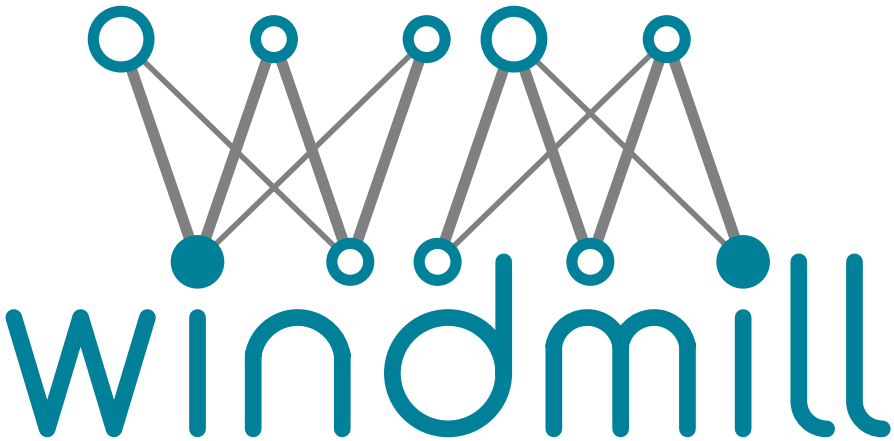
Marie Skłodowska Curie Action

WINDMILL

Machine Learning for Wireless Communications

H2020-MSCA-ITN-ETN

Grant Agreement Number: 813999



WP6–System-wide Cognitive Optimisation Schemes

D6.2–hierarchical and distributed learning architecture and multi-objective optimisation strategies

Contractual Delivery Date:	
Actual Delivery Date:	August 30, 2021
Responsible Beneficiary:	UNIPD
Contributing Beneficiaries:	AALTO, BOSCH
Dissemination Level:	Public
Version:	1.0

PROPRIETARY RIGHTS STATEMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813999.

Document Information

Document ID:	WP6/D6.2
Version Date:	August 30, 2021
Total Number of Pages:	58
Abstract:	
Keywords:	

Authors

Full name	**Beneficiary/ Organisation**	**e-mail**	**Role**
Salman Mohebi	UNIPD	salman.mohebiganjabadi @unipd.it	Contributor
Anay Ajit Deshpande	UNIPD	anayajit.deshpande @unipd.it	Contributor
Dariush Salami	AALTO	dariush.salami@aalto.fi	Contributor
Pedro Maia de Sant Ana	Bosch	pedro.maiadesantana @de.bosch.com	Contributor
Zecchin Matteo	EURECOM	zecchin@eurecom.fr	Contributor

Reviewers

Andrea Zanella	UNIPD	andrea.zanella @unipd.it	Supervisor

Version history

Version	Date	Comments

Table of Contents

List of Figures

List of Acronyms

ADC Analog to Digital Converter

AHC Agglomerative Hierarchical Clustering

AoI Age of Information

AoL Age of Loop

BS Base Station

CFAR False Alarm Rate Detection

CSI Channel State Information

DQN Deep Q-Network

DRL Deep Reinforcement Learning

EM Electromagnectic

eMBB Enhanced Mobile Broadband

FFT Fast Fourier Transform

FMCW Frequency-Modulated Continuous-Wave

FPS Farthest-Point Sampling

H-DDPG Hierarchical Deep Deterministic Policy Gradient

h-DQN hierarchical-Deep Q-Network

HAM Hierarchical Abstract Machine

HRL Hierarchical Reinforcement Learning

IF Intermediate Frequency

KPI Key Performance Indicator

LDPL Long-normal Distance Path Loss

LoS Line-of-Sight

LSTM Long Short-Term Memory

MDP Markov Decision Process

MLP Multi-Layer Perceptron

mMTC Massive Machine Type Communications

mmWave Millimeter Wave

NN Neural Network

OFDM Orthogonal Frequency Division Multiplexing

QoS Quality of Service

RF Radio Frequency

RIS Reconfigurable Intelligent Surface

RL Reinforcement Learning

RSSI Received Signal Strength Indicator

SCA Successive Convex Approximation

SINR Signal-to-Interference-plus-Noise Ratio

SMDP Semi-Markov Decision Process

UAV Unmanned Aerial Vehicle

UE User Equipment

UPA Uniform Planar Array

URLLC Ultra-Reliable Low Latency Communication

WNCS Wireless Networked Control System

XL-MIMO extra large scale massive MIMO

1. Introduction and Motivation

The modern 5G and beyond-5G wireless systems are supposed to support various classes of applications with very different and potentially conflicting service requirements, from Ultra-Reliable Low Latency Communication (URLLC) to Enhanced Mobile Broadband (eMBB), and Massive Machine Type Communications (mMTC). The increased complexity of these networks, which is caused by the enormous number of connected devices and various applications with different Quality of Service (QoS) requirements, hinders the traditional methods to efficiently and adequately operate. In the past few years, data-driven and machine learning-based techniques have attracted a lot of attention in both industry and academia as potential solutions to manage wireless networks. Recently, a massive amount of research has investigated the feasibility of machine learning techniques in solving some critical problems in wireless communications, e.g., resource management, network access, rate control, and security [3–5].

Scalability is one of the main issues in machine learning techniques, as centralized approaches may limit the scaling of the system in terms of number of connected devices and significantly reduce the system's performance. On the other hand, different Key Performance Indicators (KPIs) could be difficult to satisfy, since improving one may deteriorate others. These problems could possibly be addressed by employing distributed/hierarchical solutions or defining multi-objective optimization problems and using machine learning to solve them. In a hierarchical/distributed approach, the original problem is split into multiple simpler sub-problems, and each task is assigned to a different agent or solver that operates in different levels of abstraction.

In this deliverable, we first provide an overview of the different architectures of hierarchical learning as well as decentralized and distributed machine learning. Then we took a look at multi-objective optimization problems and discuss some use-cases and applications of these techniques in wireless communications. More specifically, in Chapter 2 we overview different Hierarchical Reinforcement Learning (HRL) techniques where the original problem is divided into multiple layers of hierarchies, and different agents or solvers solve problems in different abstract layers. As discussed, these techniques can effectively simplify the reinforcement learning agent's architecture, increase the inter-agent knowledge sharing and reduce the state and action space. We have also investigated some of the applications of HRL in wireless communications.

In Chapter 3 we discuss the framework of decentralized machine learning, a key enabler for edge intelligence. Being abroad and active field, we focalized our exposition on the important notions of robustness of the solution and the available tools to reduce the amount of information to share in the network to converge to a solution. We also introduced Agnostic Decentralized Gradient Descent Ascent (AD-GDA), an algorithm that is shown to produce fair predictors using a compressed communication.

The remaining chapters provide some practical examples of application of hierarchical and/or distributed learning. More specifically in Chapter 4, we discussed different sensing techniques using radio frequency signals. We then introduced an Millimeter Wave (mmWave) radar-based gesture recognition system called Pantomime. Pantomime uses a hierarchical learning scheme to capture spatial features of radar-generated point clouds through a set of layers and then fuse the spatial features to capture the temporal evolution of the gesture.

In Chapter 5, we present a multi-objective minimization problem for Reconfigurable Intelligent Surface (RIS) assisted Unmanned Aerial Vehicle (UAV) cellular networks. The main motivation for the network design is to provide sustainable connectivity in semi-urban/rural areas or in emergency/disaster relief scenarios. On those grounds, an optimization problem is devised which determines the UAV trajectory, RIS phase, and UAV transmission power while ensuring a ground User Equipments (UEs) receive a guaranteed rate. To solve this joint optimization problem, we employed successive convex approximation, which iteratively optimizes the UAV trajectory, RIS phase, and UAV transmission power to converge to an optimal solution. The system is being further investigated to employ more dynamic network configurations and introduce data-driven optimization methodologies such as reinforcement learning.
In Chapter 6, we addressed multi-objective problem involving control and communication. The main idea is to find a suitable network configuration capable of optimizing the control performance while saving network resources. To achieve the proposed scheme, we elaborated on a new metric, age-of-loop, that encompasses the timing behavior of a closed-loop system, followed by a learning approach. The prospective concept was utilized as a basis for scientific publications and patent applications.
Hierarchical and distributed machine learning techniques are still an active field of research as they have great potential in solving some fundamental problems in wireless communications systems, as each device can operate as an autonomous and intelligent agent. This can increase the systems' reliability and improve security by reducing sensitive data exchange between devices and central decision-makers. Still, some serious issues and challenges need to be addressed before employing a fully decentralized approach in real-world operational systems. Finding an optimal solution in a fully decentralized fashion is not easy, as the solutions may even diverge in multi-objective optimization scenarios with multiple conflicting goals. So more advanced techniques need to be introduced to deal with this increased complexity.

2. Hierarchical Reinforcement Learning: An Overview

As the name implies, the HRL is a family of Reinforcement Learning (RL) algorithms where the problem is decomposed into different hierarchies of subproblems or subtasks, where the higher-level tasks invoke the primitive lower-level actions. The goal of HRL is to learn a multi-layer policy to perform control at different levels of temporal abstraction. Different from flat RL, the agent now can choose either macro-actions (consisting of sequences of lower-level actions), as well as primitive lower level actions. Due to the hierarchy and passing the control among the layers, the time between different macro-actions may not always be the same. Fortunately, Semi-Markov Decision Process (SMDP), a generalization of Markov Decision Process (MDP), can be applied to model a system with different elapsed time between actions. HRL can effectively reduce the computational complexity of the overall problem by breaking it down into several simple subproblems and independent learning and solving these small problems. The learning procedure also can be further accelerated by transferring the knowledge between different subtasks thanks to the generalization provided by HRL. On the downside, given the hierarchy constraints, in general, there is no guarantee that the decomposed solution provided by HRL is also an optimal solution to the original RL problem [6].

2.1. Hierarchical Reinforcement Learning

The general structure of a RL system is presented in Fig. 2.1(a), where an agent learns from interacting by the environment within an action-reward loop. At each time step, based on its observation from the environment, the agent takes action and receives new observation and reward for the action taken.

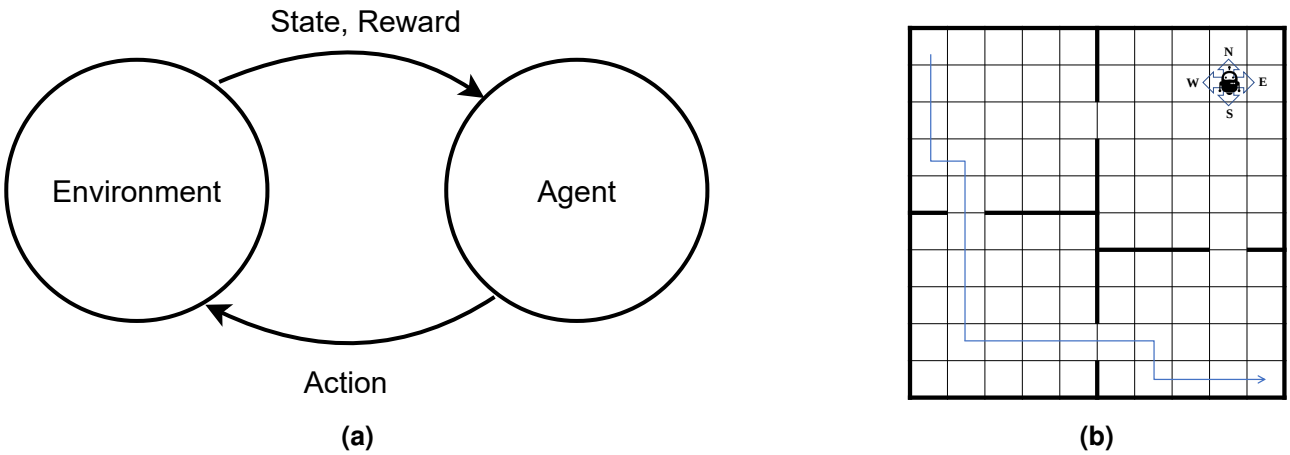


Fig. 2.1: (a) Action-reward loop in Reinforcement learning, (b) A simple environment which can be represented with two layers of hierarchy

Unlike the traditional RL, the HRL agent is provided with background knowledge about the decomposition of the environment which can be given explicitly or learned through interaction with the environment. As for traditional RL, the goal is to learn the policy that maximizes the cumulative future rewards.

To further discuss the motivation behind the problem decomposition in HRL, a simple environment is presented in Fig. 2.1(b). The figure represents a house with four rooms with a small gate between adjacent rooms. A 10×10 grid quantized the space in 100 cell. Each cell represents an individual state that the agent can be in. The agent always starts from the most top-left cell, and the goal is to reach the most bottom-right cell. The agent can sense the "state", i.e. the room that it occupies and its position in the room. In each step, the agent can take either one of the north, south, east, and west actions that move the state of the agent to the adjacent cells. The action that takes the agent to the wall leaves its location unchanged. The agent receives a -1 reward at each time step, and the goal is to reach the final state with minimum time.
Conventional flat RL needs to store 400 Q-values (100 states and four actions for each state) to solve this problem effectively. Another approach for solving this problem is to decompose the state space into two hierarchy levels: at the higher level, the agent learns which room to go to, while at the lower level, the agent learns to find the path to the gate of the room chosen by higher levels policies.
The agent's position inside each room is abstracted to the higher-level agents, as it looks at the room as a whole and aims to navigate between the rooms. Once the master agent learned the optimal policy, each "macro" action (choosing the next room to go to) will invoke and pass the control to the worker agent to move toward the room's gate. The actions of the master agent are temporally extended because once they are performed, they will persist until the agent leaves the current room and enters a new one. The worker agent will then pass the control back to the master to choose the next room. Here, the master agent requires only 8 Q-values (4 rooms and two actions, corresponding to the two exit gates from each room) to store. The worker agents also require to store the Q-values (100 for top-left room) for their own state-space or the room occupied. Assume a given higher-level policy chooses to enter the rooms, as shown in Fig. 2.1(b). In this case, the agents need to store 8, 100, 100, and 80 in total 288 Q-values, which is comparatively smaller than the values for flat RL agent. This gap even goes higher by increasing the scale of the problem. Also, having multiple similar sub-tasks will help the agents to accelerate the learning process by sharing the knowledge between worker agents.

2.1.0.1. Semi-Markov Decision Process

HRL problems can be mainly formulated by SMDP, which is a generalized version of MDP with temporally extended actions. So the transition probability of moving from state s to s' by taking action a is $P(s', \tau|s, a)$, where the τ is a random variable that represents the number of time steps that temporally extended actions take to complete. The expected reward function then can be expressed as:

$$R_{ss'}^{\tau,a} = \mathbb{E}\left\{\sum_{t=1}^{\tau} \gamma^{t-1} r_{t+\tau} | s_t = s, a_t = a, s_{t+\tau} = s'\right\} \tag{2.1}$$

where $\gamma \in [0, 1]$ is the discount factor.

2.2. Hierarchical Reinforcement Learning Models

In this section, we will review the most well-known architectures presented for the HRL problems. We will start with some classic models from the 1990s and then expand the discussion to more advanced methods that relays on deep neural networks.

2.2.1. The Options Framework

Introduced by Sutton *et. al.* in 1999, the option framework [7] is a well-known formulation for HRL that uses the SMDP to model the RL agent. The framework defines the concept of *option* as a generalization of actions that lets the agent choose between the macro and primitive actions. More formally an option is defined by a triple $t =< I_o, \pi_o, \beta_o >$, where I_o is the initiation set, $\pi_o : S \times A \rightarrow [0, 1]$ denotes the option's policy and $\beta_o : S \rightarrow [0, 1]$ defines the termination condition. By observing a new state, the agent checks whether it belongs to the initiation set I_o, in which case option o is started. The policy π_o, then, takes control and chooses the actions to be taken. At the end of each step, β_o is checked for termination condition, and if true, the control goes back to the global policy. Fig. 2.2 illustrates how actions and states with different temporal lengths could be managed in the options framework.

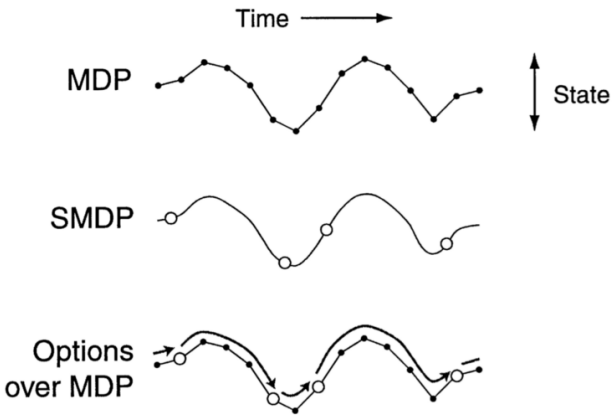


Fig. 2.2: State transitions in MDP vs. SMDP. The options let us mange temporally extended actions [7].

The algorithms following the Options framework, with both primitive actions and options, are proven to converge to the optimal policy.

2.2.2. Hierarchies of Abstract Machines

Hierarchical Abstract Machine (HAM) [8] defines the subtasks employing stochastic finite state automata named abstract machines, where each machine can call others as subroutines. HAM defines two types of machines: *action states*, that determine the action to be taken in a certain MDP state; and *choice states*, that select the next machine state with non deterministic actions.
More formally, an abstract machine can be defined by the triple $t =< \mu, I, \delta >$, where μ is the finite set of machine states, I denotes the stochastic function that maps the MDP states to

machine states, and δ represents the next-state function that receives the MDP and machine states and returns the next machine states.

2.2.3. Feudal Reinforcement Learning

Feudal reinforcement learning [9] defines a hierarchy of control, where a level of managers controls sub-managers and at the same time is controlled by higher-level super-managers or workers. Higher layer managers assign goals to the workers that are rewarded by performing actions and reaching the goals. Feudal learning is based on two concepts: *Information* and *Reward hiding*. *Information hiding* implies that managers observe the environment at different resolutions and only need to know the state of the system at the granularity of their own choice of tasks. *Reward hiding* means that the reward given to a sub-manager is based on satisfying the sub-goals assigned by the manager, which leads the sub-managers to learn a policy that satisfies the higher-level managers.

2.2.4. MAXQ

MAXQ [10] is another formulation for HRL that obtains a hierarchy of tasks by recursively decomposing the value function of original MDP into value functions of smaller constituent MDPs. Each smaller MDP essentially defines a subtask that can further decompose unless it is a terminal state.
The decomposition of a Q-values is done as

$$Q(p, s, a) = V(a, s) + C(p, s, a), \tag{2.2}$$

where $V(a, s)$ is the expected reward by taking action a in state s and $C(p, s, a)$ is the expected reward for the overall problem after the subtask p is terminated. Here, the action a can be a primitive action or a sequence of actions.

2.2.5. Feudal Networks

Inspired by Dayan's Feudal model (Section 2.2.3) for HRL, DeepMind introduced the FuUdal Networks [1] in 2015. They presented a modular architecture consisting of two independent Neural Networks (NNs) that operate as manager and worker, respectively.
The model is illustrated in Fig. 2.3, where both the manager and the worker receive a representation of the observation z. The manager creates a different latent state s_t with a higher dimension d and passes it to a recurrent NN (f^{Mrnn}) which defines the sub-goal g_t for the worker to achieve. On the other side, the worker receives the assigned sub-goals as well as the observation representation z and returns the optimal actions to take. The actions are taken in such a way that satisfies the sub-goals defined by the manager.

2.2.6. Option-Critic Networks

The Option-Critic architecture [2] is an extension to the Options framework (Section 2.2.1), where the options are not pre-defined and can be started from any state. The internal policies and the termination conditions can also be learned employing the gradient theorem as described in [11]. As presented in Fig. 2.4, two levels of policies are defined. The policy

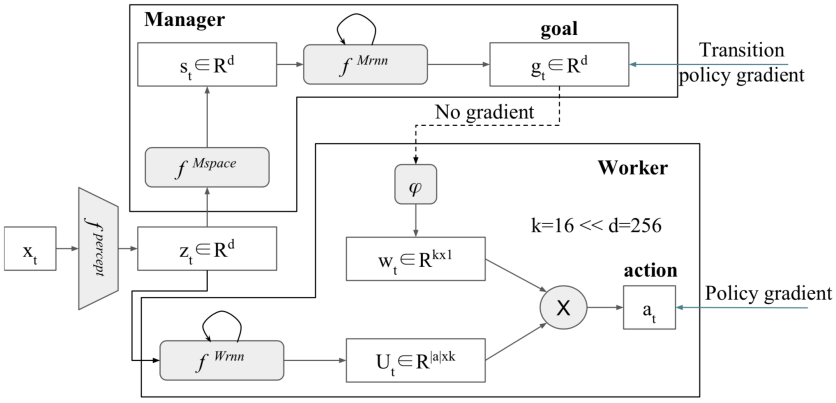


Fig. 2.3: Feudal networks [1].

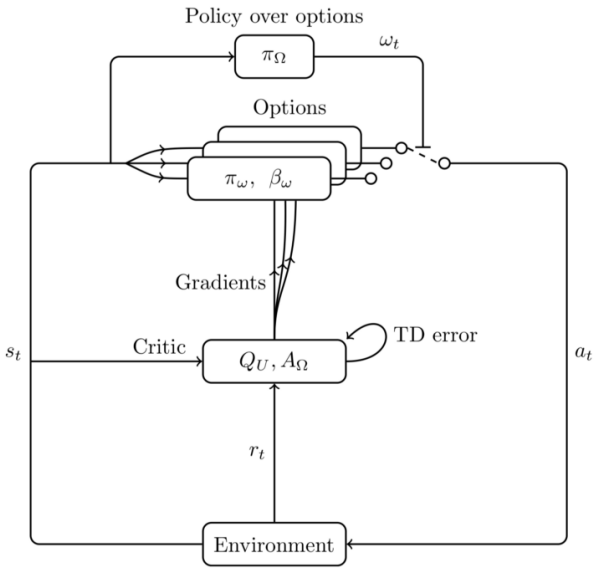


Fig. 2.4: Option-critic Networks [2].

over options chooses among options in each step and leaves the control to the sub-policy of the chosen option until it returns a terminated condition.
The Option-Critic architecture is based on actor-critic, where an *actor networks* tries different actions and a *critic network* that evaluates the performance of the chosen action and gives feedback to the actor. The *actor* consists of the local policy, termination functions, and the policy over options, while the *critic* keeps Q_U, the values of the actions taken in different state-option pairs, and A_Ω, the advantage function over options.

2.2.7. More recent models

HRL has been recently investigated in different papers. Some authors considered different aspects of HRL, e.g., sub-policy adaptation [12] and continuous action control [13], while others proposed different models employing actor-critic networks [14,15], stochastic NN [16], and model-free approaches [17].

The authors in [18] have presented HIRO, an HRL model with two layers of hierarchy. The higher-level policy defines the behavior of the lower-level policy by assigning the desired goal state and rewarding the lower-level policy upon reaching these states. The framework also adopts an off-policy correction to make the learning procedure more sample-efficient. Another HRL framework, presented in [19], is called Hierarchical Actor-Critic (HAC). Here the agent can simultaneously learn the policy of different layers. To avoid the instability issues of HRL resulted by jointly learning multiple levels of policies, HAC independently trains each level of hierarchy as if the lower levels are already optimal. Motivated from deep reinforcement learning, Kulkarni *et al.* have proposed hierarchical-Deep Q-Network (h-DQN) [20] which integrates hierarchical value function that operates at different temporal scales. The top-level value function learns a policy over intrinsic sub-goals or options, and the lower-level value function learns the policy over primitive actions to satisfy the goals determined by the top-level policy. A meta-learning approach for hierarchical policies is presented in [21] that improves the sample efficiency of unseen tasks by representing the shared information as a set of sub-policies. The authors of [22] have introduced the Modulated Policy Hierarchies (MPHs) that represent the environments with sparse rewards that can be decomposed into sub-tasks. The MPH can increase the training stability and performs better exploration thanks to rich modulation signals, temporal abstractions, and intrinsic motivation.

2.3. Hierarchical Reinforcement Learning for Wireless Communications

HRL has been recently employed to solve various problems in wireless communications. The authors in [23] have proposed a HRL-based algorithm to improve the energy efficiency in two-tier femtocell networks. Zou *et al.* [24] employed an optimization-driven HRL approach for hybrid relaying communications. In particular, they have proposed a Hierarchical Deep Deterministic Policy Gradient (H-DDPG) that integrated a model-based optimization with a conventional DDPG. The higher-level performs the relay mode selection (employing a Deep Q-Network (DQN)), and the lower-level optimizes the continuous beamforming and relays' operating parameters (by DDPG algorithm). HRL is also applied in [25] for joint relay selection and power allocation. The problem is decomposed into two hierarchical optimization objectives that are independently trained at different levels.
Liu *et al.* presented an h-DQN based dynamic multi-channel sensing approach for cognitive radio [26]. They divide the control into two layers: meta controller and sub-controller. The meta controller selects the target feature based on the action set while the sub-controller learns to approximate the target features selected by the first layer. In [27] an HRL approach is presented to solve the problem of frequency selection in the jamming environment without having prior knowledge about the jamming patterns and channel model. The problem is divided and solved into two steps: In the first step, the *band selection network* chooses the frequency bands which are passed to the *frequency selection network* that selects the specific frequency. Option-based Deep Reinforcement Learning (DRL) technique is presented in [28] to solve the backscattering data collection problems in multi UAV scenarios.

3. Communication-Efficient Distributionally Robust Decentralized Learning

Distributed learning is a key-enabler for intelligent decision making at scale as it allows to train statistical models by pooling computational resources from interconnected devices and by exploiting distributedly generated and stored data. The class of distributed training schemes contains a variety of optimization algorithms that differentiate by the reliability and capabilities of collaborating parties, the information these entities share, the way this information is aggregated and the type of communication constraints. In the following we will focus on the fully decentralized learning schemes that have gained an increasing level of attention, mainly due to their ability to harness, in a fault tolerant and privacy preserving manner, the large computational power and data availability at the network edge [29, 30]. In this framework, a set of interconnected nodes (smartphones, IoT devices, health centers, research labs, etc.) collaboratively train a machine learning model alternating between local model updates, based on in situ data, and peer-to-peer exchange of model-related information.

Compared to federated learning in which a swarm of devices communicates with a central parameter server at each communication round, fully decentralized learning has the benefits of removing the single point of failure and of alleviating the communication bottleneck inherent to the star topology. Nonetheless, as the decentralized optimization processes are envisioned to be carried out over wireless device-to-device networks, it is of prime interest reducing the amount of broadcasted information that is necessary to optimize the statistical model. Another fundamental challenge often arising in the distributed learning scenario comes from the heterogeneity of the dataset distribution at the collaborating devices. In such scenario, the solution yielded by decentralized optimization can often lead to unsatisfactory and unfair inference capabilities for certain sub-populations, and therefore devising algorithms that ensure fairness and robustness in this sense is necessary to guarantee worst case system performance. In light of these two shortcomings of decentralized optimization, after introducing decentralized stochastic gradient descent (D-SGD) (the workhorse of decentralized optimization) we will illustrate the recent progresses towards enchancing its communication efficiency and fairness.

3.1. Decentralized Optimization

The study of decentralized optimization was initiated in the 80s by the work of Tsitsiklis [31, 32] and it was motivated by the fact that many optimization problems admit a natural distributed implementation. For instance, empirical risk minimization learning rule - the bedrock of machine learning - can be decomposed leveraging its inherent data parallelism. Namely, given a model class parameterised by $\boldsymbol{\theta} \in \mathbb{R}^d$ and a dataset $\mathcal{D}$ the costumary learning objective is the minimization of the empirical estimate of the loss function $\ell : \theta \times z \to \mathbb{R}$ computed over the dataset $\mathcal{D}$:

$$f(\boldsymbol{\theta}) = \sum_{z \in \mathcal{D}} \ell(\boldsymbol{\theta}, z_i) \tag{3.1}$$

Algorithm 1: Decentralized-SGD

Input : Number of nodes m, number of iterations T, learning rates η_θ, connectivity matrix W , initial values $\theta^0 \in \mathbb{R}^d$

for t **in** $0, \dots T-1$ **do**

`// In parallel at each node` i

$\theta_i^{t+\frac{1}{2}} \leftarrow \theta_i^t - \eta_\theta \nabla_\theta g_i(\theta_i^t, \lambda_i^t, \xi_i^t)$ `// Descent Step`

send $\theta_i^{t+\frac{1}{2}}$ to $j \in \mathcal{N}(i)$ and receive $\theta_j^{t+\frac{1}{2}}$ from $j \in \mathcal{N}(i)$ `// Msgs exchange`

$\theta_i^{t+1} \leftarrow (1-\gamma)\theta_i^{t+\frac{1}{2}} + \gamma \sum_{j\in\mathcal{N}(i)} w_{i,j}\theta_j^{t+\frac{1}{2}}$ `// Averaging`

end

which, given a partition $\mathcal{D}_1, \dots, \mathcal{D}_m$ of the original dataset, can be decomposed in the sum of local loss terms as

$$f(\theta_1, \dots, \theta_m) = \sum_{i=1}^{m} g_i(\theta_i) = \sum_{i=1}^{m} \sum_{z_i \in \mathcal{D}_i} \ell(\theta_i, z_i); \tag{3.2}$$

$$\text{s.t.} \quad \theta_1 = \cdots = \theta_m; \tag{3.3}$$

where θ_i is the model estimate at the computing device i. Finally, by the linearity of the gradient operator, the first order information of the objective function $\nabla f(\theta)$ can be obtained by parallelizing the computation of $\nabla g_i(\theta)$ at each computing node and by then communicating and aggregating the local results. This simple strategy paves the way for first order distributed optimization over networks; however, it is important to notice that by distributing the computation, the network devices may store different estimates of the model parameters during the optimization process. For this reason, preliminary to the description of D-SGD - the prototypical distributed optimization procedure - it is necessary to introduce a strategy to satisfy the constraint (3.3).

Distributed average consensus is a a fundamental building block of many decentralized learning optimization algorithms. It provides a low complexity algorithm that allows a set of interconnected devices to agree on the average value of a quantity from an initial set of different local estimates. This procedure is staggeringly simple. Denoting by $\{\boldsymbol{x}_i^0\}_{i=1}^m$ the initial local estimates at m devices, the consensus protocol works by iteratively exchanging the current estimate $\{\boldsymbol{x}_i^t\}_{i=1}^m$ with their neighbors and then by averaging the received messages according to a gossip matrix $W \in \mathbb{R}^{m\times m}$, that we assume to be symmetric and doubly-stochastic. Denoting by $\mathcal{N}(i)$ the set of neighbors at node i and fixing a step size $\gamma \in (0, 1]$, the evolution of the local estimate follows the recursive relation

$$\boldsymbol{x}_i^t \leftarrow (1-\gamma)\boldsymbol{x}_i^{t-1} + \gamma \sum_{j\in\mathcal{N}(i)} w_{i,j}\boldsymbol{x}_j^{t-1} \tag{3.4}$$

and it is known to linearly converge to the mean vector $\bar{\boldsymbol{x}}$ with a rate proportional to the eigengap $\rho \in (0, 1]$ of the gossiping matrix W [33].

Decentralized stochastic gradient descent is an iterative procedure that leverages the convergence guarantees of the above scheme and alternates between local and parallel optimization of the parameter θ_i based on stochastic estimates of $\nabla \boldsymbol{g}_i$ with a consensus step.

The D-SGD is given in Algorithm 1 and it is provably convergent under mild assumptions on the loss function and the stochastic gradient. Despite its simplicity D-SGD has been shown to enjoy high adaptability to various network topologies, reliability to link failures, privacy-preserving capabilities, and potentially superior convergence properties compared to the centralized counterpart [29, 30, 34–36]. At the same time, with the intent of extending its applicability, a concurrent effort has been made to devise techniques able to reduce the delay due to inter-node communication.
D-SGD requires all nodes to broadcast the update estimate of the model parameters at each iteration of the algorithm. However, modern machine learning algorithms have a large number of trainable parameters that render the size of the payload too large in case of rate-constrained channels. For this reason, message compression techniques have been proposed to reduce the size of the exchanged messages [37–42]. Popular compression operators are:

- **Random Quantization** [40]: Given a vector $\boldsymbol{x} \in \mathbb{R}^d$, b-bit random quantization returns

$$\boldsymbol{x}_b = \frac{\text{sign}(\boldsymbol{x})\,\|\boldsymbol{x}\|}{2^b \tau} \left\lfloor 2^b \frac{|\boldsymbol{x}|}{\|\boldsymbol{x}\|} + \xi \right\rfloor \tag{3.5}$$

  where $\|\cdot\| : \mathbb{R}^d \to \mathbb{R}$ is the Euclidean norm, $|\cdot| : \mathbb{R}^d \to \mathbb{R}^d$ the element-wise absolute value operator, $\tau = 1 + \min\left\{ d/2^{2b}, \sqrt{d}/2^b \right\}$ and $\xi \sim \mathcal{U}[0,1]^{\otimes d}$. If the original precision for the entries of $\boldsymbol{x}$ is 32 bits, b-bit random quantization reduces the payload size from $32d$ bits to $(32 + bd)$ bits.

- **Sign Quantization** [41]: Given a vector $\boldsymbol{x} \in \mathbb{R}^d$, sign quantization returns

$$\boldsymbol{x}_{sign} = \frac{\|\boldsymbol{x}\|}{d} sign(\boldsymbol{x}) \tag{3.6}$$

  where $sign(\cdot)$ is the element-wise sign operator. Assuming 32 bit representation for the entries of $\boldsymbol{x}$, sign quantization reduces the payload size from $32d$ bits to $(32 + d)$ bits, one bit for each vector dimension and 32 bits to transmit the original vector Euclidean norm.

- k **Random Sparsification** [43]: Given a vector $\boldsymbol{x} \in \mathbb{R}^d$ random sparsification returns a vector $\boldsymbol{x}_{rand,k} \in \mathbb{R}^k$, with $k \leq d$, that contains k random components of $\boldsymbol{x}$. Clearly the reduction on bandwith is k/d.

- **Top-k Sparsification [37]**: It sparsifies the input vector $\boldsymbol{x} \in \mathbb{R}^d$ by producing a vector $\boldsymbol{x}_{top,k} \in \mathbb{R}^d$ that contains only the $k \leq d$ largest components of $\boldsymbol{x}$.

Message compression allows to drastically reduce the communication delay during the consensus phase while marginally harming the convergence speed of the optimization procedure both theoretically and empirically. Koloskova et al. [42] showed that employing message compression only affects lower order terms in the convergence guarantees and empirically demonstrated that deep learning models, such as ResNet20, can be effectively trained using message compression [44].
An alternative way to cut down the number of transmitted bits consists in allowing multiple local updates between two subsequent communication steps. While local training can dangerously bias the local models and prevent the convergence of the optimization process [45],

in case of homogeneous datasets allowing multiple gradient steps can speed up the convergence of the algorithms [46, 47]. Combination of local-SGD and message compression have been also considered to further reduce the communication complexity of decentralized optimization schemes [48].

3.2. Distributionally robust optimization

We now shift our point of view on the decentralized training procedure from a purely optimization perspective to a statistical one. We first highlight the fact that in the case of data heterogeneity across participating parties, a model minimizing such definition of risk (3.1) can lead to unsatisfactory and unfair [1] inference capabilities for certain subpopulations. Consider, for example, a consortium of hospitals spread across the world sharing medical data to devise a new drug. Assume that a small fraction of hospitals has medical records influenced by geographical confounders, such as local diet, meteorological conditions, etc., that modify the patient response to the drug. In this setting, a model obtained by myopically minimizing the standard notion of risk defined over the aggregate data can be severely biased towards some populations at the expense of others. This can lead to a potentially dangerous or unfair medical treatment.

To tackle this issue, distributionally robust learning (DRL) aims at maximizing the worst-case performance over a set of distributions $\mathcal{P}$, termed as uncertainty set, which possibly contains the testing distribution of interest. This typically means solving a minimax problem of the type

$$\max_{P\in\mathcal{P}} \min_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathbb{E}_{z\sim P}[f(\boldsymbol{\theta}, z)]. \tag{3.7}$$

Typical choices of the uncertainty sets are balls centered around the training distribution [50] or, whenever the training samples come from a mixture of distributions, the set of potential subpopulations resulting in such mixture [51, 52]. Robust distributed learning with heterogeneous data, in which different distributions exist at the various devices, falls in the latter category, as the natural ambiguity set is the one represented by the convex combination of the local distributions. In that case, minimizing the worst-case risk is equivalent to trying to ensure a minimum level of performance at each participating device.

Specifically for the federated case, Mohri et al. [49] introduced agnostic federated learning (AFL) as a means to ensure fairness and proposed a gradient based algorithm to solve the underlying minimax optimization problem. Specifically, denoting by $\{P_i\}_{i=1}^m$ the data-generating distributions at the network devices, it is known that the federated learning procedure seeks for a model that minimizes the risk w.r.t. the measure $P = \frac{1}{m}\sum_i P_i$; however, this choice for the target distribution is arbitraty and can lead to extremely poor local testing performance at the nodes whose local distribution differs from P. Therefore, in order to guarantee a minimum level of performance across all devices, AFL tries to find a minimizer of the worst case risk w.r.t. the set of distributions $\mathcal{P} := \left\{\sum_{i=1}^m \lambda_i P_i : \boldsymbol{\lambda} \in \Delta^{m-1}\right\}$ where Δ^{m-1} denotes the $m-1$ probability simplex. This is pursued considering the following minimax

[1] In the machine learning community, the notion of fairness has many facets. We use the term "fair" in accordance with the notion of good-intent fairness as introduced in [49].

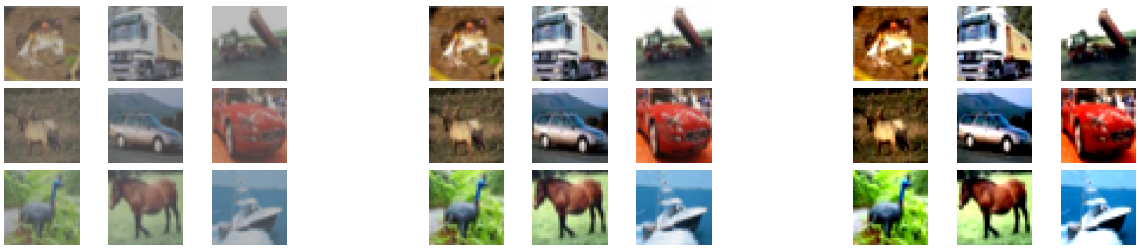
optimization objective function

$$\max_{\boldsymbol{\lambda}\in\Lambda} \min_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{i=1}^{m} \frac{\lambda_i}{|\mathcal{D}_i|} \sum_{z\in\mathcal{D}_i} \ell(\boldsymbol{\theta_i}, z_i) + \alpha r(\boldsymbol{\lambda}) \tag{3.8}$$

which can be interpreted as an adversarial two players game, with one player trying to minimize the risk while the other adversarially picks the testing distribution from all possible mixtures of the local distribution. The term $r(\boldsymbol{\lambda})$ acts as a regularization term which prevents degenerate solutions, for instance the one in which only one local loss term is minimized. Mohri also proposed a gradient ascent-descent strategy to solve the optimization problem (3.8) in the federated learning context. Later in [53], a communication-efficient version of the optimization algorithm, which avoids frequent retransmissions of the dual variables $\boldsymbol{\lambda}$, was proposed.
The distributionally robust learning paradigm can be extended to the fully decentralized set-up. In this context it becomes even a more promiment problem considering the heterogeneous nature of collaborating devices and sparser communication topology. To illustrate the drawback of the standard objective function of decentralized learning procedures and at the same time highlight the positive effect of the distributionally robust procedure, we propose the following experiment. We consider an image classification task using the CIFAR-10 dataset [54], which contains RGB images from 10 different classes. Data heterogeneity is introduced by evenly partitioning the training set across 10 network nodes and changing the contrast of the images stored at the network devices. The pixel value $P \in [0, 255]$ is modified using the following transformation

$$f_c(P) = \mathcal{P}_{[0,255]}\big[128 + c(P - 128)\big] \tag{3.9}$$

where $\mathcal{P}_{[0,255]}$ is the projection from $\mathbb{R}$ to the set of natural number from 0 to 255, extrema included. For $c < 1$ the contrast is reduced while for $c > 1$ it is enhanced. We consider one network node storing images with reduced contrast ($c = 0.5$), one storing images with increased contrast ($c = 1.5$), and the rest of nodes storing images with the original contrast level ($c = 1$).
In Fig. 3.1 we report a batch of images whose contrast is tuned using these three different contrast levels. We arrange the 10 network nodes in a ring topology to collaboratively train a neural network that comprises two convolutional layers with 16 and 32 filters of size 3×3 followed by two fully connected layers with 64 and 32 hidden units. The neural network is trained using the SGD optimizer for $T = 5000$ iterations. We first study how the regularization parameter α in (3.8) affects the robustness of the final solution when using the regularizing function $r(\boldsymbol{\lambda}) := \sum_i \frac{(\lambda_i - n_i/n)^2}{n_i/n}$. Large values of α stabilize the dual variable $\boldsymbol{\lambda}$ to match the relative fraction of data samples at each node, therefore making the distributionally robust objective similar to the standard one. Conversely, when α is small, $\boldsymbol{\lambda}$ can change freely in order to prioritize and weight more the local loss terms associated to nodes with lower accuracy which, in turns, enhances fairness.
We show this in Fig. 3.2 in which we let α take values $\{10, 1, 0.1\}$ and we plot the average accuracy together with the worst-node distribution accuracy. As expected the α parameter allows to interpolate between the standard and distributionally robust approach; in particular for $\alpha = 0.1$ the worst-node accuracy increases by 5% compared to the highly regularized solution ($\alpha = 10$), while the average accuracy only slightly reduces. We also compare the



(a) $c = 0.5$ **(b)** $c = 1$ **(c)** $c = 1.5$

Fig. 3.1: Examples of images with differently tuned contrast.

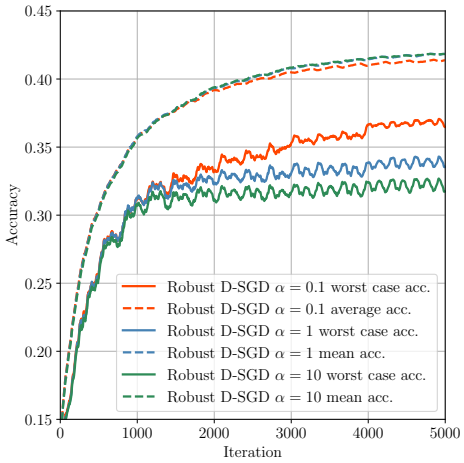
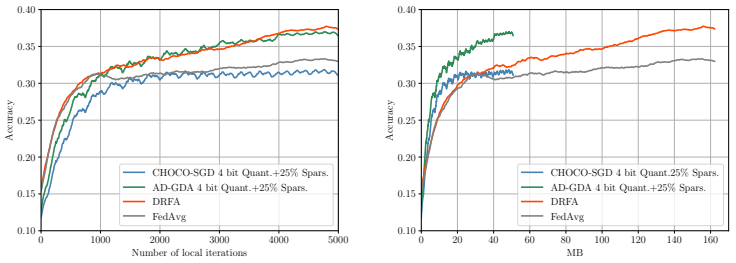


Fig. 3.2: Effect of regularization.



(a) Worst-node accuracy vs. oracle calls **(b)** Worst-node accuracy vs. transmitted bits

Fig. 3.3: Comparison of worst-node accuracy attained by Distributionally Robust Decentralized SGD, Choco-SGD, DRFA, and FedAvg.

proposed distributionally robust approach with CHOCO-SGD [42]. In both cases we employ 4-bit random quantization combined with top-25% sparsification. We also consider the federated scenario with 10 devices orchestrated by a parameter server and with each device storing one of the previously defined data partitions. In this setting we consider the distributionally robust algorithm DRFA [53] with half user participation and the standard federated averaging scheme (FedAvg) [55]. In both cases, aggregation is performed after 10 local iterations. For all schemes, images with reduced contrast represent the most challenging subpopulation. In Fig. 3.3 we report accuracy of the final model when tested on reduced contrast samples. Distributionally robust methods improve by 5% the worst-case distribution accuracy of the final model compared to the standard approach and, despite sparser communication topology, the decentralized model obtains similar worst-case guarantees for the same number of per-node local iterations. Furthermore, thanks to message compression techniques, it is possible to perform decentralized distributionally robust optimization sending only a fraction of bits compared to the federated counterparts.

4. Machine Learning for Wireless Sensing

Wireless sensing is a technology through which information about a remote target is acquired using Radio Frequency (RF) signals (e.g. WiFi). With the advance of wireless sensing technologies, many applications have emerged, ranging from fall-detection [56] and gesture-based human-robot interactions [57] to vision-based wireless communications [58]. Many physical layer properties of a wireless channel can be utilized to realize wireless sensing. Received Signal Strength Indicator (RSSI), Channel State Information (CSI), Doppler shift, and Frequency-Modulated Continuous-Wave (FMCW) techinques are the four commonly used properties to perform wireless sensing.

4.1. Sensing Techniques

4.1.1. RSSI

Most WiFi devices provide the path loss of wireless signals with respect to a certain distance d which can be derived from Long-normal Distance Path Loss (LDPL) model [58]:

$$P(d) = P(d_0) + 10\gamma log\frac{d}{d_0} + X_\sigma \tag{4.1}$$

where $P(d)$, $P(d_0)$, γ, and X_σ are RSSI measurment indicating path loss at distance d (dB), path loss at the reference distance d_0, path loss exponent, and a zero-mean normal noise caused by flat fading respectively.
Since the surrounding environment causes signal attenuation which leads to a variation in RSSI measurements, RSSI-based wireless sensing has been successfully applied to indoor localization [59], crowd density estimation [60], and breathing rate monitoring [61]. However, given the coarse-grained channel state information used in RSSI-based sensing, it is not feasible to utilize this technique in applications requiring fine-grained details like gesture recognition and sign language detection. Moreover, the stability of RSSI-based sensing is not guaranteed even for a static indoor environment [62] which makes it not suitable for many real-life applications.

4.1.2. CSI

For applications that require more details of the target, we should use fine-grained CSI representing the effect of scattering, fading, and power based on the distance of target.

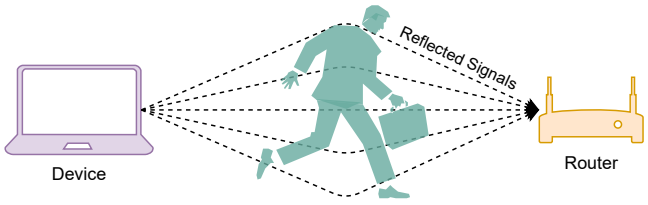


Fig. 4.1: Schematic illustration of the effect of surrounding environment on wireless signals

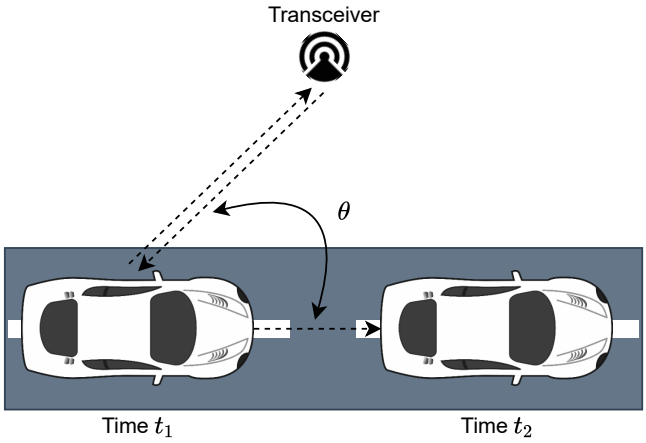


Fig. 4.2: Schematic illustration of the Doppler effect used for speed estimation

As shown in Fig. 4.1, we can use wireless signals reflected by humans to recognize gestures and activities, since multiple reflected rays will be affected by the surrounding environment including the human target [63]. Unlike RSSI, the set of complex values existing in CSI reflects information for both amplitude and phase of Orthogonal Frequency Division Multiplexing (OFDM) sub-carriers. Given the fact that each sub-carrier undergoes a different fading effect, they can contribute to the wireless channel characteristics making it fine-grained enough for detecting even subtle movements. CSI is widely used for human activity recognition [64], intrusion detection, and gait characteristics estimation.

4.1.3. Doppler Shift

The Doppler effect or Doppler shift is the change in frequency of the received signal in relation to an observer who is moving relative to the transmitter. As shown in Fig. 4.2, if we consider the signal reflected from the car as a signal emitted from a transmitter, any movement in the car would result in a Doppler shift. The amount of change in the frequency can be derived as [63]:

$$\Delta f = \frac{2vcos(\theta)}{c} f \tag{4.2}$$

where v, θ, c, and f are relative speed of the target with respect to the receiver, direction of movement, speed of light, and the center frequency of signal respectively.
Human activity [65] and gesture recognition [66] applications are two of the most common areas which have benefited from Doppler effect.

4.1.4. FMCW

As the name implies, FMCW radars transmit a frequency-modulated signal continuously to measure range, angle, and velocity of target with respect to the radar. This differs from traditional pulsed-radar systems, which transmit short pulses periodically. FMCW radars use a specific type of signals called chirps frequency of which increases through time as shown in Fig. 4.3. A chirp is characterized by its start frequency (f_0), bandwidth (B), and duration (T_c). An FMCW radar sense the environment by transmitting a chirp and capturing the reflected signal by objects in its path.

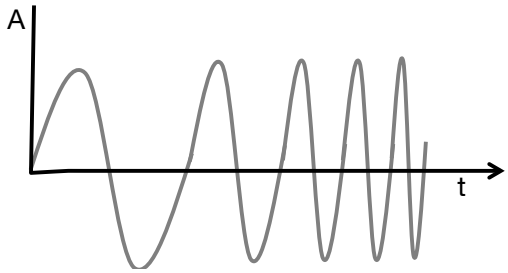


Fig. 4.3: A chirp signal in time domain

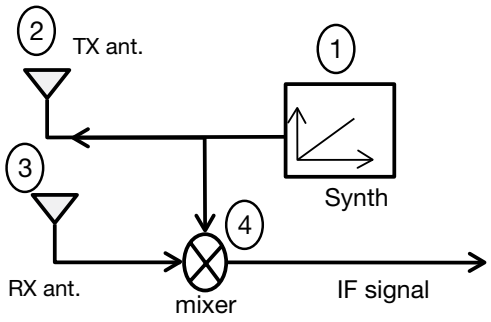


Fig. 4.4: Block diagram of an FMCW radar

As shown in Fig. 4.4, an FMCW radar has the following components:

- A synthesizer (synth) which is responsible for generating chirp signals.
- A transmitter antenna (TX ant.) that transmits the generated chirp.
- A receiver antenna (RX ant.) which receives the reflected chirp signal from the environment.
- A mixer which combines the transmitted and the received signals to generate the Intermediate Frequency (IF) signal.

For two sinusoidal signals x_1 and x_2:

$$x_1 = sin(\omega_1 t + \phi_1) \tag{4.3}$$

$$x_2 = sin(\omega_2 t + \phi_2) \tag{4.4}$$

The IF signal, x_{out}, has an instantaneous frequency equal to the difference between the instantaneous frequency of x_1 and x_2 which is a constant value:

$$x_{out} = sin[(\omega_1 - \omega_2)t + (\phi_1 - \phi_2)] \tag{4.5}$$

In summary, for an object at a distance d from the radar, the IF signal will be a sine wave:

$$Asin(2\pi f_0 t + \phi_0) \tag{4.6}$$

where $f_0 = \frac{s2d}{c}$, $\phi_0 = \frac{4\pi d}{\lambda}$, s is the frequency of IF signal, c is the speed of light, and λ is the wavelength. So far, we discussed the case with a single object in front of the radar. In case of multiple objects, we will have more than one delayed signal resulting in multiple tones in the IF signal. A Fast Fourier Transform (FFT) should be applied to the IF signal with multiple tones to separate the different tones. Fourier transform will result in a frequency spectrum with separate peaks for each tone, implying the presence of an object at a specific distance. As mentioned before, FMCW radars can also measure the angle and velocity of targets [67]. Given the fine-grained information provided by the radars, many applications which require higher resolutions, increasingly use FMCW radars. Gesture recognition using mmWave FMCW radars as a privacy-preserving and non-invasive way of non-verbal communications is one of the examples which will be discussed in more detail in the following section.

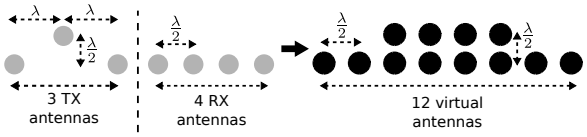


Fig. 4.5: Chirp signal transmission. A block of chirps completes one transmission loop of the three Tx antennas.

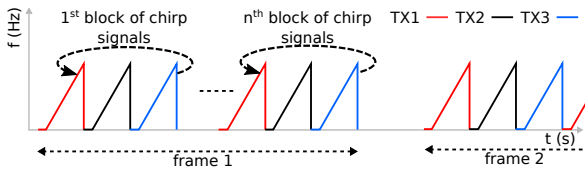


Fig. 4.6: Basic configurations of the radar sensor.

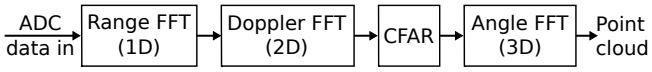


Fig. 4.7: Radar signal processing of the IWR1443 sensor.

4.2. Gesture Recognition Using mmWave Radars

Gesture interaction is an active research topic in Human-Computer Interaction (HCI), with a history dating back to the 1960s and Sutherland's Sketchpad [68] or The Ultimate Display [69] projects. Gestures can be categorized into two broad groups: (1) *mid-air* or *motion* gestures, used e.g. in consumer electronics such as gaming consoles, and (2) *stroke* gestures, used e.g. in touch-capable devices such as smartphones. Motion gestures do not require a handheld or a dedicated input device (e.g. a stylus) and allow for natural interaction with less spatial constraints than stroke gestures. Mid-air interaction exploits whole body movements through gestures and postures to interact with digital content [57].
Pantomime system is designed to recognize sparse gesture motions by means of a hybrid model architecture for optimized spatio-temporal feature extraction. Pantomime works on point clouds generated using an IWR1443 mmWave FMCW radar. A point cloud is a sequence of *frames*, each comprising an unordered set of points in a 3D space.

4.2.1. Signal Processing

As shown in Fig. 4.5, the IWR1443 sensor has three Tx antennas, separated by one wavelength (λ), and four Rx antennas, each $\lambda/2$ apart. This configuration yields 12 virtual antennas. During each time slot, the corresponding antenna sends a chirp signal, which is a frequency ramp for the FMCW operation; cf. Fig. 4.6.
Fig. 4.7 illustrates the signal processing pipeline of the radar sensor, starting with Analog to Digital Converter (ADC) and ending with the computed point cloud:

1. *Range-FFT*: The radar sends a chirp signal and the mixing operation between the transmitted and received chirps produces an intermediate frequency signal. The range of the detected object is linearly proportional to the frequency of such signal, which is computed using the FFT operation.

2. *Doppler-FFT*: At least two time-separated chirps are used to estimate the radial velocity of an object. The phase difference of two chirps at the range-FFT peak is proportional to the radial velocity of the detected object. An FFT operation on the signal range (i.e., a 2D-FFT or a Doppler-FFT) produces a peak at the velocity of the object.

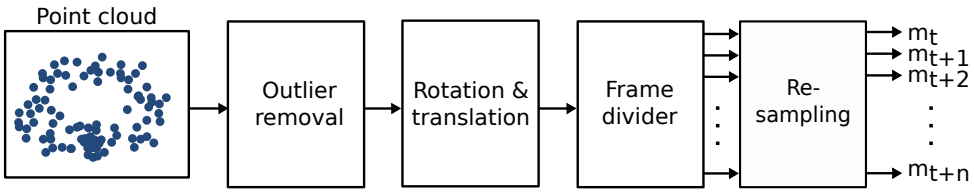


Fig. 4.8: Point cloud preprocessing block

3. *False Alarm Rate Detection (CFAR)*: The summation of the Doppler-FFT matrices creates a pre-detection matrix for each virtual antenna. The constant CFAR algorithm identifies peaks in the pre-detection matrix that correspond to the detected objects.

4. *Angle-FFT*: For each object, an FFT of the angle is performed on the corresponding CFAR peaks across multiple Doppler-FFTs. Velocity-induced phase changes are Doppler-corrected before computing the angle-FFT.

4.2.2. Point Cloud Processing

As shown in Figure 4.8, the preprocessing block consists of four stages: outlier removal, rotation and translation, frame divider, and re-sampling.

4.2.2.1. Outlier Removal

As hinted previously, the point cloud contains noise because of scatterings and reflections from the environment. This happens often in cluttered environments. Fortunately, this noise is sparse and can be removed using the following procedure.
The point cloud is received at a fixed frame rate but the number of points in each frame can vary due to the CFAR operation, see Section 4.2.1 for more details. We first wait until all frames corresponding to a single gesture are collected over time, and then aggregate all the frames to construct the point cloud of the gesture. When the whole point cloud is available, it is easy to separate the outliers based on point density. For this, we apply the DBSCAN algorithm with $\epsilon = 1$, which is the maximum distance between two points that can be considered as neighbors, and set the minimum number of points in a cluster to 3. We fine-tuned these parameters using grid search. The output of the DBSCAN algorithm produces a major cluster around 0.5 m diameter, representing the points reflected from the user's body and few small clusters representing the environmental noise. We keep the points in the major cluster and consider all other clusters as outliers.

4.2.2.2. Rotation and Translation

Once the major cluster is identified, we measure the cluster centroid's angle with respect to the boresight angle (i.e., in relation to the centre-line of the antenna) and estimate its distance with respect to a reference distance. In our experiments the reference distance is set to 1.5 m. If the centroid deviates from the reference angle and reference distance, the detected object is rotated and translated by the deviated amount. This is done to normalize the position and to reduce the effect of the position on the recognition accuracy.

4.2.2.3. Frame Divider

The aggregated point cloud is rearranged into ordered frames, based on the time at which each point was received. To reduce the time complexity of our model and to study the effect of the number of frames on the recognition accuracy, we decrease the original number of frames received from the radar device; e.g. for a 2 s gesture we reduce the number of frames from 100 to 2, 4, or 8 but retain all the points in those frames. To do so, we apply *time decay* on the points such that we rearrange the first set of k/f points in the first frame, where k is the total number of points in a gesture and f is the desired number of frames; second set of k/f points as the second frame, and so on. Although this procedure fixes the number of frames, we still have gestures with different number of points per frame. However, the PointNet++ architecture dictates a fixed number of points in each frame for all the gestures, so we apply a re-sampling algorithm, as explained next.

4.2.2.4. Resampling

We resample the number of points in a frame to achieve a fixed number of points in each frame while preserving the shape and the density of the point cloud. Following Cohen et al. [70], we use Agglomerative Hierarchical Clustering (AHC) for upsampling and the K-means algorithm for downsampling. In each step of the AHC algorithm, the centroid of each cluster is added to the point cloud as a new point, which prevents producing singleton clusters. We run AHC iteratively until reaching our predefined fixed number of points in a frame. For downsampling, we use K-means with K equal to the fixed number of points per frame and select the centroids of the clusters as the points in the point cloud.

4.2.3. Point Cloud Classification

In this section, we introduce Pantomime model which aims to capture spatial features of point clouds through a hierarchical feature learning scheme. To learn the temporal evolution of the gestures, Pantomime fuses spatial features using a two layer Long Short-Term Memory (LSTM) model.

4.2.3.1. Hierarchical Feature Learning on Static Point Clouds

Direct processing of point clouds instead of voxelizing or projecting them, was introduced by PointNet [71]. Given an unordered set of points $\{x_1, x_2, ..., x_n\}$ where $x_i \in \mathbb{R}^d$, we can use a set function $f : \chi \rightarrow \mathbb{R}$ to transform the set of points to a vector:

$$f(x_1, x_2, ..., x_n) = \gamma \left(\max_{i=1,...,n} \{h(x_i)\} \right) \tag{4.7}$$

where γ and h are usually Multi-Layer Perceptron (MLP) networks. It has been shown in [71] that the set function in (4.7) is invariant to input point permutation which is highly important for point cloud processing. The response of h can then be considered as the spatial encoding of a point. However, by design, PointNet does not capture local structures as well as features from different scales. While PointNet uses a single max pooling layer to extract the representation for the whole point cloud, PointNet++ [72] applies PointNet recursively on a nested partitioning of the input point cloud to learn features from different scales and capture local structures through *set abstraction* layers. At each *set abstraction* layer, which

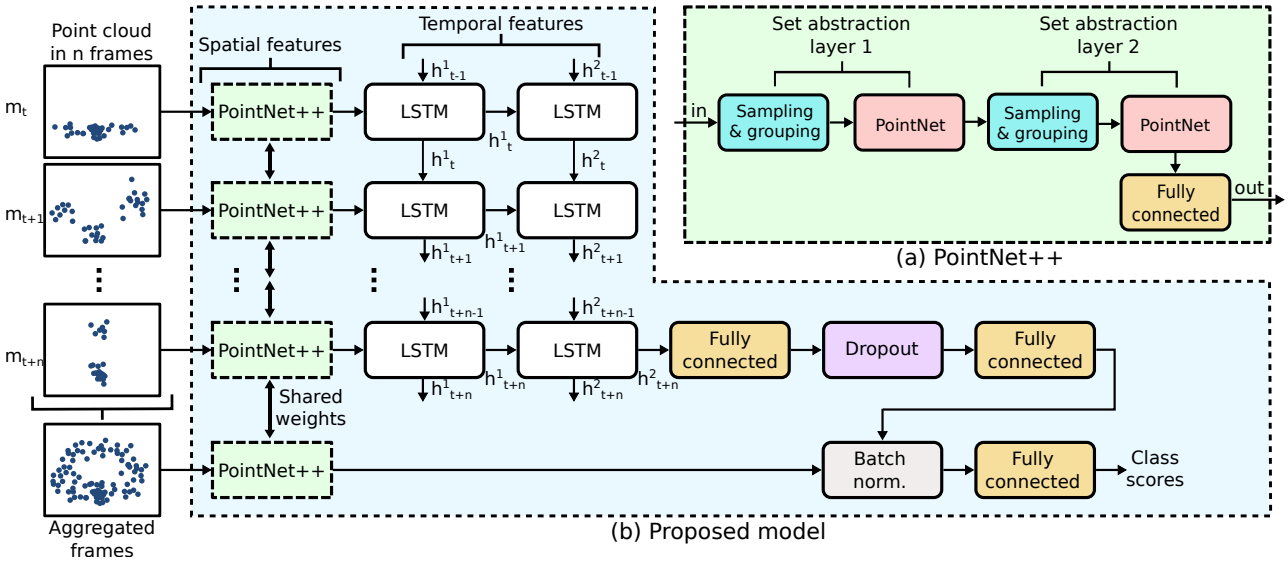


Fig. 4.9: Sparse point cloud classification architecture.

consists of sampling, grouping, and PointNet layers, the goal is to produce a new set with fewer points. The sampling layer selects a subset of points through Farthest-Point Sampling (FPS) operator, the grouping layer constructs a local region by finding neighboring points using ball query algorithm, and PointNet layer encodes local region patterns into feature vectors.

4.2.3.2. Learning Spatio-Temporal Features of Sparse Point Clouds

We propose a hybrid architecture combining the PointNet++ architecture followed by LSTM modules for frame-wise spatio-temporal feature extraction, as shown in Figure 4.9. We follow this approach because, on the one hand, in highly sparse point clouds, applying PointNet++ on each frame may not capture the spatial features efficiently because fine-grained local features (e.g. skeletal structure of the hand) are not well preserved between consecutive frames. On the other hand, applying PointNet++ on the aggregated point cloud will preserve spatial features but the directionality of certain gestures (e.g. clockwise circling and anti-clockwise circling) would be lost because temporal features are not captured. As a result, we use PointNet++ to hierarchically extract spatial features from each frame and the aggregate frames, then we use LSTM layers to fuse the spatial features and capture the temporal evolution of the gestures.

4.2.4. Dataset

To evaluate the model, we collected a dataset from more than 41 participants in 5 environments, with 7 angles, 3 speeds, and 5 distances. In total we acquired more than 10,000 samples which is publicly [1] available.

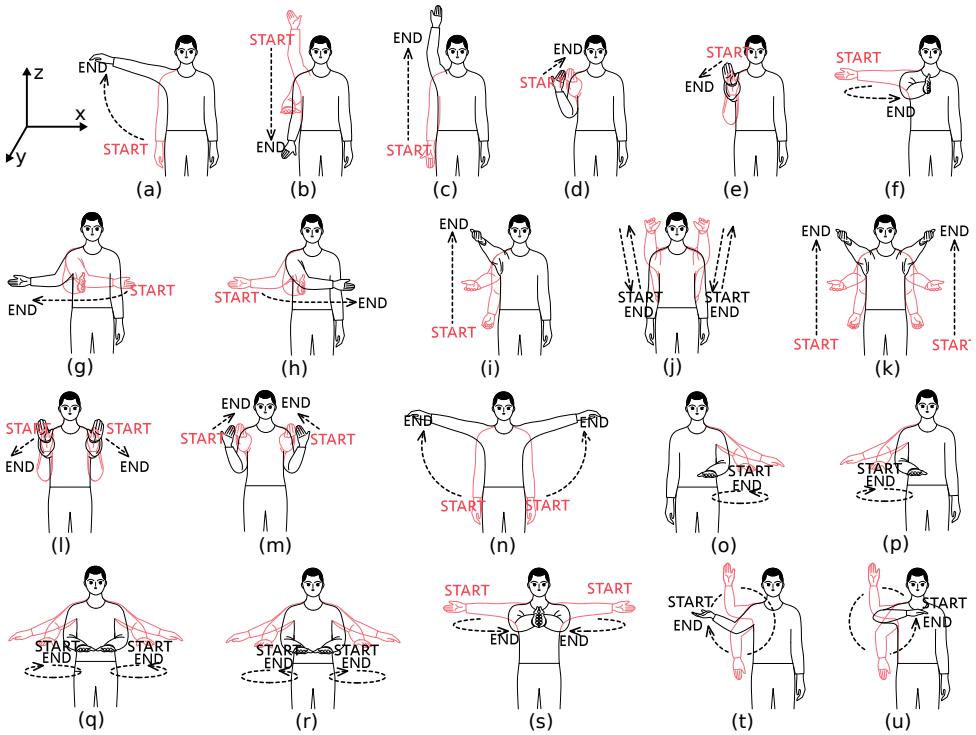


Fig. 4.10: Gesture set used in our experiments. EASY set: (a) 'lateral raise', (b) 'push-down', (c) 'lift', (d) 'pull', (e) 'push', (f) 'lateral-to-front', (g) 'swipe right', (h) 'swipe left', (i) 'throw'. COMPLEX set: (j) 'arms swing', (k) 'two-hand throw', (l) 'two-hand push', (m) 'two-hand pull', (n) 'two-hand lateral-raise', (o) 'left-arm circle', (p) 'right-arm circle', (q) 'two-hand outward circles', (r) 'two-hand inward circles' (s) 'two-hand lateral-to-front', (t) 'circle clockwise', (u) 'circle counter-clockwise'.

4.2.5. Results

Table 4.1 shows the accuracy and AUC values for the three datasets for the 6 models. Importantly, Pantomime achieves the best performance, exceeding the accuracy of the next best classifier by 5.6%, 0.85%, and 5.7% for EASY, COMPLEX and ALL gestures, respectively. It is interesting to note that the results of Pantomime are consistent throughout the three gesture sets with just 1.5% difference between the best and the worst results, whereas the accuracy differences of other classifiers fluctuate between 3.5% and 7% (worst and best results, respectively).

We consider angles between -45° to $+45^\circ$ in increments of 15° with regard to the boresight angle. We selected this set of angles because the field of view of the radar ranges between -55° and $+55^\circ$ azimuth angles. Therefore, within $\pm 45^\circ$ angles it is possible to capture the whole body of the participants within the field of view of the radar. Under this scenario, we collected 2940 gestures (420 samples for each angle) from 8 participants that were used as test data. The results are shown in Fig. 4.11.

As can be observed, the results indicate excellent recognition performance ($> 89\%$) when the user is located within the range of $\pm 15^\circ$. Then, recognition accuracy drops to 84% for angles comprised between -30° to $+30^\circ$ and there is a decrease of about 20 % at $\pm 45^\circ$. This degradation is mostly caused by self-shadowing of one of the participant's arm, due to

[1] https://zenodo.org/record/4459969#.YNxgjnUzaV4

Table 4.1: Comparison with the state of the art. Both Accuracy and AUC are reported in percentages. The best results are denoted in bold typeface.

	EASY		COMPLEX		ALL	
Model	**Acc.**	**AUC**	**Acc.**	**AUC**	**Acc.**	**AUC**
PointNet	79.7	98.4	82.5	98.7	81.6	99.4
PointNet++	79.7	98.1	84.9	99.0	83.6	99.4
O&H	77.7	96.0	83.2	98.1	79.1	97.9
Std. Arch.	83.3	98.5	88.4	99.5	86.3	99.4
RadHAR	91.6	98.9	94.3	99.6	89.9	99.5
Pantomime	**96.6**	**99.8**	**95.1**	**99.8**	**95.0**	**99.9**

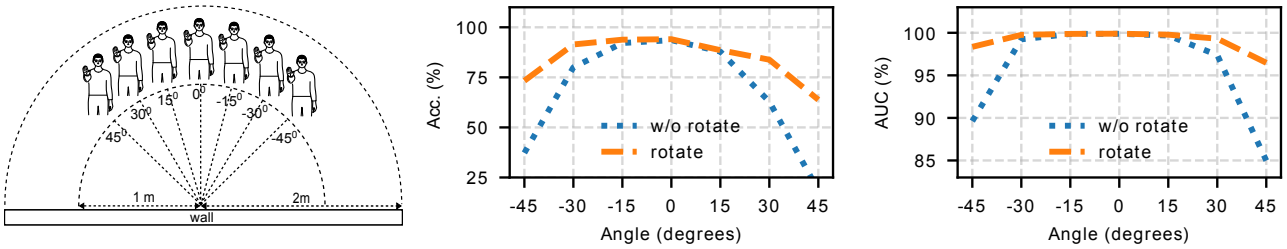


Fig. 4.11: Performance analysis as a function of the angle between the person and the radar. (a) Participant's position (1–2 m) and orientation (parallel to the wall)., (b) Accuracy vs angle. (c) AUC vs angle.. Note: 'rotate' denotes the participant's point cloud is rotated to 0° angle and 'w/o rotate' denotes no rotation of the point cloud.

the orientation of the body, as it was parallel to the radar at all times. So, for extreme angular values the radar simply cannot fully sense the participant's moving arms.
Another important observation is that accuracy in positive angles is slightly lower than the accuracy recorded in negative angles. Since our single-handed gestures are predominantly right-handed, we conjecture that this differences may be a combination of a calibration error of the radar and the right hand of the participants reaching beyond the +55° field of view while performing gestures.

4.2.6. Conclusion

In this section, we briefly discussed the most common techniques in wireless sensing including CSI, RSSI, Doppler shift, and FMCW. Then, we discussed one of our recently published systems called Pantomime which is designed to recognize human gestures using mmWave FMCW radar generated point clouds. We showed that using the proposed model is able to recognize gestures with more than 95% accuracy outperforming state-of-the-art models on a dataset collected from more than 40 participants with 10,000 samples.

5. Sustainable Transmission Design by Joint UAV Trajectory and RIS Phase Optimization

5.1. Introduction

Unmanned Aerial Vehicles (UAVs) and Reconfigurable Intelligent Surface (RIS) are being explored as potential new technologies to enhance the network coverage, and thereby, the service availability of cellular networks. The conceptual design of RIS consists of several reflective elements which can be configured so as to reflect and, in particular, beamform a signal towards a certain direction. The advantage of using UAVs and RIS in conjunction is the dynamic network configurations and flexibility that can be exploited to adapt the system based on the network load and service requirements. Indeed, UAVs and RIS can be used to create mobile micro cells to serve temporary hotspots, i.e., areas with very high service requirements at a certain time as well as providing necessary service coverage in semi-urban and rural environment with minimal infrastructure. So, the joint usage of UAVs and RIS can be utilized to learn and adapt the network based on the information such as user mobility and density to satisfy the user service requirements [73–75]. Additionally, the rapid deployment of high frequency communication technologies, such as mmWave, to satisfy the higher bandwidth requirements in beyond 5G cellular networks, has contributed to increase the importance of exploiting the unconstrained mobility of UAVs and adaptable Electromagnectic (EM) signal processing capabilities of RIS. These new technologies have been analytically shown to be able to provide significant improvements in terms of service availability for dense urban networks. But the usage of high frequency technologies in conjunction with both UAV and RIS raises new challenges in terms of network optimization. In particular, a significant issue is created with regards to the communication range and coverage quality in a dense urban scenario.

One of the major hurdles while using both these technologies is the energy consumption of the system. UAVs, especially quadcopters, generally run on small batteries and the energy consumption is very high when the UAV is in flight. Therefore, to provide sustained coverage to the User Equipments (UEs) with high Quality of Service (QoS) requirements, the trajectory of the UAV has to be optimized. The use of RIS can help satisfy the required QoS potentially reducing the need for UAVs to travel further, with a small trade off on the energy consumed for operating by the RIS [76].

In this chapter, we devised a multi-objective joint optimization problem to determine UAVs trajectory, UAV transmission power and RIS phase with an aim to reduce the transmission energy consumption of the entire system while providing a certain level of QoS to the UEs in the area. We incorporate the Successive Convex Approximation (SCA) technique to determine a nearly optimal solution. In short, the devised optimization strategy using SCA is able to optimize different objectives of UAV trajectory, UAV transmission power and RIS phase jointly while satisfying the service requirements of the UEs.

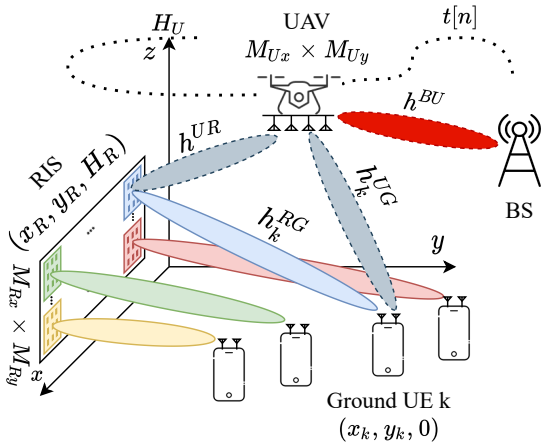


Fig. 5.1: The Scenario Environment with UAV, RIS, BS and UEs

5.2. Scenario Definition

Consider a network environment shown in Fig. 5.1 with K UEs randomly spread in the area. There are L UAVs available at the Base Station (BS) to provide a wide spread connectivity to the UEs. In the environment, there are R RISs in fixed and known positions. Hence, the assumptions made for the scenario are as follows:

- User association and additional control information needed for data transfer are exchanged between BS and UEs by means of a dedicated long range control channel.
- UEs are aware of their own position (e.g., calculated through triangulation with respect to the BSs in the area).
- The UEs periodically communicate their position information to the BS that congeals this information to devise mobility patterns and traffic requirements in the environment.
- BSs, UAVs and UEs are equipped with Uniform Planar Array (UPA) antennas so as to perform concurrent beamforming in different directions.
- An extra large scale massive MIMO (XL-MIMO) RIS deployment is considered in which every UE is served by a specific region of the surface. This holds when the RIS dimensions is large and the UEs are sufficiently spaced apart to have partial observability of the surface [77].

5.2.1. UAV-BS Link

BS and UAVs are supposed to always maintain Line-of-Sight (LoS). So, the Signal-to-Interference-plus-Noise Ratio (SINR) received at the j^{th} UAV with respect to the associated BS is given by,

$$\gamma_j[n] = \frac{P_{BS}^{T}[n] \, |\mathrm{H}_j^{\mathrm{BU}}[n]|^2}{\mathrm{I} + \sigma_n^2}, \tag{4.1}$$

where $\mathrm{P}_{BS}^{T}[n]$ is the transmit power of the BS, $|\mathrm{H}_j^{\mathrm{BU}}[n]|$ denotes the channel gain at timestep n and the term I refers to the interference caused by signals from different BSs and signals

from the associated BS to other UAVs in the area. Finally, σ_m^2 is the white noise power. Using (4.1) and Shannon-Hartley's Theorem, the rate received in $bit/s/Hz$ at the j^{th} UAV is assumed to be equal to

$$C_j[n] = \log_2(1 + \gamma_j[n]), \tag{4.2}$$

5.2.2. UAV-UE and UAV-RIS-UE Links

For the link between UAV and UE, which is LoS, and between UAV and UE through RIS, we adapt the channel model from [76]. Hence, the SINR of the LoS link from the j^{th} UAV to the k^{th} ground UE (UG) and that through the RIS (URG) are given by

$$\gamma_{j,k,1} = \frac{P_{j,k}^T |\mathrm{H}_k^{\mathrm{UG}}[n]|^2}{\mathrm{I} + \sigma_n^2}, \tag{4.3}$$

$$\gamma_{j,k,2} = \frac{P_{j,k}^T |\mathrm{H}_k^{\mathrm{URG}}[n]|^2}{\mathrm{I} + \sigma_n^2}. \tag{4.4}$$

where $P_{j,k}^T$ is the transmit power of the UAV and $|\mathrm{H}_k^{\mathrm{UG}}[n]|$ and $|\mathrm{H}_k^{\mathrm{URG}}[n]|$ denote the channel gains at timestep n for LoS and RIS link, respectively. Using the above equations, the total rate received in $bits/s/Hz$ by the k^{th} UE can be computed as,

$$R_{j,k,1} = \log_2(1 + \gamma_{j,k,1}), \tag{4.5}$$

$$R_{j,k,2} = \log_2(1 + \gamma_{j,k,2}), \tag{4.6}$$

$$R_{j,k} = \sum_{i=1}^{2} R_{j,k,i} = \log_2((1 + \gamma_{l,k,1})(1 + \gamma_{l,k,2})). \tag{4.7}$$

5.3. Optimization Problem Definition

Considering the assumptions, the objective is to find a sustainable UAV path in order to minimize the overall transmission power consumption under minimum QoS and maximum UAV energy budget constraints over a certain time horizon of N timesteps. Hence, the optimization variables defined for the problem are the following:

- P denotes the UAV and BS transmission power.
- Z denotes the UAV trajectory
- V denotes the UAV velocities over the trajectory
- Φ denotes the RIS phase configurations.

The cost function in the minimization problem in (4.8) is the overall power consumption for transmission. Even though the power consumption of RISs is small compared to that of UAVs, it is necessary to be considered with respect to the overall power consumption of the system as the power consumed by the RIS depends on the frequency of reconfiguration of the surface, which is very high when the RIS is deployed in UAV cellular networks.

Additionally, the constraints for the optimization can be defined as follows,

- **C1–Guaranteed Rate Constraint**: C1 is devised to provide a guaranteed service rate to each one of the K UEs We recall that $R_{j,k}$ is the sum rate achieved over the LoS and RIS links, which has to stay above the guaranteed rate R_{min}.

- **C2—Backhaul Capacity Constraint**: C2 ensures that the backhaul link capacity is greater than or equal to the minimum guaranteed rate of the UEs so that the UAVs have enough bandwidth capacity towards the BS to provide at least the minimum guaranteed rate to all the UEs.

- **C3—RIS Energy Budget**: C3 is devised to define an energy budget E_{max}^{RIS} to limit the usage of the RIS over a period of time, thereby limiting the power consumption P^{RIS} of the RIS.

- **C4—Phase Shit Constraint**: C4 limits the phase shift with respect to the incident signal from 0 to 2π. With the assumption of XL-MIMO surface for RIS, the phase shift can be considered almost continuous from 0 to 2π.

- **C5—UAV Energy Budget**: C5 limits the total sum power consumption P_{j}^{UAV} of the UAV over the N timesteps to threshold E_{max}^{UAV}. This threshold defines the maximum energy the UAV can consume before recharging and, implicitly, the maximization length of the UAV path.

- **C6—Timestep Position Constraint**: C6 constraints the position Z[n] of the UAVs in successive timesteps thereby limiting the movement of UAVs in one timestep.

- **C7—Maximum Velocity Constraint**: C7 is devised to constraint the velocity $v[n]$ of the UAVs in one timestep to the maximum velocity V_{max} thereby limiting the maximum distance the UAVs can travel in one timestep.

- **C8—Timestep Velocity Constraint**: C8 is devised to determine the velocity $v[n]$ of the UAVs in successive timesteps based on the acceleration ΔV_{max} of UAVs in one timestep and thereby devising the overall velocity over the total period.

- **C9—Minimum Velocity Constraint**: C9 constraints the minimum velocity of the UAVs, thereby limiting the minimum distance the UAVs can travel over one timestep. Note that if the UAV it can hover at one place in one timestep, then the minimum velocity is zero.

- **C10/C11—Initial/Final Position Constraint**: C10 and C11 set the starting and ending point of the trajectory determined by the optimization problem.

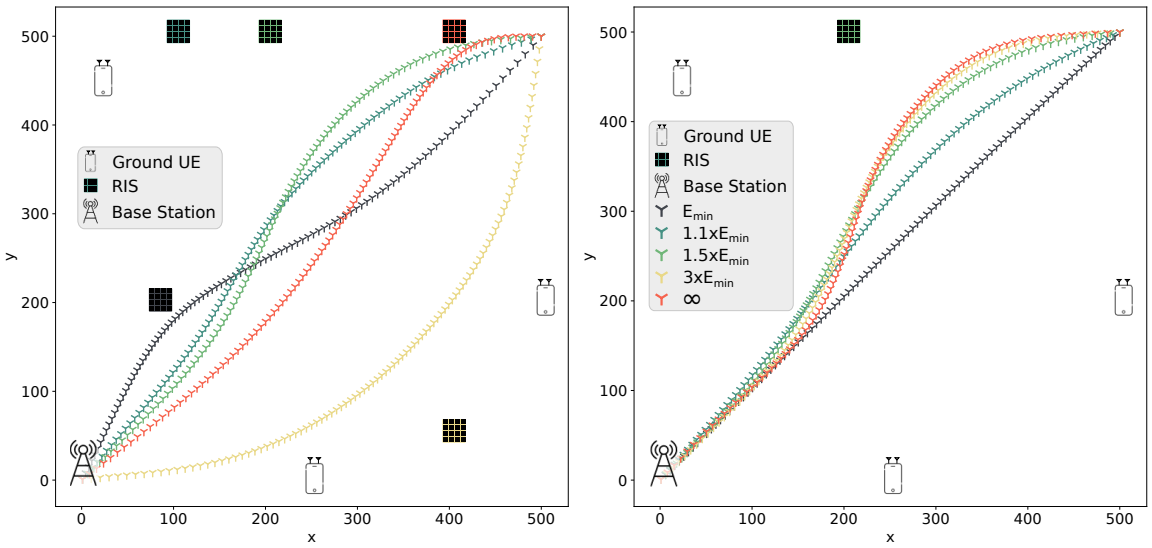
The minimization problem is hence given by

$$
\begin{aligned}
&\min_{\mathrm{P,Z,V},\Phi} \sum_{j=1}^{L}\sum_{k=1}^{K}\sum_{n=1}^{N} \mathrm{P}_{j,k}^{T}[n] + \sum_{r=1}^{R}\sum_{n=1}^{N} \mathrm{P}_{r}^{RIS}[n] + \sum_{n=1}^{N} \mathrm{P}_{BS}^{T}[n] \\
&\quad s.t. \\
&C1: \; R_{j,k}[n] \geq R_{min}, \; k \in 1,\dots,K, \; n \in 1,\dots,N, \; j \in 1,\dots,L; \\
&C2: \; C_{j}[n] \geq KR_{min}, \; \forall \, j,n; \\
&C3: \; \sum_{r=1}^{R}\sum_{n=1}^{N} \mathrm{P}_{r}^{RIS}[n] \leq \mathrm{E}_{max}^{RIS}; \\
&C4: \; 0 \leq \phi\,[n] \leq 2\pi; \\
&C5: \; \sum_{l=1}^{L}\sum_{n=1}^{N} \mathrm{P}_{l}^{UAV}[n] \leq \mathrm{E}_{max}^{UAV}; \\
&C6: \; \mathrm{Z}[n+1] = \mathrm{Z}[n] + \mathrm{v}[n]\tau, \; n = 1,\dots,N-1; \\
&C7: \; \|\mathrm{v}[n]\| \leq V_{max} \; \forall n; \\
&C8: \; \|\mathrm{v}[n+1] - \mathrm{v}[n]\| \leq \Delta V_{max}\tau, \; n = 1,\dots,N-1; \\
&C9: \; \|\mathrm{v}[n]\| \geq 0; \\
&C10: \; \mathrm{Z}[1] = \mathrm{Z}_{0}; \\
&C11: \; \mathrm{Z}[N] = \mathrm{Z}_{F};
\end{aligned}
\tag{4.8}
$$

5.4. Analytical Solution

The optimization problem discussed in the previous section is clearly non-convex, and hence, quite difficult to solve in itself. But we can determine a suboptimal solution by considering an initial transmission power for the UAVs and BS and jointly optimize the UAV trajectory and RIS phase. Once, UAV trajectory and RIS phase are determined, the transmission power is minimized for the given trajectory and phase configuration. This process is iteratively repeated until it finally converges, i.e., the trajectory, phase and power values are no longer changing. This procedure, Successive Convex Approximation (SCA) is used to solve the optimization problem in (4.8).

Under SCA, the UAV trajectory and RIS phase are determined by fixing the transmission power. To determine the UAV trajectory, and accordingly the RIS phase, with fixed transmission power, additional slack variables Λ and M for distance between UAV and UEs/RIS and BS respectively. Also, another slack variable Π is defined to alter the in-flight velocity of the UAV. The three slack variables Λ, M and Π are used by the SCA to determine the UAV trajectory and the corresponding RIS phase. Once the UAV trajectory and RIS phase are determined, the transmission power for UAV and BS can be minimized for the pre-defined trajectory. This approach is iterated for J_{max} number of iterations or till the difference between the total transmission power consumed in consecutive iterations is smaller than ϵ. Algorithm



(a) RIS position **(b)** UAV Energy Budget

Fig. 5.2: Impact of different RIS positions and UAV energy budget on UAV optimal trajectory.

2 shows the approach to obtain a potential solution for the optimization problem defined in (4.8).

Algorithm 2: Joint Trajectory, RIS Phase Configuration and Transmission Power Control algorithm

Result: UAV Trajectory Z, UAV Velocity V, Total Transmission Power P^{Total}

Initialize trajectory Z, Initial velocity V, Maximum number of iteration J_{max}, Initial iteration index $j = 0$, UAV and BS transmission power P and Convergence tolerance ϵ;

while $j \leq J_{max}$ *or* $\frac{P^{j}_{Total} - P^{j-1}_{Total}}{P^{j}_{Total}} \leq \epsilon$ **do**

  Set $j = j + 1$ and $\{P^j, V^j, \Lambda^j, M^j, \Pi^j\} = \{P, V, \Lambda, M, \Pi\}$;

  Solve optimization problem (4.8) to obtain Z, V, Λ, M and Π for a Particular P;

  Solve optimization problem (4.8) to obtain P and P_{Total} for a Particular Z, V, Λ, M, Π

  Update $P^{j}_{Total} = P_{Total}$;

end

5.5. Results

The algorithmic solution explained in the previous section is implemented in a MATLAB environment. The simulation environment consists of one UAV, one RIS, one BS and three UEs. The positions of the UEs, BS and RIS are fixed. The starting and ending point of the UAV is also fixed. We simulated different configurations to show the effect of a certain design parameter on the overall UAV trajectory.

We analyze the impact of different RIS positions on the optimal UAV trajectories for static UEs and BS positions. Fig. 5.2a shows the impact of the different positions of RIS on the UAV trajectory. The first discernible observation is that the UAV attempts to go as close as possible to the RIS. This is due to the fact that transmission power necessary to satisfy the rate requirement for the users is considerably lower when using the RIS compared to directly transmitting to the users. This, however, is compensated for by the higher path loss that is encountered by the signal when the total distance the signal has to cover using RIS is higher than the direct distance between UAV and UE. Due to this fact, the UAV cannot use the RIS to serve all the users at all time steps. Hence some users have to be directly served by the UAV. This creates a push-pull effect on the UAV that cannot venture very close to one user without making other users receive lower rate than required. Hence, determining an optimal position for RIS is important while designing the network.

To study the impact of UAV energy budget on the UAV trajectory, we determine the UAV trajectories for different energy budget values. The energy consumed by the UAV over straight line path (i.e., shortest path), is the minimum in-flight energy consumption necessary for the UAV to reach its final destination Z_F and is hence set as reference, E_{min}. Hence, the energy budget for the UAV is defined as a multiple of E_{min}. Fig. 5.2b denotes the impact of energy budget on the trajectory optimization. As visible from the figure, with an increasing budget, the UAV is able to deviate further away from the shortest path trajectory. But eventually, it cannot go much further as it would risk not serving the users on the opposite side (as discussed previously). This sets an upper bound to the advantage obtained by increasing the energy budget. Indeed, continuously increasing the energy budget starts to provide diminishing returns for values over 300% or E_{min}. This is important when considering the limited battery capacities of the UAVs.

5.6. Conclusion

To conclude, in this work, we devised a multiple objective joint optimization problem to determine an optimal UAV trajectory, RIS phase and UAV transmission power to provide a certain guaranteed service rate to each of the UEs on the ground. To this end, the usage of convex approximation techniques can provide a close to optimal solution.

Moving forward, the usage of reinforcement learning seems very attractive especially due to the sensitive nature of convex approximation schemes to different network configurations.

6. Joint Control and Network Optimization in Wireless Networked Control System

Wireless Networked Control System (WNCS) are essentially part of many industrial domains, such as factory automation, logistics, or transportation. They enable mobile control applications where high flexibility is required. However, due to the nature of the wireless medium, reliability of WNCS remains an open challenge, in particular for low-latency applications.
In a conventional approach, one would separately derive worst case requirements from the control system and impose them on the communication system. Communication latency metric is then often used as a benchmark metric to design and evaluate the communication system. However, such decoupling in system design for low-latency and high reliability leads to over-provisioning communication network resources.
For systems with a sense-compute-actuate cycle, where the receiver is interested in fresh knowledge of the remotely controlled system, rather than the individual packet delay, the notion of Age of Information (AoI) [78] has been proposed as more representative than communication latency. The AoI defines the time that has elapsed since the newest update available at the destination was generated at the source, and it captures not only the communication delays but also the impact of the packet generation at the controlled process.
In recent work on WNCS, authors have been increasingly exploring the inter-relation between the control and communication systems with help of AoI. In [79] and [80], for example, the authors demonstrate how the latency and reliability trade-off directly impacts the system level stability, proposing a co-design of both control and communication entities.
Despite its benefits, the drawback of AoI is that it has been used so far to separately optimize transmissions in uplink (UL) and downlink (DL). However, WNCS applications are closed-loop applications, where the UL communication can affect the DL and vice-versa, resulting in system performance changes or in the use of network resources. In this context, we propose to explicitly address the two-way nature of the control-communications interplay by proposing a new metric, named Age of Loop (AoL), that extends the current AoI definition to take into consideration both UL and DL of the control loop in WNCS, and thus providing a more precise system state estimation. This is essentially a **multi objective** optimization where we coordinate control and communication trade-off by using a learning approach.

6.1. Control system and communication modeling

In this section, we will introduce the system and WNCS model, respectively.

6.1.1. System Model

We consider the classical inverted pendulum system model, a widely used benchmark problem in both control and RL domain. As illustrated in Figure 6.1, a pole is attached by a joint to a cart, which can be moving along a frictionless track. The pendulum starts upright at a random initial angle $\theta_0 \in (\theta_{0,min}, \theta_{0,max})$, and the goal is to prevent it from falling over by applying a force to the cart. While conceptually simple, the system dynamics are highly unstable and continuously requires fast control cycles to keep stability. When, in turn, being

controlled over a wireless channel, the problem becomes an illustrative model of strict timing requirement.

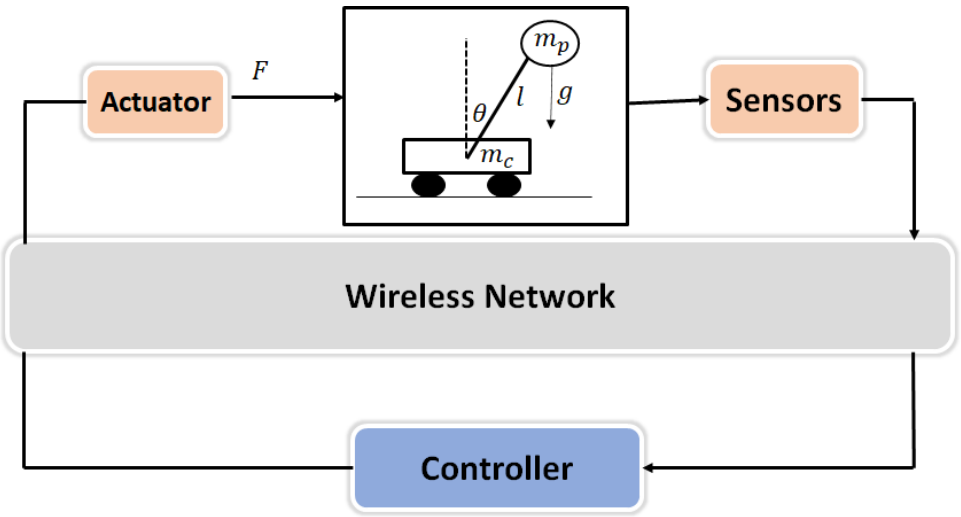


Fig. 6.1: Inverted pendulum system model.

6.1.2. Control System Model

The system dynamics can be represented by the differential equations [81]:

$$\ddot{\theta} = \frac{g \cdot \sin(\theta) + \cos(\theta)\left(\frac{-F - m_p l \dot{\theta}^2 \sin(\theta)}{m_c + m_p}\right)}{l\left(\frac{4}{3} - \frac{m_p \cos^2(\theta)}{m_c + m_p}\right)}, \qquad \ddot{x} = \frac{F + m_p l(\dot{\theta}^2 \sin(\theta) - \ddot{\theta}\cos(\theta))}{m_c + m_p}, \tag{6.1}$$

where x and θ are, respectively, the cart position coordinates and the pole angle from vertical reference. The mass of the cart is m_c, and the mass of the pendulum is m_p, while l is the length of the pendulum, and F is the force applied to the cart under gravity g. We use the Newton's notation ($\dot{\square}$, $\ddot{\square}$) to represent derivatives w.r.t time.
By defining a state space vector $X = [x, \dot{x}, \theta, \dot{\theta}]$, we can design a standard optimal controller in two steps. First, computing the Jacobian of (6.1) around the operating point $X = [0, 0, 0, 0]$ to linearize the plant, so that the system dynamic takes the linear time-invariant form:

$$\begin{cases} \dot{X} = AX + Bu + w, \\ u = -KX, \end{cases} \tag{6.2}$$

where u is the linear state feedback control policy of gain K, w is a process disturbance modeled as a zero-mean and one-standard deviation Gaussian white noise, composed by the system transition matrix A and vector B, respectively calculated as [82]:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-12 m_p g}{13 m_c + m_p} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{12(m_p g + m_c g)}{l(13 m_c + m_p)} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{13}{13 m_c + m_p} \\ 0 \\ \frac{-12}{l(13 m_c + m_p)} \end{bmatrix}. \tag{6.3}$$

The second step consists of finding the optimal control policy, u^*, subject to (6.2) that minimizes the cost function,

$$J(u) = \int_0^\infty \left(X^T Q X + u^T R u\right) dt, \tag{6.4}$$

where R and Q are arbitrary positive defined matrices in which we can assign weights to state space variables and control signal. In control theory this kind of problem formulation is known as Linear-Quadratic-Regulator (LQR) [83].
The optimal control policy then can be defined by solving the Algebraic Riccati Equation [83] as:

$$\begin{aligned} &A^T P + PA - PBR^{-1}B^T P + Q = 0, \\ &K^* = R^{-1}B^T P, \\ &u^* = K^* X. \end{aligned} \tag{6.5}$$

For (A, B) controllable, the infinite horizon LQR with $Q, R > 0$ gives a convergent closed-loop system [83], where the stability can be easily guaranteed.

6.1.3. Networked Control Model

As defined in [84], we adopt a similar NCS model to define the system behavior over the wireless medium operating in Frequency Division Duplexing (FDD) mode with separated frequency bands for the uplink (UL) and downlink (DL) directions, which makes the medium access for UL and DL independent from each other in time domain. Figure 6.2 illustrates the proposed model, showing the details of interaction between the communication and application control loop. First, the sensor readings of the application describe the system states, X_i, which are stored in memory and communicated to the controller over the uplink channel. The readings and transmissions of sensor values are done strictly periodically with the cycle time ΔT_{in}, as it is commonly done across various control systems [85].
At the controller, the received sensor values are also stored into the memory. The control application gets the recent values, and produces a control signal u_i according to (6.5). Immediately after producing a control command, the controller sends it over a downlink channel to the controlled system. After finishing the current transmission, the controller keeps waiting for the next state update from the controlled device, and starts the procedure once again.
At the controlled system side, the received command u_i is stored in the memory. The output application for actuators control (e.g., motor drives) is called periodically with the time interval ΔT_{out}, calls the most recently stored command values from the memory and applies them to the application drives, producing the system dynamics of (6.1).

6.1.3.1. Wireless channel model

Both the DL and UL transmissions can suffer latency while delivering the information, which, in this model, depends on two main factors: the current channel quality and the total bandwidth allocated for the transmission. To evaluate this behavior, we consider the 3GPP 4-bit CQI Table 7.2.3-1 [86], where we can estimate the amount of time to deliver the data considering both the channel quality indicator (CQI) and the total bandwidth allocated at the transmission. The following two assumptions have been adopted: a) the UL finishes its transmission within ΔT_{in}, and b) the DL only starts a new transmission after finishing the

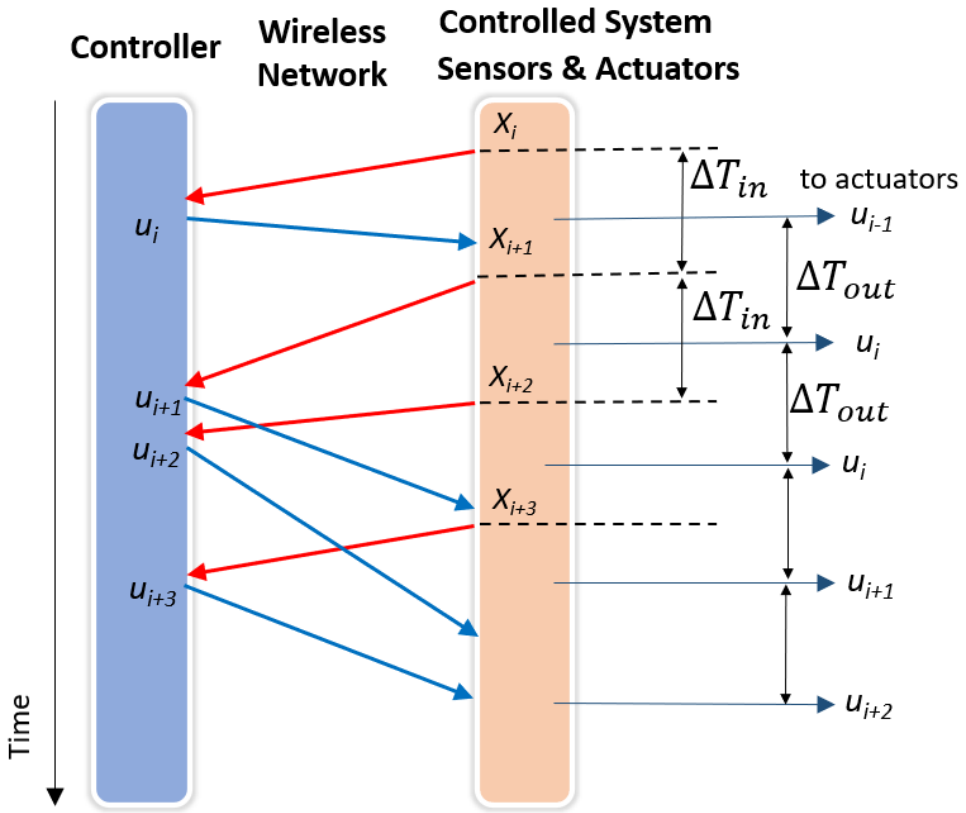


Fig. 6.2: WNCS Model.

current one. The details of the bandwidth allocation problem are discussed in the next section.

6.1.3.2. System Model Discussion

It is important to emphasize that (6.4) is guaranteed to be bounded according to the Riccati-equation [83]. However, the combination of two main factors might affect the system LQR performance. The first is the uplink effect, which represents the level of knowledge the controller has about the plant, meaning that, if ΔT_{in} is too high or the uplink takes overly long to deliver sensor data, the controller will compute the control signal using old state feedback, causing the control command to be ineffective or even harmful for the plant. The second is the downlink effect, or simply the overall delay to deliver the control signal. This is important because if the plant applies outdated control commands for too long, the stability of the controlled system might also be compromised.

Each of these factors might affect the plant in different ways and cannot be independently decoupled, which means that a delay in the UL will impact the DL transmission, provoking cumulative effects at the plant and at the network resources.

6.2. Age of Information and Age of Loop

Age of information (AoI) provides a measure for quantifying the freshness of the knowledge we have about the status of a remote system. It represents the time duration between the generation time of the freshest received data and the current time. We can refer to its formal definition as in [78, 87], where, at time t, if the newest data (i.e., with the largest generation time) received at the destination was generated at time $U(t)$, the AoI $\Delta(t)$ is defined as $\Delta(t) = t - U(t)$.
The formal AoI definition, however, is inherited to a single communication link. Papers which so far explored WNCS related problems using AoI are limited to specific analysis over only the UL [80, 88, 89] or DL [79, 90] transmissions. However, wireless networked control systems, as the one considered in this paper, rely intrinsically on both DL and UL with a closed-loop, where the UL communication can affect the DL and vice-versa, impacting system performance and the use of network resources. A simple intuitive example that can illustrate this idea is that a high UL AoI implicates less knowledge that the controller has about the plant, which demands more urgency to deliver the control signal and, as a consequence, more network resources usage by the DL link. To address this implications, we propose a new metric to evaluate the overall age of a WNCS closed-loop, which we refer to as Age-of-Loop (AoL).
Specifically, we can first verify that the state values X_i are periodically generated and transmitted at time intervals of $t_i = \{i \cdot \Delta T_{\text{in}}\}, \forall i \in \mathbb{N}^+$, where we can define $\{X_i, t_i\}$ the sequence of generated state values and its respective time step. The control signal, in turn, is asynchronous and must finish the current DL transmission to start a new one upon reception of a new status update. We can define a sequence $\{u_j, \hat{t}_j\}\, \forall j \in \mathbb{N}^+$ with aperiodically generated control commands u_j at time step $\hat{t}_j$. If $\{X_i, t_i\}$ is the freshest state feedback that spawned a new control signal, we can extend the DL transmission definition to include state time information, i.e., DL : $\{u_j, \hat{t}_j, t_i\}$. Reciprocally, every state feedback also occurs under the input of the freshest control command, so that we can also extend the UL transmission definition to include control time information, i.e., UL : $\{X_i, t_i, \hat{t}_j\}$.
We consider two plausible definitions of the AoL depending on the selected time origin: the DL-AoI is DL-initiated, meaning that the time origin is a new control command; the UL-AoI is UL-initiated, i.e., the time origin is a new status update in the UL. The DL AoL metric captures the time elapsed since the control command that led to the latest received update in the controller was generated. Analogously, the UL AoL metric refers to the time elapsed since the status update that caused the latest applied control command was generated at the sensor. Mathematically, if the origin is the DL, the current AoL is the difference between the current time t and the time when the freshest received control command was generated:

$$\text{DL AoL}(t) = t - \hat{t}_j. \tag{6.6}$$

Likewise, if the time origin is the UL, the AoL is calculated as the difference between the current time and the time when the freshest received state was spawned:

$$\text{UL AoL}(t) = t - t_i. \tag{6.7}$$

Essentially, the main idea of AoL is to establish a metric that encompasses the behavior of two separated and locally measured entities (DL and UL) into a single instance, in which we can observe from different perspectives. It is important to note that, in the case of two

independent AoI links, we inherently need an instantaneous and perfect feedback channel to the source to know the instantaneous age at the destination, thus making complex and potentially imprecise the union of two directions; AoL fixes this. In practice, it also offers the possibility to design solutions that enclose the whole closed-loop behavior by checking the loop age from either an UL or DL perspective. For example, we can potentially design a power allocation policy for the UL by observing the current UL AoL status. Likewise, we are able to define a modulation coding scheme algorithm for the DL transmissions by observing the DL AoL. It will be proven that they are both valid to optimize the stability of the WNCS. To illustrate the proposed concept, Figure 6.3 shows a representative time diagram of the AoL behavior.

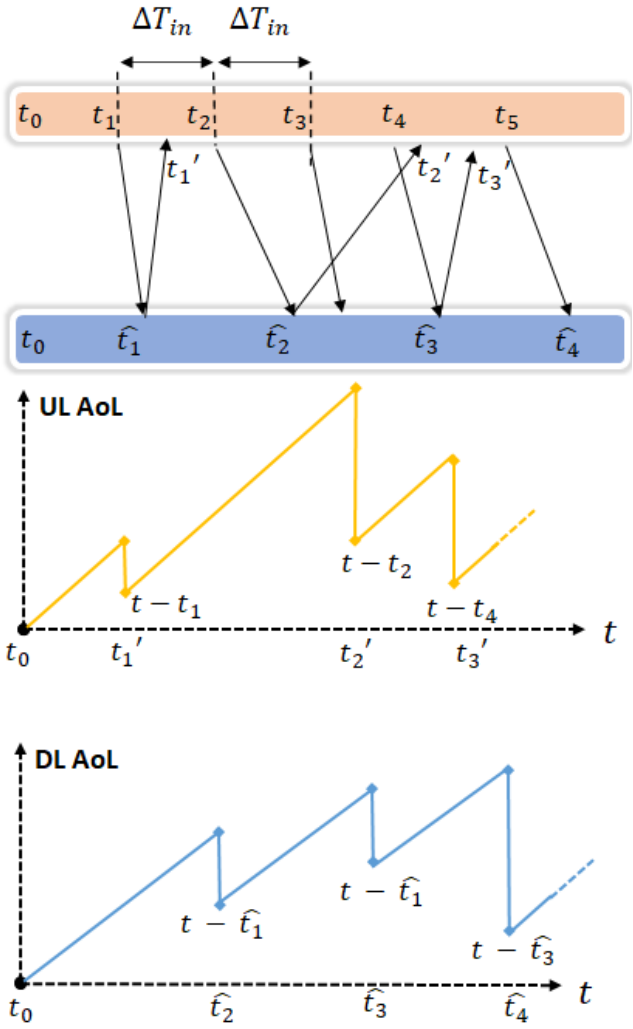


Fig. 6.3: Time Diagram of AoL Behavior.

6.3. Bandwidth Allocation Problem using Age of Loop (AoL)

The AoL status of a WNCS can provide an estimation of the system LQR performance, so that we can use the learned value function to build a policy. In this work, we explore the

bandwidth allocation problem of a remote controller, where two main objectives must be satisfied: minimize the LQR cost while using the minimum amount of bandwidth.
In more details, we can define $B = \{b_1, b_2, \dots, b_i, \dots, b_N\}, b_{i+1} > b_i$ a set of bandwidths in which the controller, for every DL transmission, must decide for a certain bandwidth allocation $b \in B$ given the current AoL state information and the current channel quality. So, for $T = \{t_1, t_2, \dots, t_i, \dots t_N\}\ t_{i+1} > t_i$ the time instances where control packets starts transmission and $C = \{c_1, c_2, \dots, c_i, \dots, c_N\}$ the corresponding CQI of each transmission, the goal is to find an allocation policy $\pi : \{\Delta_{\text{AoL}}(t_i), c_i \rightarrow b_i\}, \forall t_i \in T, \forall c_i \in C, \forall b_i \in B$ that minimizes the infinite-horizon LQR cost plus the amount of bandwidth usage over the system trajectory, i.e.,

$$\pi^* = \arg\min_{\pi} \left(\int_0^{\infty} (X^T Q X + u^T R u) dt + \sum_{i=1}^{N} \frac{b_i}{b_N} \right),$$
$$\text{s.t. (6.1), (6.5).} \tag{6.8}$$

6.3.1. Solution Proposal

We can decompose the problem in (6.8) into sub-problems, where between two consecutive control transmissions $[t_i, t_{i+1}), \forall t_i \in T$, we select at t_i a bandwidth $b_i \in B$ based on the AoL and CQI state $\{\Delta_{\text{AoL}}(t_i), c_i\}$. Receiving, as consequence, a one-stage decision cost of:

$$\int_{t_i}^{t_{i+1}} (X^T Q X + u^T R u) dt + \frac{b_i}{b_N}, \tag{6.9}$$

which depends only on the present state and the decision taken on that state. Such decision-making model is a typical Markov Decision Proces (MDP) [91], where we can optimally solve each sub-problem with actual state transitions and overlap those solutions to build the overall optimal solution. In this context, we can define the following MDP configuration:

- **State Space:** Comprised of 20 AoL values, each representing regions of 5 ms from 0 to 100 ms. In addition, 15 possible CQI values for each AoL, resulting in a total of 300 states.
- **Action Space** Represented by the bandwidth set with ten possible values: $B = \{100, 200, 300, \dots, 1000\}$ kHz.
- **Reward** The immediate cost as defined in (6.9).
- **Scenario** We evaluate the proposed MDP considering the NCS model described in section 6.1, assuming the following inverted pendulum configuration: $m_c = 1.0$ kg, $m_p = 0.1$ kg, $l = 0.5$ m, $g = 9.8$ m/s^2, control packet size of 1024 bits and $\Delta T_{\text{out}} = 1$ ms. For each run, the CQI is randomly chosen $\{1, 2, 3, \dots, 15\}$. The evaluation is also performed under different sensor feedback $\Delta T_{\text{in}} = 1, 5, 10, 15$ and 20 ms.

To solve the proposed MDP, we advocate a RL methodology for two main reasons. First, the MDP transitions probabilities are not easily tractable since the AoL variation will simultaneously depend on the channel and resource allocation of both UL and DL links. So, the UL behavior might be analytically unpredictable from the DL perspective and vice-versa.

Second, learning a value function from the AoL states means that we have a prediction of system performance given the current AoL condition. In other words, this methodology offers the possibility for the network to essentially learn the control system behavior, where the bandwidth allocation policy is just one of multiple network functions in which it can serve. We could easily extend the learned values to find optimal polices, for example, in terms of packet length, power allocation, antenna configuration and so on.
Hence, we solved the proposed MDP by applying a TD RL algorithm, based on a ϵ-greedy decision making during training, with exponential learning and exploration rate decay [91].

6.4. Results

We compare the proposed solution with a bandwidth allocation scheme based on predefined delay requirements, which is the general solution currently used in industry. In more details, given an arbitrary requirement of T_r ms for the control packet to be delivered, we can directly calculate the minimum amount of bandwidth to achieve the necessary requirement using the 3GPP 4-bit CQI Table 7.2.3-1 [86] and the total packet size. These baseline approaches, as well as the RL scheme, were evaluated on the scenario described in Section 6.1.
We analyze the results for three common network requirements, $T_r = 1$ ms, $T_r = 5$ ms and $T_r = 10$ ms. In each case, we analyzed the total bandwidth usage and the total LQR cost, which are respectively illustrated in Figure 6.4 and Figure 6.5.

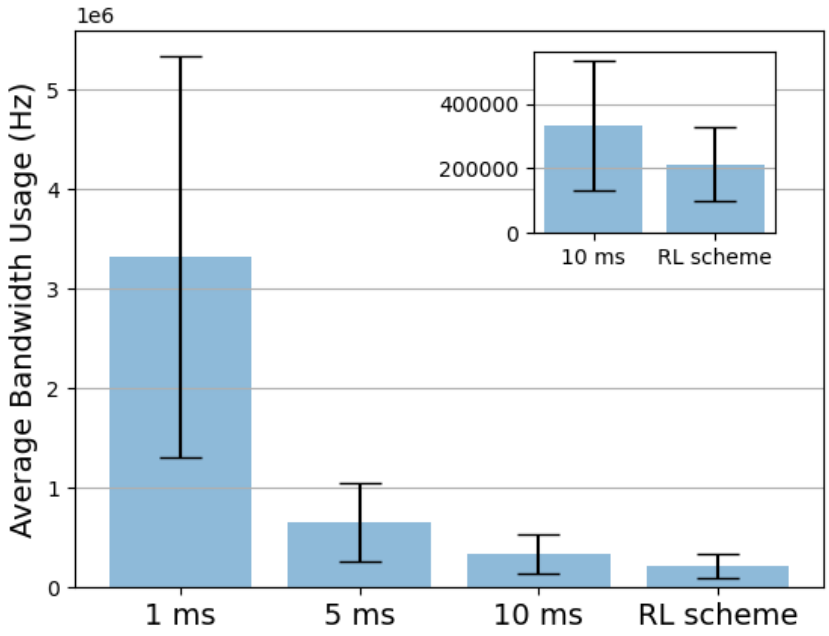


Fig. 6.4: Total amount of bandwidth usage for each method.

The immediate conclusion we can verify is that the RL scheme was capable to learn the system delay requirement, such that we can relate a slight increase at the LQR cost, which does not seriously affect the plant general performance, in exchange for more bandwidth saving.

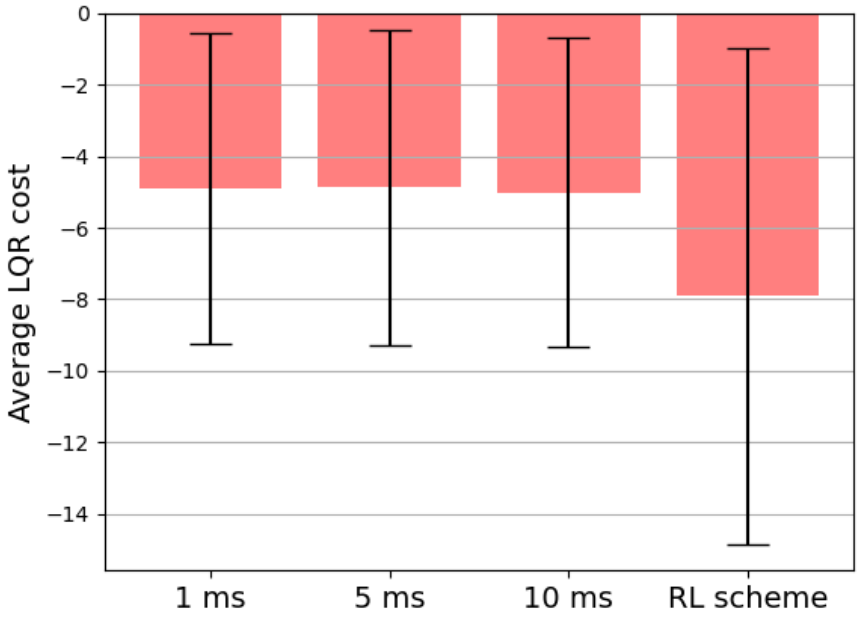


Fig. 6.5: Total amount of LQR cost for each method.

The second conclusion is that, as expected, strict latency requirement (T_r = 1 ms) demands more bandwidth usage. Compared to T_r = 10 ms, however, the RL scheme could still save 36% more bandwidth, which is an indication that 10 ms is still a sub-optimal requirement, but we can learn it from the RL algorithm.

6.5. Final Remarks

The information contained in the present chapter was essentially used and extended to produce a scientific contribution submitted and accepted at the 2021 IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (IEEE PIMRC 2021), which has become one of the two major conferences of the IEEE Communications Society (ComSoc) in the field of wireless communications and networking.

7. Conclusion

Hierarchical and distributed machine learning techniques are still an active field of research in wireless communications systems as they can efficiently solve the big optimization problem by dividing it into simpler sub-problems and solving them by different autonomous solvers. Here each device can operate as an autonomous and intelligent agent where they can learn and operates independently and/or collaboratively by sharing the information among different agents. This increases the system's reliability by defining multiple operational agents and also improves security by reducing sensitive data exchange between devices and central decision-makers. Still, some serious issues and challenges need to be addressed before employing a fully decentralized approach in real-world operational systems. Finding an optimal solution in a fully decentralized fashion is not easy, as the solutions may even diverge in multi-objective optimization scenarios with multiple conflicting goals. So more advanced techniques need to be introduced to deal with this increased complexity.
In this deliverable, we first provide an overview of the different architectures of hierarchical and decentralized machine learning architectures. In Chapter 2 we discussed different Hierarchical Reinforcement Learning (HRL) systems where they divide the original problem into multiple layers of hierarchies and solve them in different abstract layers. Then in Chapter 3 we discussed the framework of decentralized machine learning as a key enabler for edge intelligence. We focused on the important notions of solution's robustness and the available tools to reduce the amount of information to share in the network for converging to a solution. We next introduced AD-GDA, an algorithm that is shown to produce fair predictors using a compressed communication.
Then we took a look at multi-objective optimization problems and discussed some use-cases and applications of these techniques in wireless communications, which can be addressed by hierarchical and/or distributed learning. We discussed different wireless sensing techniques in Chapter 4 and then introduced an Millimeter Wave (mmWave) radar-based gesture recognition system that uses a hierarchical learning scheme to capture spatial features of radar-generated point clouds through a set of layers and then fuse the spatial features to capture the temporal evolution of the gesture. Next, in Chapter 5, we present a multi-objective minimization problem for RIS assisted UAV communication to provide sustainable connectivity in semi-urban/rural areas. We devised an optimization problem that determines the UAV trajectory, RIS phase, and UAV transmission power while ensuring a ground UEs receive a guaranteed rate. We then employed SCA to find the optimal solution. Finally, in Chapter 6, we addressed multi-objective problems involving control and communication, where the idea was to find a suitable network configuration capable of optimizing the control performance while saving network resources.
The applications mentioned above are only a few problems that can be expressed as multi-objective optimization problems and possibly solved by hierarchical/distributed learning techniques. Hierarchical and distributed learning techniques are still under question, and they have a long way to go before employing them in a real-world operational system.

8. References

[1] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *International Conference on Machine Learning*, pp. 3540–3549, PMLR, 2017.

[2] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[3] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3072–3108, 2019.

[4] S. M. Aldossari and K.-C. Chen, "Machine learning for wireless communication channel modeling: An overview," *Wireless Personal Communications*, vol. 106, no. 1, pp. 41–70, 2019.

[5] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3039–3071, 2019.

[6] B. Hengst, *Hierarchical Reinforcement Learning*, pp. 495–502. Boston, MA: Springer US, 2010.

[7] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[8] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," *Advances in neural information processing systems*, pp. 1043–1049, 1998.

[9] P. Dayan and G. Hinton, "Feudal reinforcement learning," *NIPS'93*, 1993.

[10] T. G. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *Journal of artificial intelligence research*, vol. 13, pp. 227–303, 2000.

[11] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, *et al.*, "Policy gradient methods for reinforcement learning with function approximation," in *NIPs*, vol. 99, pp. 1057–1063, Citeseer, 1999.

[12] A. C. Li, C. Florensa, I. Clavera, and P. Abbeel, "Sub-policy adaptation for hierarchical reinforcement learning," *arXiv preprint arXiv:1906.05862*, 2019.

[13] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass, "Hierarchical deep reinforcement learning for continuous action control," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 11, pp. 5174–5184, 2018.

[14] F. Röder, M. Eppe, P. D. Nguyen, and S. Wermter, "Curious hierarchical actor-critic reinforcement learning," in *International Conference on Artificial Neural Networks*, pp. 408–419, Springer, 2020.

[15] Z. Cao and C.-T. Lin, "Reinforcement learning from hierarchical critics," *arXiv preprint arXiv:1902.03079*, 2019.

[16] C. Florensa, Y. Duan, and P. Abbeel, "Stochastic neural networks for hierarchical reinforcement learning," *arXiv preprint arXiv:1704.03012*, 2017.

[17] J. Rafati and D. C. Noelle, "Learning representations in model-free hierarchical reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 10009–10010, 2019.

[18] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *arXiv preprint arXiv:1805.08296*, 2018.

[19] A. Levy, G. Konidaris, R. Platt, and K. Saenko, "Learning multi-level hierarchies with hindsight," *arXiv preprint arXiv:1712.00948*, 2017.

[20] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *arXiv preprint arXiv:1604.06057*, 2016.

[21] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman, "Meta learning shared hierarchies," *arXiv preprint arXiv:1710.09767*, 2017.

[22] A. Pashevich, D. Hafner, J. Davidson, R. Sukthankar, and C. Schmid, "Modulated policy hierarchies," *arXiv preprint arXiv:1812.00025*, 2018.

[23] X. Chen, H. Zhang, T. Chen, and M. Lasanen, "Improving energy efficiency in green femtocell networks: A hierarchical reinforcement learning framework," in *2013 IEEE International Conference on Communications (ICC)*, pp. 2241–2245, IEEE, 2013.

[24] Y. Zou, Y. Xie, C. Zhang, S. Gong, D. T. Hoang, and D. Niyato, "Optimization-driven hierarchical deep reinforcement learning for hybrid relaying communications," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2020.

[25] Y. Geng, E. Liu, R. Wang, and Y. Liu, "Hierarchical reinforcement learning for relay selection and power optimization in two-hop cooperative relay network," *arXiv preprint arXiv:2011.04891*, 2020.

[26] S. Liu, J. Wu, and J. He, "Dynamic multichannel sensing in cognitive radio: Hierarchical reinforcement learning," *IEEE Access*, vol. 9, pp. 25473–25481, 2021.

[27] Y. Li, Y. Xu, Y. Xu, X. Liu, X. Wang, W. Li, and A. Anpalagan, "Dynamic spectrum anti-jamming in broadband communications: A hierarchical deep reinforcement learning approach," *IEEE Wireless Communications Letters*, vol. 9, no. 10, pp. 1616–1619, 2020.

[28] Y. Zhang, Z. Mou, F. Gao, L. Xing, J. Jiang, and Z. Han, "Hierarchical deep reinforcement learning for backscattering data collection with multiple uavs," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3786–3800, 2020.

[29] J. Chen and A. H. Sayed, "Diffusion adaptation strategies for distributed optimization and learning over networks," *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4289–4305, 2012.

[30] F. Yan, S. Sundaram, S. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2483–2493, 2012.

[31] J. N. Tsitsiklis, "Problems in decentralized decision making and computation.," tech. rep., Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, 1984.

[32] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE transactions on automatic control*, vol. 31, no. 9, pp. 803–812, 1986.

[33] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[34] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[35] Q. Ling, Z. Wen, and W. Yin, "Decentralized jointly sparse optimization by reweighted ℓ_q minimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1165–1170, 2012.

[36] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *arXiv preprint arXiv:1705.09056*, 2017.

[37] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," *arXiv preprint arXiv:1809.07599*, 2018.

[38] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017.

[39] D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli, "The convergence of sparsified gradient methods," *arXiv preprint arXiv:1809.10505*, 2018.

[40] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1709–1720, 2017.

[41] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signsgd: Compressed optimisation for non-convex problems," in *International Conference on Machine Learning*, pp. 560–569, PMLR, 2018.

[42] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *International Conference on Machine Learning*, pp. 3478–3487, PMLR, 2019.

[43] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," *arXiv preprint arXiv:1710.09854*, 2017.

[44] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," *arXiv preprint arXiv:1907.09356*, 2019.

[45] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[46] S. U. Stich, "Local sgd converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.

[47] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5693–5700, 2019.

[48] N. Singh, D. Data, J. George, and S. Diggavi, "Squarm-sgd: Communication-efficient momentum sgd for decentralized optimization," *arXiv preprint arXiv:2005.07041*, 2020.

[49] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*, pp. 4615–4625, PMLR, 2019.

[50] P. M. Esfahani and D. Kuhn, "Data-driven distributionally robust optimization using the wasserstein metric: Performance guarantees and tractable reformulations," *Mathematical Programming*, vol. 171, no. 1, pp. 115–166, 2018.

[51] J. C. Duchi, T. Hashimoto, and H. Namkoong, "Distributionally robust losses against mixture covariate shifts," *Under review*, 2019.

[52] J. Duchi and H. Namkoong, "Learning models with uniform performance via distributionally robust optimization," *arXiv preprint arXiv:1810.08750*, 2018.

[53] Y. Deng, M. M. Kamani, and M. Mahdavi, "Distributionally robust federated averaging," *arXiv preprint arXiv:2102.12660*, 2021.

[54] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[55] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, PMLR, 2017.

[56] Y. Hu, F. Zhang, C. Wu, B. Wang, and K. R. Liu, "A wifi-based passive fall detection system," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1723–1727, IEEE, 2020.

[57] S. Palipana, D. Salami, L. A. Leiva, and S. Sigg, "Pantomime: Mid-air gesture recognition with sparse millimeter-wave radar point clouds," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 1, pp. 1–27, 2021.

[58] T. Nishio, Y. Koda, J. Park, M. Bennis, and K. Doppler, "When wireless communications meet computer vision in beyond 5g," *arXiv preprint arXiv:2010.06188*, 2020.

[59] M. Bocca, O. Kaltiokallio, N. Patwari, and S. Venkatasubramanian, "Multiple target tracking with rf sensor networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1787–1800, 2013.

[60] M. Nakatsuka, H. Iwatani, and J. Katto, "A study on passive crowd density estimation using wireless sensors," in *The 4th Intl. Conf. on Mobile Computing and Ubiquitous Networking (ICMU 2008)*, Citeseer, 2008.

[61] O. Kaltiokallio, H. Yiğitler, R. Jäntti, and N. Patwari, "Non-invasive respiration rate monitoring using a single cots tx-rx pair," in *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, pp. 59–69, IEEE, 2014.

[62] A. T. Parameswaran, M. I. Husain, S. Upadhyaya, *et al.*, "Is rssi a reliable parameter in sensor localization algorithms: An experimental study," in *Field failure data analysis workshop (F2DA09)*, vol. 5, IEEE, 2009.

[63] J. Liu, H. Liu, Y. Chen, Y. Wang, and C. Wang, "Wireless sensing for human activity: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1629–1645, 2019.

[64] Y. Wang, J. Liu, Y. Chen, M. Gruteser, J. Yang, and H. Liu, "E-eyes: device-free location-oriented activity identification using fine-grained wifi signatures," in *Proceedings of the 20th annual international conference on Mobile computing and networking*, pp. 617–628, 2014.

[65] B. Tan, K. Woodbridge, and K. Chetty, "Awireless passive radar system for real-time through-wall movement detection," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 5, pp. 2596–2603, 2016.

[66] B. Tan, K. Woodbridge, and K. Chetty, "A real-time high resolution passive wifi doppler-radar and its applications," in *2014 International Radar Conference*, pp. 1–6, IEEE, 2014.

[67] C. Iovescu and S. Rao, "The fundamentals of millimeter wave sensors," *Texas Instruments*, pp. 1–8, 2017.

[68] I. E. Sutherland, "Sketchpad: A man-machine graphical communication system," Tech. Rep. 296, Lincoln Laboratory, MIT, 1963.

[69] I. E. Sutherland, "The ultimate display," in *Proc. IFIP Congress*, pp. 506–508, 1965.

[70] G. Cohen, M. Hilario, H. Sax, S. Hugonnet, and A. Geissbuhler, "Learning from imbalanced data in surveillance of nosocomial infection," *Artif. Intell. Med.*, vol. 37, no. 1, pp. 7–18, 2006.

[71] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[72] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[73] B. Sheen, J. Yang, X. Feng, and M. M. U. Chowdhury, "A deep learning based modeling of reconfigurable intelligent surface assisted wireless communications for phase shift configuration," *IEEE Open Journal of the Communications Society*, 2021.

[74] K. Li, W. Ni, E. Tovar, and A. Jamalipour, "On-board deep q-network for uav-assisted online power transfer and data collection," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12215–12226, 2019.

[75] M. Yi, X. Wang, J. Liu, Y. Zhang, and B. Bai, "Deep reinforcement learning for fresh data collection in uav-assisted iot networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 716–721, IEEE, 2020.

[76] Y. Cai, Z. Wei, S. Hu, D. W. K. Ng, and J. Yuan, "Resource allocation for power-efficient irs-assisted uav communications," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–7, IEEE, 2020.

[77] E. D. Carvalho, A. Ali, A. Amiri, M. Angjelichinoski, and R. W. Heath, "Non-stationarities in extra-large-scale massive mimo," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 74–80, 2020.

[78] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Minimizing the age of information through queues," *IEEE Transactions on Information Theory*, 2019.

[79] W. Liu, G. Nair, Y. Li, D. Nesic, B. Vucetic, and H. V. Poor, "On the latency, rate and reliability tradeoff in wireless networked control systems for IIoT," *IEEE Internet of Things Journal*, 2020.

[80] K. Gatsis, H. Hassani, and G. J. Pappas, "Latency-reliability tradeoffs for state estimation," *IEEE Transactions on Automatic Control*, 2020.

[81] R. V. Florian, "Correct equations for the dynamics of the cart-pole system," *Center for Cognitive and Neural Studies (Coneural), Romania*, 2007.

[82] Z. M. Wang, D. F. Yang, K. Yang, L. Y. Guo, and J. M. Tan, *Machine Tool Technology, Mechatronics and Information Engineering*. TransTech Publications Ltd, 2014.

[83] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[84] P. M. de Sant Ana, N. Marchenko, P. Popovski, and B. Soret, "Wireless control of autonomous guided vehicle using reinforcement learning," in *IEEE Global Communications Conference*, 2020.

[85] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless network design for control systems: A survey," *IEEE Communications Surveys & Tutorials*, 2017.

[86] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures," Technical Specification (TS) 36.213, 3rd Generation Partnership Project (3GPP), 10 2014. Version 12.3.0.

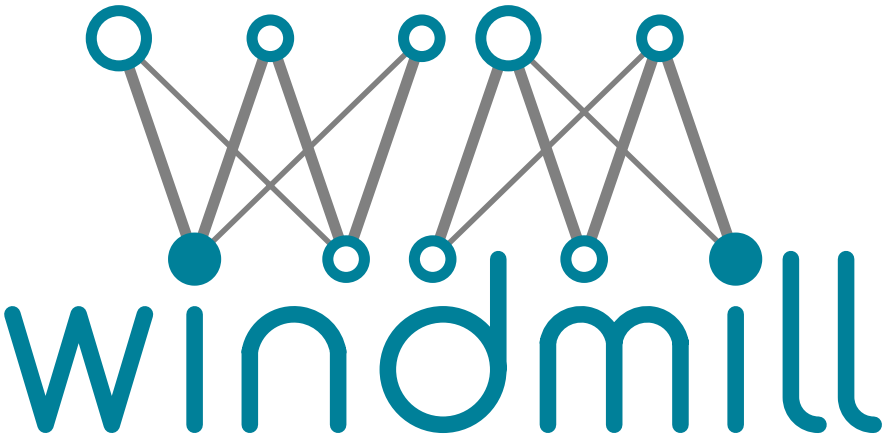
[87] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, 2021.

[88] J. P. Champati, M. H. Mamduhi, K. H. Johansson, and J. Gross, "Performance characterization using AoI in a single-loop networked control system," in *IEEE Conference on Computer Communications Workshops*, 2019.

[89] M. Klügel, M. H. Mamduhi, S. Hirche, and W. Kellerer, "AoI-penalty minimization for networked control systems with packet loss," in *IEEE Conference on Computer Communications Workshops*, 2019.

[90] Huang, Kang and Liu, Wanchun and Li, Yonghui and Savkin, Andrey and Vucetic, Branka, "Wireless feedback control with variable packet length for industrial IoT," *IEEE Wireless Communications Letters*, 2020.

[91] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.



www.windmill-itn.eu