paper award.<sup>1</sup>

Future developments can focus on two different efforts: the first one is to explore the improvement of each individual subsystem, especially the communications subsystem, as it concentrates several of the simulation core functionalities and is also the main responsible for increasing the computational needs. The second effort is to investigate the use of the same set of tools used in the development of the simulation, to also enable the use of digital twin approaches in research topics that involve the need to have knowledge about the system deployment site. In the next subsection, the improvement on the subsystems is discussed in more depth in hopes to be useful to any researcher intending to continue the efforts.

---

<sup>1</sup><https://www.itu.int/en/ITU-T/academia/kaleidoscope/2021/Pages/default.aspx>

## 7.1 Improvements on the Subsystems

Concerning the subsystems improvements, studies must be conducted on how to closely integrate the artificial intelligence inside the simulation loop, paying close attention to the trade-off where one must loosen up the realism of the simulated elements and phenomena in order to maintain the real-time aspect of the simulation. Focusing, on the communications subsystem, there are three main improvements to be considered. Without order of importance, they are: the scalability of the number of users in the simulation, the packet traffic simulation, and the channel models used.

Regarding the scalability issues that arise from increasing users in a scheduling scenario, one could explore studies such as the one that focus on the concept of *ego vehicles* to tackle the problems related to the simulation of V2X systems and the need to run large-scale simulations to develop and test solutions for that area [7]. This work shows that only a small number of vehicles, around 60, were needed to perform a complete simulation from the perspective of the ego vehicle in a typical urban scenario, while keeping the its completeness and maintaining the experimentation close to a full, non-real-time simulation.

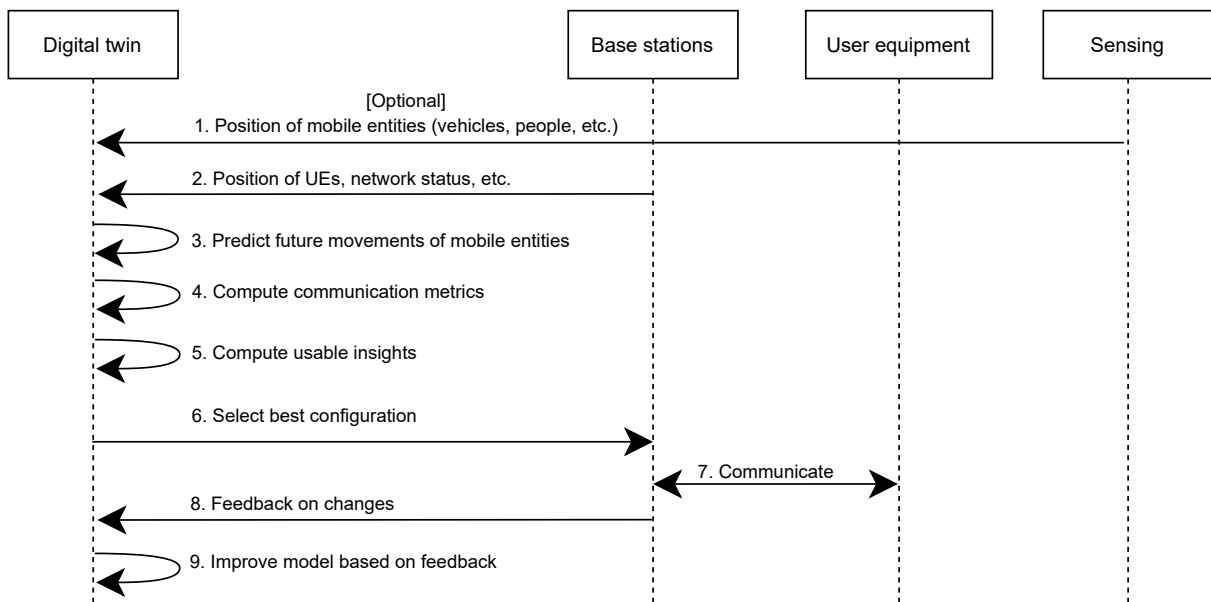
About the packet traffic simulation, software such as ns-3 or OMNeT++ are event-based network simulators, and need special attention when there is a need to integrate them with a more dynamic simulation loop. In Section 3.2, the LIMoSim [6] and Veneris[5] were briefly described, with both being hybrid simulators that use ns-3 and OMNeT++, respectively, to deal with the network packet traffic. Currently, the CAVIAR environment is being extended with the addition of the ns-3 simulator, due to the module 5G-LENA [60], that enables support for the New Radio (NR) Access Network. In this effort, it is recommended to study the methods adopted by these hybrid simulators to orchestrate the event-based simulators with the continuous ones, to enable more realistic experiments, while maintaining performance.

Now, with respect to the channel models, the current system could be integrated with ray tracing simulators via orchestration, similar to [4] or [5], however, the real-time aspect could become a bottleneck of such integration. Other studies, for example, the work in [61], could also become an alternative to be explored. It focuses on using already existent capabilities of the rendering plataform, to propose a path loss model called Light Intensity Model (LIM), based on the use of light physics in the Unity3D game engine, with its performance surpassing a 3rd Generation Partnership Project (3GPP) [62] based and a New York University (NYU) [63] path loss model.



## 7.2 Using the Realistic Simulation Tools Described in this Work to Develop Simulations that Explore Concepts Related to Digital Twins

As the simulation grows in realism, it also increases the possibility of generating a digital twin of the deployment site. In this section, the objective is to propose a possible digital twin approach, built upon the the same set of tools used throughout this work. Figure 7.1 shows the main steps required to achieve the proposed solution. This digital twin system must be capable of computing physical layer aspects of the communication environment and use them to improve the system performance. Here, it is presented a quick summary of steps to perform these tasks, followed by examples of what each step might mean or be used for.



**Figure 7.1:** Overall messages needed by the proposed digital twin solution, together with the main processing steps.

First, the digital twin must receive information about the real world environment, in order to compose a virtual scenario:

1. the digital twin receives the position of the mobile entities to place them into the scene. They could be any object that influences propagation (scatterers). It gathers this data from a sensing component, that might be a base station-mounted LiDAR, GPS, or any other sensor;

2. the digital twin also receives UE information, so that their properties are set and they are placed inside the scene. The information can include their buffer status, their type, etc. With this data, the digital twin knows, for instance, which channels are important to compute and what might cause interference;

Given the environment built inside the virtual world subsystem, the AI/ML models interact with the twin in the next steps to find what could be improved in the system.

3. the models uses the current position of the users and the scatterers, as well as their possible trajectories, to predict future situations, such as blocking, that would affect the communication channel;
4. the models configures parameters that are to be evaluated in the twin. The changes that they apply to the environment might include different beam allocation and different hypothesis of their utilization, which affects interference and other system behavior.
5. the digital twin generates a virtual environment implementing the changes and computes the communication metrics, such as the channel between UEs and base stations;
6. the models evaluates the impact that the tested parameters had in the virtual environment and either proceeds to compute an optimal configuration or loops back to step 4 to evaluate additional parameters.

With such simulations in hand, the AI/ML models can compute hints that help with the communication aspects on-the-fly, without human intervention, and can also indicate insights for operators to optimize the network. One example of benefit that could be achieved by such strategy is in the problem of beam selection, investigated in this work, where the traditional process of beam sweeping could be executed inside the virtual environment generated in step 5, such that it reduces the overhead in spectrum utilization. Not only that, but, by acquiring and storing the data from the real environment, the digital twin can also help in the generation of data to be fed to AI/ML models, to evaluate a hypothesis before applying any actuation in the real network.

So, once the model is satisfied with the data it got from the digital twin, it now applies the configuration changes into the real network:

7. the model applies the computed configuration, introduced by step 6, into the network;

8. this step is the proper operation of the network, where it functions as usual and communicates with the UEs.

Finally, the network metrics are used by the models and the twin.

9. the base stations report the experienced performance and communication metrics, so the models can use the feedback to understand if the desired effect has been achieved and also to improve its knowledge. This feedback might include the achieved power in a transmission, for example. Similarly, the models also report the communication metrics to the twin, so that it can learn by experience.

# Bibliography

- [1] I. Nascimento, R. Souza, S. Lins, A. Silva, and A. Klautau, “Deep Reinforcement Learning Applied to Congestion Control in Fronthaul Networks,” *Proceedings - 2019 IEEE Latin-American Conference on Communications, LATINCOM 2019*, pp. 1–6, 2019.
- [2] Y. Kim and H. Lim, “Multi-Agent Reinforcement Learning-Based Resource Management for End-to-End Network Slicing,” *IEEE Access*, vol. 9, pp. 56 178–56 190, 2021.
- [3] X. Wang and T. Zhang, “Reinforcement Learning Based Resource Allocation for Network Slicing in 5G C-RAN,” in *2019 Computing, Communications and IoT Applications (ComComAp)*, 2019, pp. 106–111.
- [4] A. Klautau, P. Batista, N. González-Prelcic, Y. Wang, and R. W. Heath, “5G MIMO data for machine learning: Application to beam-selection using deep learning,” in *2018 Information Theory and Applications Workshop (ITA)*. IEEE, 2018, pp. 1–9.
- [5] E. Egea-Lopez, F. Losilla, J. Pascual-Garcia, and J. M. Molina-Garcia-Pardo, “Vehicular Networks Simulation with Realistic Physics,” *IEEE Access*, vol. 7, pp. 44 021–44 036, 2019.
- [6] Sliwa, Benjamin and Patchou, Manuel and Heimann, Karsten and Wietfeld, Christian, “Simulating Hybrid Aerial- and Ground-based Vehicular Networks with ns-3 and LIMoSim,” in *Proceedings of the 2020 Workshop on Ns-3*, 2020, pp. 1–8.
- [7] Buse, Dominik S. and Dressler, Falko, “Towards real-time interactive V2X simulation,” in *2019 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2019, pp. 1–8.
- [8] A. Klautau, A. de Oliveira, I. P. Trindade, and W. Alves, “Generating MIMO Channels For 6G Virtual Worlds Using Ray-tracing Simulations,” *arXiv preprint arXiv:2106.05377*, 2021.

- [9] A. Alkhateeb, “DeepMIMO: A generic deep learning dataset for millimeter wave and massive MIMO applications,” in *Proc. of Information Theory and Applications Workshop (ITA)*, San Diego, CA, Feb 2019, pp. 1–8.
- [10] K. K. Patel, “A Simulation Environment with Reduced Reality Gap for Testing Autonomous Vehicles,” Ph.D. dissertation, University of Windsor (Canada), 2020.
- [11] A. Amini, I. Gilitschenski, J. Phillips, J. Moseyko, R. Banerjee, S. Karaman, and D. Rus, “Learning Robust Control Policies for End-to-End Autonomous Driving From Data-Driven Simulation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1143–1150, 2020.
- [12] A. Klautau, N. González-Prelcic, and R. W. Heath, “LIDAR data for deep learning-based mmWave beam-selection,” *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 909–912, 2019.
- [13] W. Jiang, B. Han, M. A. Habibi, and H. D. Schotten, “The road towards 6G: A comprehensive survey,” *IEEE Open Journal of the Communications Society*, vol. 2, pp. 334–366, 2021.
- [14] A. Oliveira, F. Bastos, I. Trindade, W. Frazão, A. Nascimento, D. Gomes, F. Müller, and A. Klautau, “Simulation of Machine Learning-Based 6G Systems in Virtual Worlds,” *Submitted to ITU Journal on Future and Evolving Technologies*, 2021.
- [15] Shital Shah and Debadeepta Dey and Chris Lovett and Ashish Kapoor, “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles,” in *Field and Service Robotics*, 2017. [Online]. Available: <https://arxiv.org/abs/1705.05065>
- [16] “The most powerful real-time 3D creation tool,” <https://www.unrealengine.com>, accessed: 2021-10-19.
- [17] Chevalier-Boisvert, Maxime and Willems, Lucas and Pal, Suman, “Minimalistic Grid-world Environment for OpenAI Gym,” <https://github.com/maximecb/gym-minigrid>, 2018.
- [18] Heath, Robert W. , et al., “An Overview of Signal Processing Techniques for Millimeter Wave MIMO Systems,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 3, pp. 436–453, 2016.

- [19] F. W. Vook, A. Ghosh, and T. A. Thomas, “MIMO and Beamforming Solutions for 5G Technology,” in *2014 IEEE MTT-S International Microwave Symposium (IMS2014)*. IEEE, 2014, pp. 1–4.
- [20] L. Zhang, H. Zhao, S. Hou, Z. Zhao, H. Xu, X. Wu, Q. Wu, and R. Zhang, “A survey on 5G millimeter wave communications for UAV-assisted wireless networks,” *IEEE Access*, vol. 7, pp. 117 460–117 504, 2019.
- [21] M. Giordani, M. Mezzavilla, C. N. Barati, S. Rangan, and M. Zorzi, “Comparative Analysis of Initial Access Techniques in 5G mmWave Cellular Networks,” in *2016 Annual Conference on Information Science and Systems (CISS)*. IEEE, 2016, pp. 268–273.
- [22] J. Choi, V. Va, N. Gonzalez-Prelcic, R. Daniels, C. R. Bhat, and R. W. Heath, “Millimeter-wave Vehicular Communication to Support Massive Automotive Sensing,” *IEEE Communications Magazine*, vol. 54, no. 12, pp. 160–167, 2016.
- [23] J. Kim and A. F. Molisch, “Fast Millimeter-wave Beam Training with Receive Beamforming,” *Journal of Communications and Networks*, vol. 16, no. 5, pp. 512–522, 2014.
- [24] P. Zhou, X. Fang, Y. Fang, Y. Long, R. He, and X. Han, “Enhanced Random Access and Beam Training for Millimeter Wave Wireless Local Networks with High User Density,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 12, pp. 7760–7773, 2017.
- [25] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2020.
- [26] Schulman, John and Wolski, Filip and Dhariwal, Prafulla and Radford, Alec and Klimov, Oleg, “Proximal Policy Optimization Algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [27] Schulman, John and Levine, Sergey and Abbeel, Pieter and Jordan, Michael and Moritz, Philipp, “Trust Region Policy Optimization,” in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.
- [28] MacCartney, George R and Yan, Hangsong and Sun, Shu and Rappaport, Theodore S, “A Flexible Wideband Millimeter-wave Channel Sounder with Local Area and NLOS to LOS Transition Measurements,” in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

- [29] A. Taha, Y. Zhang, F. B. Mismar, and A. Alkhateeb, “Deep reinforcement learning for intelligent reflecting surfaces: Towards standalone operation,” in *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2020, pp. 1–5.
- [30] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, “Microscopic traffic simulation using SUMO,” in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: <https://elib.dlr.de/124092/>
- [31] M. Dias, A. Klautau, N. González-Prelcic, and R. W. Heath, “Position and LIDAR-aided mmWave beam selection using deep learning,” in *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2019, pp. 1–5.
- [32] Alrabeiah, Muhammad and Hredzak, Andrew and Liu, Zhenhao and Alkhateeb, Ahmed, “Viwi: A Deep Learning Dataset Framework for Vision-Aided Wireless Communications,” in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–5.
- [33] “Unity Engine,” <https://unity3d.com>, accessed: 2021-12-12.
- [34] Bewley, Alex and Rigley, Jessica and Liu, Yuxuan and Hawke, Jeffrey and Shen, Richard and Lam, Vinh-Dieu and Kendall, Alex, “Learning to Drive from Simulation without Real World Labels,” in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 4818–4824.
- [35] W. Sun, S. Lei, L. Wang, Z. Liu, and Y. Zhang, “Adaptive Federated Learning and Digital Twin for Industrial Internet of Things,” *IEEE Transactions on Industrial Informatics*, vol. 3203, no. c, pp. 1–1, 2020.
- [36] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, “Digital Twin in Manufacturing: A Categorical Literature Review and Classification,” *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.
- [37] B. R. Barricelli, E. Casiraghi, and D. Fogli, “A survey on digital twin: Definitions, characteristics, applications, and design implications,” 2019.

- [38] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, “Digital Twin in Industry: State-of-the-Art,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, apr 2019.
- [39] Öhlén, Peter, “The future of digital twins: what will they mean for mobile networks?” <https://www.ericsson.com/en/blog/2021/7/future-digital-twins-in-mobile-networks>, 2021.
- [40] Lindbo, Dag and Ceballos, German and Lundevall, Magnus , “Next-generation simulation technology to accelerate the 5G journey,” <https://www.ericsson.com/en/blog/2021/4/5g-simulation-omniverse-platform>, 2021.
- [41] “Intelligent Site Engineering: A new era for the deployment and maintenance of network sites,” <https://www.ericsson.com/49bb60/assets/local/network-services/doc/ericsson-intelligent-site-engineering-paper.pdf>, 2021.
- [42] Puleri, Marzio and Pepe, Teresa and Bottari, Giulio and Sabella, Roberto, “A look at digital twins in port operations,” <https://www.ericsson.com/en/blog/2020/12/digital-twins-port-operations>, 2020.
- [43] “Azure Digital Twins: use IoT spatial intelligence to create models of physical environments,” <https://azure.microsoft.com/en-us/services/digital-twins/>, Accessed: 2021-12-18.
- [44] “What is a digital twin?” <https://www.ibm.com/topics/what-is-a-digital-twin>, Accessed: 2021-12-18.
- [45] “Electrical Digital Twin,” <https://new.siemens.com/global/en/products/energy/energy-automation-and-smart-grid/electrical-digital-twin.html>, Accessed: 2021-12-18.
- [46] “The ‘Digital Twin’ in Hardware in the Loop (HiL) Simulation: A Conceptual Primer,” <https://www.opal-rt.com/the-digital-twin-in-hardware-in-the-loop-hil-simulation-a-conceptual-primer/>, Accessed: 2021-12-18.
- [47] “Introducing A New Digital Twin Scene - Paris, France!” <https://www.cognata.com/introducing-a-new-digital-twin-scene-paris-france/>, Accessed: 2021-12-18.
- [48] “Eclipse Ditto,” <https://github.com/eclipse/ditto>, Accessed: 2021-12-18.



- [49] “iTwin.js: a starter kit for developing web applications for infrastructure digital twins.” <https://www.itwinjs.org/>, Accessed: 2021-12-18.
- [50] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, “Deep reinforcement learning for stochastic computation offloading in digital twin networks,” *arXiv*, vol. 3203, no. c, pp. 1–10, 2020.
- [51] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, “Communication-Efficient Federated Learning and Permissioned Blockchain for Digital Twin Edge Networks,” *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 1–1, 2020.
- [52] —, “Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks,” *arXiv*, vol. 3203, no. c, pp. 1–10, 2020.
- [53] R. Dong, C. She, W. Hardjawana, Y. Li, and B. Vucetic, “Deep Learning for Hybrid 5G Services in Mobile Edge Computing Systems: Learn from a Digital Twin,” *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4692–4707, 2019.
- [54] R. W. Heath, N. González-Prelcic, S. Rangan, W. Roh, and A. M. Sayeed, “An Overview of Signal Processing Techniques for Millimeter Wave MIMO Systems,” vol. 10, no. 3, pp. 436–453, Apr. 2016.
- [55] Balanis, Constantine A, *Antenna theory: analysis and design*. John wiley & sons, 2015.
- [56] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, “Stable baselines3,” <https://github.com/DLR-RM/stable-baselines3>, 2019.
- [57] He, Shiwen and Wu, Yongpeng and Ng, Derrick Wing Kwan and Huang, Yongming, “Joint optimization of analog beam and user scheduling for millimeter wave communications,” *IEEE Communications Letters*, vol. 21, no. 12, pp. 2638–2641, 2017.
- [58] Sun, Shu and Li, Xiaofeng, “Deep-Reinforcement-Learning-Based Scheduling with Contiguous Resource Allocation for Next-Generation Wireless Systems,” *arXiv preprint arXiv:2010.11269*, 2020.
- [59] Singh, Avi and Yang, Larry and Hartikainen, Kristian and Finn, Chelsea and Levine, Sergey, “End-to-End Robotic Reinforcement Learning without Reward Engineering,” *arXiv preprint arXiv:1904.07854*, 2019.

- [60] Patriciello, Natale and Lagen, Sandra and Bojovic, Biljana and Giupponi, Lorenza, “An E2E simulator for 5G NR networks,” *Simulation Modelling Practice and Theory*, vol. 96, p. 101933, 2019.
- [61] Inca, Saúl and Prado, Danaisy and Martín-Sacristán, David and Monserrat, Jose F., “Channel Modelling based on Game Engines Light Physics for mmW in Indoor Scenarios,” in *2020 14th European Conference on Antennas and Propagation (EuCAP)*. IEEE, 2020, pp. 1–5.
- [62] 3GPP TSG RAN, “Study on channel model for frequencies from 0.5 to 100 GHz,” 3GPP, TR 38.901 v14.0.0, May 2017.
- [63] Maccartney, George R. and Rappaport, Theodore S., et al., “Indoor Office Wideband Millimeter-Wave Propagation Measurements and Channel Models at 28 and 73 GHz for Ultra-Dense 5G Wireless Networks,” *IEEE Access*, vol. 3, pp. 2388–2424, 2015.



# **Appendices**

# Appendix A

## Toy Environment for MIMO Beam Selection

During the development of this work, one of the secondary contributions was the development of a toy environment for beam selection based on a minigrid format, that can be used as an educational material for the use of RL approaches for beam selection. In this appendix, first, the MiniGrid environment will be briefly described and then posed as a beam selection problem, in the called MIMO MiniGrid.<sup>1</sup>

### A.1 MiniGrid Environment

The Minimalistic GridWorld Environment (MiniGrid) [17] is based on the OpenAI Gym environment and designed to represent the classic gridworld scenario. Originally, the state space is represented by the position of the RL agent in the grid and the action space represents the 4 directions to which the agent can move inside the grid. Each movement made by the agent is rewarded with 0 or a negative value depending on how it is implemented. When the agent finds the final stage it is rewarded with a maximum value, this value is decreased by the number of steps the agent takes to find the terminal stage. The episode ends when the final stage is encountered.

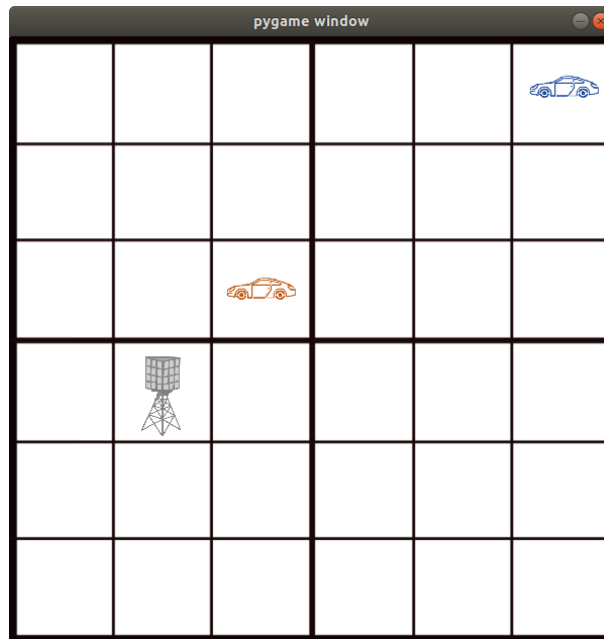
The MiniGrid platform brings several variations of the original gridworld, adding different levels of difficulty. There are scenarios with adjustable size, different number of rooms, obstacle and elements. This environment is used as a basis for the beam selection version, due

---

<sup>1</sup>[https://github.com/lasseufpa/ml4comm-icc21/tree/main/mimo\\_rl](https://github.com/lasseufpa/ml4comm-icc21/tree/main/mimo_rl)

to its robust implementation of the simple MiniGrid and easiness of being extended for other purposes via integration with different packages.

## A.2 MIMO Beam Selection MiniGrid Environment



**Figure A.1:** Example of the proposed MIMO beam selection environment.

As illustrated by Figure A.1, the MiniGrid environment can be used for beam selection problems in which the BS and UEs exist in a  $M \times M$  gridworld, with  $M^2$  invariant channels depending only on position. We can consider the RL agent being executed at a BS serving single-antenna users on downlink, with an analog MIMO antenna array, using an architecture with  $N_b$  beam vector indices. The BS RL agent actions consist in scheduling one of the  $N_u$  users and choosing one among the  $N_b$  beams to serve it, creating an action space with  $N_u \times N_b$  dimension.

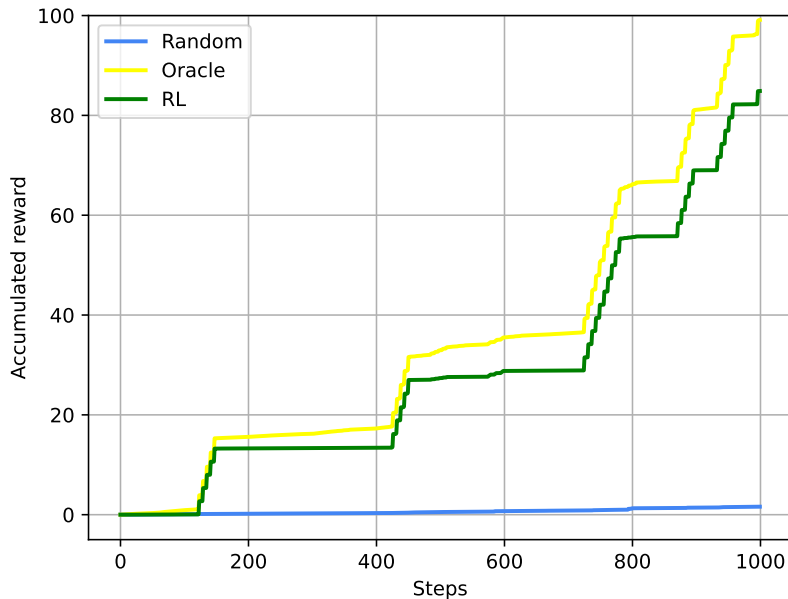
An episode lasts  $N_s$  time slots, where  $N_s$  represents the number of time steps (also called *scenes*) that compose the given episode. At each episode, the UEs moves left, right, up or down. The state to be observed is defined as the users' positions and a list of  $N_a - 1$  indices of the previously scheduled users, where  $N_a$  is equal to 3, and represents a memory that stores the last  $N_a$  actions already taken. All in a state space dimension of  $M^{2N_u} \times (N_u)^{N_a-1}$ , where  $N_u$  is the number of users.

To conclude, the reward at the end of the episode is the normalized magnitude of the combined channel given the chosen beam. A penalty of  $-0.5$  is added if a user is not allocated for 10 consecutive slots. The final return is the sum of the rewards.

### A.3 MIMO MiniGrid environment results

For the MIMO MiniGrid environment two brief experiments were conducted, where first only one user, and then two users, roamed inside a  $6 \times 6$  grid world with no additional obstacles to hinder their movement. The MIMO system used has several similarities with the one described with more details in Section 5.2. The main differences are that it used a Uniform Linear Array (ULA) with  $N_t = 32$  antenna elements at the BS.

Both experiments uses PPO agents with default parameters and actor and critic networks defined with two hidden layers of 128 neurons each, with *tanh* activation functions. In the first scenario, the environment is used for beam selection only, having a single user be served.



**Figure A.2:** Performance of a PPO agent in the task of beam selection inside the MIMO MiniGrid environment.

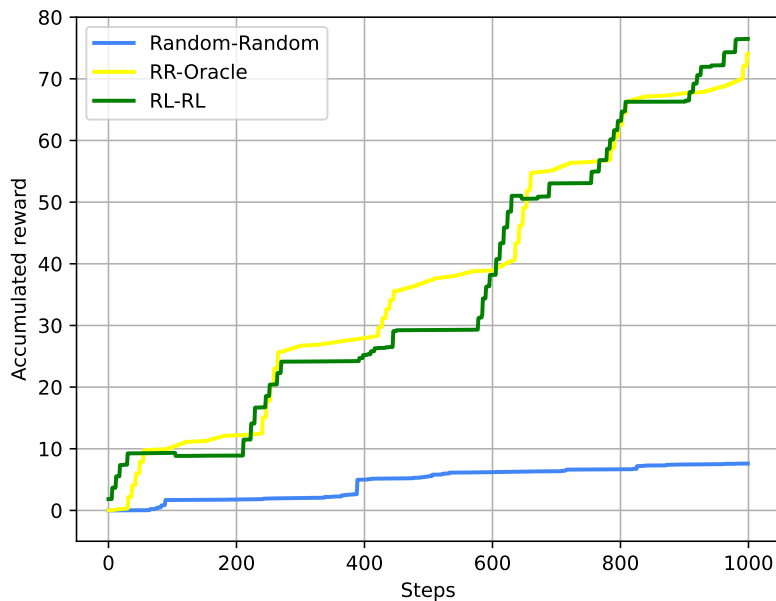
Figure A.2 and Table A.1 shows that the performance of the agent, in terms of accumulated reward throughout 1000 simulation steps, achieves around 85% of the optimal performance, represented by the *oracle*, which, similarly with the baseline approach presented in

**Table A.1:** Comparison of the methods performance on 1000 steps on a scenario with one user

Method	Accumulated reward	Mean reward per step	Reward variance per step
Oracle	99.115	0.099	0.203
RL	84.892	0.084	0.204
Random	1.602	0.001	5.348

Section 5.6, finds the best beam index by exhaustive search.

In a second experiment, the number of users was raised to 2 and, this time, the PPO agent was able to surpass the performance of the baseline that implements round-robin for user scheduling and serve using the optimal choice of beam (RR-Oracle), achieving the performance shown in Figure A.3 and Table A.2.



**Figure A.3:** Performance of a PPO agent in a slightly more complex task, with the introduction of one additional user to the dynamic of the MIMO MiniGrid environment.

These simple experiments are meant to showcase the basic functionality and act as a proof-of-concept for the use of a MiniGrid-based toy environment for the problems related to the 5G PHY.



**Table A.2:** Comparison of the methods performance on 1000 steps on a scenario with two users

<b>Method</b>	<b>Accumulated reward</b>	<b>Mean reward per step</b>	<b>Reward variance per step</b>
RR-Oracle	74.102	0.074	0.113
RL-RL	76.466	0.076	0.143
Random-Random	7.598	0.007	0.006