SAPIENZA
UNIVERSITÀ DI ROMA

A Reinforcement Learning Approach to Demand Balancing and Tariff Optimization in Blockchain-Based Smart Water Networks

Facoltà di Ingegneria dell'informazione, informatica e statistica

Corso di Laurea Magistrale in Artificial Intelligence and Robotics

Candidate

Filippo Ansalone

ID number 1950936

Thesis Advisor

Prof. Ioannis Chatzigiannakis

Academic Year 2025/2026

A Reinforcement Learning Approach to Demand Balancing and Tariff Optimization in Blockchain-Based Smart Water Networks
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: ansalone.1950936@studenti.uniroma1.it

Abstract

The increasing complexity of Water Distribution Networks, together with growing demands for energy efficiency and operational sustainability calls for innovative approaches to optimize resource management and mitigate inefficiencies. This thesis presents a comprehensive framework for smart WDNs that integrates hydraulic simulation, blockchain based transactions, and Reinforcement Learning to enable adaptive and decentralized water management.

The proposed system combines hydraulic modeling through the WNTR library, built on top of EPANET, with blockchain based smart contracts developed in Solidity. At each simulation timestep, network events trigger on-chain transactions, enabling transparent and automated data collection used for tariff enforcement defined by water suppliers. Users are equipped with local storage tanks and can adapt their consumption patterns according to dynamically updated tariffs applied via smart contracts. By concentrating a big portion of the consumption toward lower tariff periods, users contribute to peak demand reduction and overall load balancing across the network.

To find optimal user behaviors, a Reinforcement Learning algorithm is integrated into the framework, interacting with the hydraulic simulation environment. Through iterative learning, the agent identifies consumption strategies that balance economic savings for users with operational stability for the water network.

The results demonstrate that distributing tariff and load driven consumption patterns can effectively mitigate demand peaks and improves overall energy efficiency. By combining RL, smart contracts, and IoT infrastructures, this work highlights a scalable approach to next-generation smart WDNs. The proposed framework addresses both user-centric objectives, such as cost minimization, and operator-centric goals, including demand smoothing and infrastructure protection, contributing to the advancement of intelligent and decentralized water management systems.

Contents

Chapter 1

Introduction

1.1 Motivation

Water scarcity has become one of the most pressing challenges for many regions worldwide. Rising temperatures, prolonged droughts, and increasingly irregular precipitation patterns are putting severe stress on existing water infrastructures. Mediterranean areas are particularly vulnerable, and territories such as the island of Sicily face recurring water shortages due to their arid climate and extended dry spells.

To deal with these problems, residential buildings in these regions often rely on small cisterns and private water tanks to buffer supply interruptions and manage daily consumption. While these decentralized storage solutions partially alleviate water scarcity issues, they introduce additional complexity into the already intricate structure of Water Distribution Networks. Recent studies have shown that existing infrastructures can be leveraged to improve both energy efficiency and water resource management, particularly by optimizing pumping schedules and storage utilization. However, integrating distributed storage systems into a coordinated, sustainable, and scalable management strategy remains an open research problem.

The challenge lies not only in the physical complexity of WDNs (characterized by nonlinear hydraulic dynamics, pressure constraints, and geographically distributed components) but also in the need to balance multiple, often conflicting, objectives. Water utilities must ensure reliability, minimize operational costs, reduce leakages caused by pressure fluctuations, and satisfy increasingly variable demand patterns.

In this context, Digital Twin technology can represent a promising paradigm. By continuously integrating real time measurements with simulation models, a DT of a WDN enables predictive analysis and scenario evaluation. However, effectively coordinating individual users consumption behaviors through feedback mechanisms while preserving autonomy and ensuring system optimality remains largely unexplored.

By leveraging the recent technological advancements, real-time monitoring of water usage opens new opportunities for advanced optimization and control strategies in order to make them more dynamic, adaptive, and data-driven.

This thesis is motivated by the need to design and validate an integrated framework capable of supporting sustainable water resource management in smart WDNs. The proposed system aims to enable the optimization of water allocation, balance demand peaks, and provide actionable feedback to end users. In addition, a key component of the framework is the integration of blockchain based smart contract transactions providing a transparent and automated mechanism to enforce dynamic tariff schemes, register water consumption, and guarantee trust.

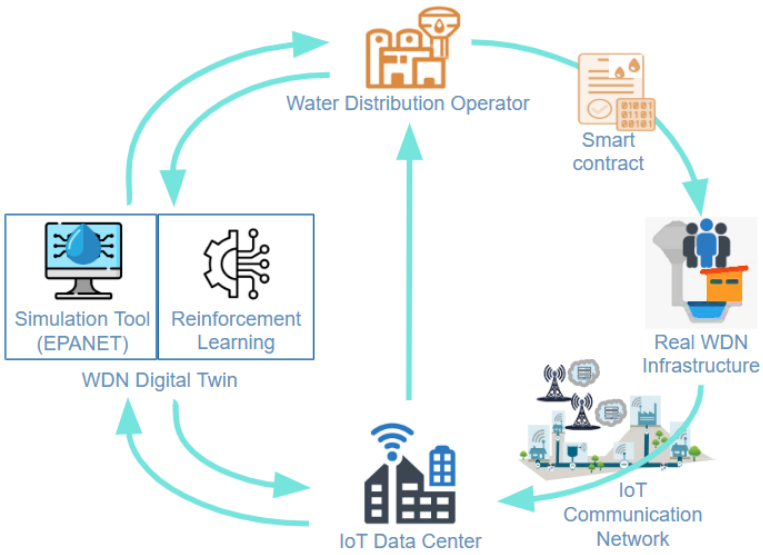

Figure 1.1. System infrastructure for smart WDN. DT processes the data related to the real WDN network with the EPANET simulation environment and the RL algorithm.

1.2 Thesis Objectives

The objective of this thesis is to design and evaluate an integrated framework for the intelligent management of water consumption in Water Distribution Networks. The proposed system aims to leverage recent technological advances such as smart metering, digital twins, blockchain infrastructures, and machine learning techniques to support more sustainable and efficient water resource management.

In particular, the work focuses on the development of a system capable of identifying optimized water consumption patterns for users while considering the physical constraints of the network. This is achieved by combining hydraulic simulations of the WDN with data-driven optimization techniques. A Digital Twin of the network is used to simulate the physical behavior of the system and evaluate the effects of different demand patterns on pressure stability and infrastructure reliability.

Another key objective is the integration of smart contracts to manage water related transactions and enforce pricing policies in a transparent and automated way. Through this mechanism, the system can associate economic incentives with consumption patterns, encouraging users to adopt behaviors that reduce demand peaks and improve overall network stability.

Finally, the thesis explores the use of reinforcement learning techniques, specifically Deep Q-Networks (DQN), to identify optimal consumption strategies that balance user demand, infrastructure constraints, and operational costs.

1.3 Achieved Results

The work presented in this thesis results in the implementation of a complete simulation pipeline that integrates hydraulic modeling, blockchain-based transaction simulation, and reinforcement learning–based optimization.

First, a realistic Water Distribution Network is modeled and simulated to generate physical data describing the evolution of water demand and pressure across the

network. These data are used to reproduce the operational conditions of the system and to evaluate the impact of different consumption patterns.

Second, a blockchain-based transaction simulator is implemented to model the economic layer of the system. Through smart contract logic, the simulator records water usage, computes associated costs, and applies transaction fees, enabling the representation of a decentralized and transparent accounting mechanism.

Finally, a Deep Q-Network agent is designed and trained to learn optimal allocations of consumption patterns among users. The learning agent interacts with the simulated environment and identifies configurations that improve the overall efficiency of the network.

The experimental evaluation demonstrates that the proposed framework is capable of effectively coordinating user consumption patterns, mitigating peak demand, and supporting more balanced network operation, showing the potential of combining digital twins, blockchain, and reinforcement learning for the management of modern smart water infrastructures.

The main results of this work have been published in [1] at IFIP, 2025, where the proposed framework integrating WDN simulation and reinforcement learning methodology is presented and evaluated.

1.4 Thesis Structure

This thesis is structured as follows:

- In Chapter 2 contains a comprehensive review of the relevant literature related to blockchain based water management systems involving the use of smart contracts, reinforcement learning and its applications in WDNs. It highlights previous works and current approaches showing the research gaps that motivated this work;

- In Chapter 3, the proposed simulation layer is introduced. First, the concept of a WDN is defined, describing the main elements that compose the system and their role within the hydraulic infrastructure. Then, WNTR is introduced as it is the simulation tool used in this work, it is employed to simulate the hydraulic behavior of the network starting from EPANET models. The chapter then describes the integration of a blockchain based layer into the framework, explaining how the results of the hydraulic simulations are used to trigger economic transactions. Finally, the concept of smart contracts is introduced, highlighting their role in defining rules for water related transactions within the proposed system;

- In Chapter 4 the reinforcement learning layer adopted for the water consumption optimization problem is presented. First, the problem is formalized as a Markov Decision Process, defining the state space, action space, reward function, and discount factor that define the interaction between the agent and the water distribution network environment. Then the Deep Q-Network approach is introduced as the learning method used to derive an effective policy for assigning consumption patterns to network nodes. The main components of the DQN algorithm are described, followed by the details of the implemented architecture and training procedure;

- In Chapter 5, the results of the proposed approach are presented. First, baseline strategies are introduced as reference solutions to evaluate the performance

of the method. Then, the experimental setup is described, followed by the analysis of the training results and the comparison with the baseline values.

- In Chapter 6, the main outcomes of this thesis are discussed. It reviews the problem addressed, the proposed methodology based on reinforcement learning and hydraulic simulation, and the main experimental results. The chapter also highlights the key contributions of the work, discusses the main limitations of the proposed approach, and outlines possible directions for future research.

Chapter 2

Previous and Related Works

2.1 Blockchain and WDNs

As previously discussed, many territories characterized by arid climates and extended periods of drought, such as the Sicilian island in Italy, address water scarcity by relying on small cisterns and storage tanks installed in residential buildings. These systems help support the management of household water demand during supply interruptions or shortages. Such solutions have been examined in the literature [2][3], and different approaches have been proposed to improve their efficiency, including strategies aimed at optimizing the associated energy costs. Existing studies suggest that the widespread presence of this distributed storage infrastructure can be leveraged not only to enhance water resource management but also to achieve potential energy savings within the urban water supply system [4]. Due to the complexity of Water Distribution Networks, providing sustainable solutions remains an open problem. As a result, existing infrastructures still struggle to meet the needs of both local residents and the continuously increasing tourist flows, which place additional seasonal pressure on already stressed water resources and distribution systems [5][6][7].

In the context of this work, the large-scale deployment of smart metering technologies across WDNs represents an approach to improve system monitoring and management [8]. Smart meters enable the collection of high resolution data on water consumption, providing real time information on usage patterns across the network [9]. This capability allows utilities and researchers to gain a deeper understanding of demand dynamics, detect anomalies such as leaks or abnormal consumption, and support the development of innovative strategies for more efficient and sustainable water resource management [10].

The use of water tanks has also been discussed as they could be used as intelligent storage that provide information about capacity [11].

Speaking of Water Distribution Networks and smart contracts, an important previous work has been [14]: this paper provides a structured and automated framework that allows stakeholders to define key system parameters (such as water sources, distribution nodes, consumption patterns, and contractual rules) through predefined templates and configurable contract logic. The tool automates essential processes, including water allocation, usage tracking, and penalty enforcement, while ensuring secure execution through blockchain integration. Additionally, the authors address practical concerns such as scalability, regulatory compliance, and interoperability, advancing the adoption of blockchain mechanisms for transparent and efficient water management. This represents the base of hydraulic and blockchain

simulations that have been extended in this work. [15] investigates the role of blockchain technologies in fostering citizen engagement within smart cities, with focus on urban water resource management. The authors present a proof of concept decentralized application in which a supervisory smart contract governs the execution of policy smart contracts based on community decisions. Since the blockchain can guarantee transparency, fairness, and secure access to data, the study highlights the potential of decentralized infrastructures to enable citizen participation to governance mechanisms in smart urban water systems. [16] contributes to the field of Blockchain Oriented Software Engineering (BOSE). The work is focused on Solidity design patterns trying to extend their description with additional fields to better support development decisions. Through their application to a water management use case, the paper demonstrates how enhanced design pattern documentation can guide developers in addressing critical aspects such as efficiency in smart contract design.

2.2 Markov Decision Process

In the context of Reinforcement Learning, a Markov Decision Process (MDP) [18] provides a formal framework for modeling sequential decision making problems under uncertainty. It is defined by a set of states, a set of actions, a transition probability function, and a reward function, which together describe the dynamics of the environment and the consequences of the agent's actions. Solving an MDP typically involves finding a policy that maximizes expected cumulative reward over time. This framework has become a foundational concept in reinforcement learning, supporting modern RL approaches such as Deep Q-Networks.

2.3 Deep Q-Network

Mnih et al. introduced DQN in [19] which marked a significant advance in RL by combining Q-learning with deep neural networks to achieve human level performances in complex control tasks. Key innovations in this work include the use of experience replay and a target network to stabilize learning and break the correlations between consecutive frames. This study has been the foundation for future research in combining deep learning with reinforcement learning, showing that neural networks can successfully approximate value functions in high-dimensional sequential decision making problems. From this point, several advancements and extensions have been proposed to improve the stability and performance of the original DQN. Double DQN addresses [20] the overestimation bias of Q-values of normal DQN by separating action selection from action evaluation, leading to more accurate value estimates. Dueling DQN [21] introduces a network architecture that separately estimates the state value and the advantage of each action, allowing the agent to better differentiate between how good are states and how much advantage can an action give. Another example is Prioritized Experience Replay [22] which improves DQN by sampling transitions from the replay buffer based on their temporal difference error, ensuring that samples with higher TD are revisited more frequently. Together, these enhancements form the foundation for many subsequent deep reinforcement learning algorithms and have been widely adopted in both research and applied settings.

2.3.1 DQN in WDNs

In general, Machine Learning techniques for WDNs have previously been reviewed [17] arguing that the enormous amount of data that IoT devices can extract could

indeed be used for predictive models, as happens in this work.

On the other hand, the application of DQN to WDNs [23][24][25] remains relatively limited in the existing literature and is mainly focused on pump scheduling and pump speed regulation, with the objective of reducing energy consumption while maintaining hydraulic constraints. On the other side, broader applications of DQN in WDNs such as demand-side management, tariff optimization, user behavior modeling, or integration with decentralized economic mechanisms are still largely unexplored. This highlights a research gap in leveraging deep reinforcement learning beyond purely hydraulic control toward more comprehensive smart water management strategies.

Chapter 3

Water Distribution Networks

The proposed system infrastructure is designed to support water distribution providers in implementing the proposed approach by integrating real-world data, advanced simulation tools, and IoT technologies. This infrastructure enables suppliers to optimize water allocation, balance demand, and improve the efficiency of the WDN. As visible in Fig.1.1, the system integrates real-time data from the real WDN and simulated data generated by the WDN digital twin, using a continuous data flow collected from the IoT network. This iterative process ensures that the network is constantly optimized and adapted to changing conditions, such as varying demand patterns, weather events, or infrastructure updates. The WDN, visible on the right side of the figure, includes pipelines, storage tanks, pumps, and smart meters that monitor water flow, pressure, and consumption in real time. This infrastructure is equipped with IoT-enabled sensors that, as shown in the data flow figure, form an IoT network that connects the physical infrastructure to the digital system. The smart meters deployed across the network continuously collect real-time data on water usage, pressure levels, and system performance. This data is transmitted to a central platform for processing and analysis, enabling the suppliers to monitor the network's status and mime it in simulation.

Within the DT model, there exists an uninterrupted, two-way exchange of data that allows for continual adjustments and tuning, predicated on the feedback drawn from the actual, physical system. This dynamic interaction serves to perpetually refine the system optimization and regulatory processes. Specifically, within the realm of WDNs, DT assumes a pivotal role in the rejuvenation of worn-out infrastructure. They significantly improve performance operations and offer solutions to pressing challenges, such as ensuring sustainability and enabling decisions informed by substantial data analysis. By harnessing these technological advantages, Digital Twins equip users with the ability to orchestrate intricate systems with improved effectiveness, identify and address irregularities preemptively, and make swift, data-driven decisions.

This work builds on a previous project by leveraging the SWIM (Smart Water Interaction & Monitoring) [13] platform which is an innovative solution that integrates DT technology, IoT devices interconnected via LoRaWAN, and ML techniques to support the integration of smart contracts in the current task. SWIM was developed to enable smarter and more efficient SWDN management, and now provides a solid foundation to extend control logic through blockchain enabled mechanisms. On top of it, SWIM incorporates machine learning models for anomaly detection, predictive maintenance, and system optimization, supporting more intelligent and proactive WDN management. It also interfaces with hydraulic simulation tools such as EPANET and WNTR. EPANET enables detailed simulation of pressurized

WDNs by modeling components such as pipes, pumps, and storage facilities while tracking hydraulic and water quality parameters. WNTR further extends these capabilities with Pressure-Driven Analysis (PDA) and a pressure-dependent leak model, offering more realistic representations of water demand and loss dynamics. SWIM DT is used to mirror the real infrastructure in a virtual environment. This

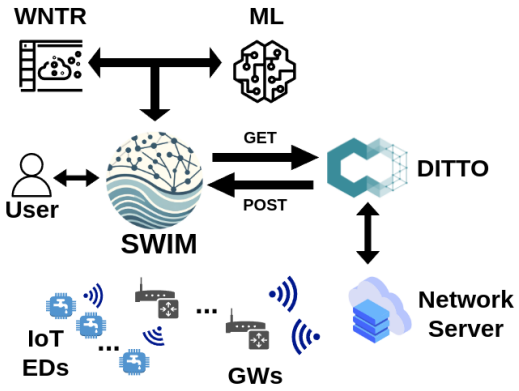

Figure 3.1. SWIM framework architecture.

DT is continuously updated with data from the IoT network, ensuring that the simulation accurately reflects the current state of the physical system. The present components, as machine learning algorithms, adapt to real-time data, learning from historical patterns, dynamically adjusting water allocation strategies to minimize costs, and balancing pressure picks in WDN infrastructure. It integrates a simulation framework with the WNTR/EPANET platform. In this system, EPANET is used to simulate multiple scenarios by various user consumption patterns, simulation results are used to feed ML algorithms, such as Reinforcement Learning (RL) model, specifically a Deep Q-Learning (DQN) based RL algorithm. Continuing to align with the flow shown in Figure 1.1, the water distribution provider uses the DQN results to derive optimized water allocation patterns. There exists a direct link between the data center and the supplier, facilitating the use of results to advise stakeholders, schedule maintenance, and address potential issues proactively. The proposed system introduces an innovative lifecycle that is adaptable to any WDN infrastructure. Integration is conveniently facilitated by loading a new EPANET model that mirrors the current WDN.

3.1 Definition

In the context of this thesis, the term Water Distribution Network (WDN) is used to refer to a comprehensive system composed of multiple interconnected components that ensure the supply and distribution of water. Specifically, a WDN in this work includes:

- a reservoir, which represents the source of water;
- pumps, to regulate and control water flow, are considered in this work either at fixed speed or absent from the network;
- nodes, representing demand points in the network.

These elements are connected by pipes, which enable the transport of water from sources to end users. With this definition in mind it will be a lot easier to model

WDNs in order to then study, simulate, and optimize water flows within them. EPANET [26] is a software developed by the United States Environmental Protection

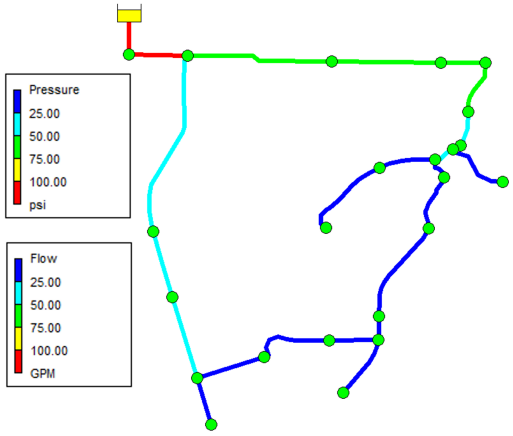

Figure 3.2. Schematic representation of the WDN topology used, comprising 23 nodes (junctions), 1 reservoir (top left), and 24 pipes.

Agency (EPA) used to model Water Distribution Networks (WDNs). It allows to simulate the flow of water through pipes, the operation of pumps and valves, the pressures and demand for every node over time. In the context of this thesis, EPANET has been used to create the WDN model by defining all the components to simulate. Once a network is designed, it is possible to export the model in the *.inp* file format, which is a standardized input file that encodes structural and operational information of the network. These *.inp* files can then be used by external simulation tools (as it happens in this work) to perform studies as, for example, time based analysis or data collection.

In particular, the WDN shown in figure 3.2 is created extracting a subnetwork from a previous model used in [12].

3.2 WNTR Simulator

Water Network Tool for Resilience (WNTR) [27][28] is a simulator developed by EPA, available as Python package, that is used with WDN models created with EPANET. In the context of Water Distribution Networks, WNTR and EPANET are indeed very useful tools as they can be used not only for hydraulic behavior testing but also for water quality monitoring (contaminant transport and decay), testing system resilience under particular conditions (for example pipe leakages) providing eventually a number of metrics about the simulations.

The following code block highlights an example usage of WNTR to perform some simulation steps on an imported WDN model:

```
import wntr

# load the WDN model
wn = wntr.network.WaterNetworkModel(model_path)

# set needed parameters
# this will specify the duration of each simulation step
wn.options.time.duration = sim_duration

# run simulation and gather data
```

```
for hour in sim_duration_hours:
    result = wntr.sim.WNTRSimulator(wn).run_sim()
    ...
```

In this work, a central component of the simulation framework is the module responsible for receiving and assigning consumption patterns to each node in the Water Distribution Network. This functionality is crucial for both main parts of the framework. In the first part, it enables the simulation engine to generate realistic water demand scenarios and then pass the resulting data to the blockchain module, where smart contract based transactions are executed. In the second part, it allows the RL environment to maintain an accurate record of the assigned patterns, which are then used as inputs for the WNTR simulator to evaluate the effects of different control strategies. The idea here is that a single simulator is used for both purposes, ensuring consistency between the RL environment and the blockchain transaction system. Following, a simplified code snippet showing how is this part managed:

```
def sim_step(patterns):
    ... # setup
    nodes = wn.junction_name_list
    for juncID in nodes:
        junc_obj = wn.get_node(juncID)
        junc_obj.add_demand(base=..., pattern_name=patterns[
                                            index_new_pattern])
    ...
    # run sim now
```

This mechanism disconnects the simulation from the water consumption pattern assignment policy.

The dataset summarized in Table 3.1 contains an example of data that the WDN simulation produces as output. Each row corresponds to a specific node and contains certain informations about it at a given hour, and the columns report the following information

- Hour: the simulation timestamp;
- Node ID: the identifier of the node in the network;
- Base demand: the amount of water requested by the node;
- Demand value: the actual water consumption resulting from the simulation, this is based on how much water was it possible to satisfy;
- Pressure value: the water pressure at the node;
- Type: the type of the current node (junction or reservoir).

These data are crucial for the blockchain simulator as they provide the actual consumption needed to execute smart contracts, calculate payments, and evaluate refunds. It is worth noting that the WNTR simulation framework is capable of producing a broader range of outputs beyond those reported in this example, however, they were not needed for this specific case.

3.3 Blockchain Integration

The integration of the blockchain layer into the proposed framework follows a sequential process. First, the hydraulic simulation of the Water Distribution

Table 3.1. Example of data produced by the Water Distribution Network physical simulation.

Hour	Node ID	Base Dem	Dem Value	Pressure Value	Type
0:00:00	8628	0.04	0.04	39.117	Junction
0:00:00	8630	0.013	0.013	39.117	Junction
0:00:00	8632	0.035	0.035	36.722	Junction
0:00:00	8634	0.042	0.042	32.443	Junction
0:00:00	8636	0.026	0.026	30.51	Junction
0:00:00	8638	0.002	0.002	29.011	Junction
0:00:00	8640	0.034	0.034	28.372	Junction
0:00:00	8642	0.006	0.006	28.499	Junction
0:00:00	8644	0.002	0.002	27.19	Junction
0:00:00	8646	0.024	0.024	25.442	Junction
0:00:00	8648	0.002	0.002	24.529	Junction
0:00:00	8650	0.035	0.035	23.843	Junction
0:00:00	8686	0.037	0.037	25.079	Junction
0:00:00	8688	0.029	0.027	18.992	Junction
0:00:00	8690	0.014	0.011	14.479	Junction
0:00:00	8692	0.043	0.027	10.493	Junction
0:00:00	8694	0.009	0.005	7.529	Junction
0:00:00	8696	0.032	0.013	6.334	Junction
0:00:00	8698	0.017	0.006	5.679	Junction
0:00:00	8700	0.044	0.011	4.649	Junction
0:00:00	8702	0.018	0.003	4.124	Junction
0:00:00	8738	0.027	0.004	3.933	Junction
...					

Network is executed for the next time period using the WNTR simulator. As seen before, this simulation produces an output file containing all relevant output data, including water consumption at each node and other hydraulic variables. The generated file is then provided as input to the blockchain simulator, which is responsible for executing the corresponding smart contract based transactions. For each user and timestamp, the blockchain module records and processes the amount of water consumed in order to calculate the total cost for the user according to the applicable tariff. This module eventually produces as output all the new balances, paid amounts, the transaction fee associated with the blockchain operations, together with additional user related information. In this way, the physical behavior of the network and the blockchain layer implemented through smart contracts are coupled, enabling a coherent simulation of both hydraulic dynamics and decentralized financial transactions.

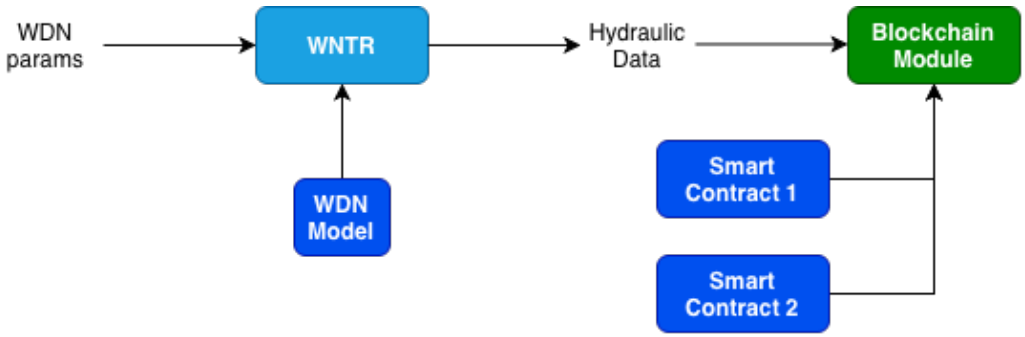

Figure 3.3. Flowchart of Water Network and Blockchain simulations.

It is important to also note that fees are related to each operation performed by the smart contracts, indeed they all consume a certain amount of gas which is basically a computational cost.

The integration of the blockchain layer is conceptually based on the assumption that the Water Distribution Network is equipped with interconnected IoT sensors positioned at key points, such as user nodes. These sensors are responsible for monitoring variables such as water consumption and pressure, and for transmitting the data to the digital infrastructure. As mentioned earlier, in this framework the data generated by the hydraulic simulation represent the information that would realistically be collected by such IoT devices.

3.3.1 Smart Contracts definition

A smart contract is a program deployed on a blockchain that is used to execute predefined actions once specified conditions are met. The main pro of using smart contracts, unlike more traditional approaches, is that their logic is stored on the blockchain, and their execution is validated by the network consensus mechanism. This ensures transparency and immutability as the contract's code and transaction history cannot be altered once deployed.

In the context of this work, smart contracts are implemented using Solidity, a high-level programming language specifically designed for developing contracts on Ethereum compatible blockchains. Defining a smart contract in Solidity means writing code that specifies the contract's variables, functions (for example payment execution or tariff updates) and other control rules. Once compiled and deployed, the Solidity contract becomes an autonomous component of the blockchain system,

capable of executing transactions and permanently recording the outcomes. In the context of this thesis, smart contracts formalize and automate the economic transactions associated with water consumption, ensuring that payments and fees are processed transparently and consistently with the simulation results.

Following, an example of how prices are calculated following a time varying schedule:

```
function getPrice(uint256 hour) public view returns (uint256) {
    uint256 nightPrice = baseDemand * k; # compute lower price
    if (hour >= 22 || hour <= 6) {
        return nightPrice;
    }
    return nightPrice * 2;
}
```

This example will correspond to a halved tariff during night hours:

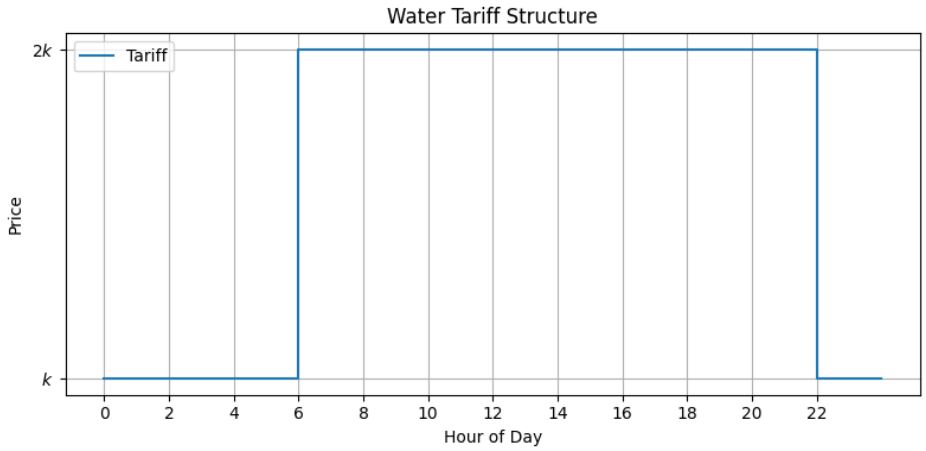

Figure 3.4. Applied tariff by hour.

It is important to note that smart contracts can implement significantly more dynamic and complex rules than the static example presented here, such as adaptive tariff schemes, incentive mechanisms, or conditional pricing policies based on other parameters. Indeed the flexibility of Solidity allows users to encode customized logic that can adapt to different regulatory or market scenarios. For instance, in the experiments conducted in this thesis, the implemented smart contracts included a refund mechanism that reimbursed users proportionally to the amount of unsatisfied water demand.

```
function getRefundValue(uint256 hour) public view returns (uint256) {
    if (satisfied) {
        return 0;
    }
    uint256 nightRefund = (baseDemand - demandValue) * k;
    if (hour >= 22 || hour <= 6) {
        return nightRefund;
    }
    return nightRefund * 2;
}
```

The code shown is used to calculate the refund ensuring that only the actually consumed water is paid by the user.

Moreover, when the blockchain simulator is initialized, it is possible to select which smart contract to use for the simulation among those previously defined by the

user. This design choice ensures modularity in order to make possible the evaluation of alternative configurations within the same simulation framework.

3.3.2 Blockchain Simulation Output

Table 3.2. Example of data produced by the Blockchain simulation.

Node ID	Balance	Usage	Blockchain Fee	Water Cost
8628	9.9334	2.6473	0.0043	0.0023
8630	9.9334	2.6473	0.0043	0.0023
8632	9.9333	2.7242	0.0043	0.0024
8634	9.9334	2.7242	0.0043	0.0024
8636	9.9333	2.6473	0.0043	0.0023
8638	9.9335	2.7242	0.0043	0.0024
8640	9.9333	2.7242	0.0043	0.0024
8642	9.9334	2.7242	0.0043	0.0024
8644	9.9335	2.6473	0.0043	0.0023
8646	9.9333	2.7242	0.0043	0.0024
8648	9.9333	2.6473	0.0043	0.0023
8650	9.9332	2.6473	0.0043	0.0023
8686	9.9333	2.7242	0.0043	0.0024
8688	9.9334	2.6473	0.0043	0.0023
8690	9.9334	2.7242	0.0043	0.0024
8692	9.9333	2.7242	0.0043	0.0024
8694	9.9333	2.7242	0.0043	0.0024
8696	9.9335	2.6473	0.0043	0.0023
8698	9.9335	2.6473	0.0043	0.0023
8700	9.9333	2.7242	0.0043	0.0024
8702	9.9331	2.7242	0.0043	0.0024
8738	9.9333	2.7242	0.0043	0.0024
...				

The output of the blockchain simulation is summarized in Table 3.2, which reports a sample of the data generated after processing the transactions associated with the physical network simulation. Each row corresponds to a specific node in the network and represents the financial outcomes resulting from the interaction between water usage data and the blockchain payment mechanism. The dataset contains the following columns:

- Node ID: the identifier of the node associated with the transaction;
- Balance: the final balance of the user after the execution of the transactions;
- Usage: the amount of water consumed by the node, derived from the physical simulation data;
- Blockchain fee: the transactions cost associated with executing the operations on the blockchain;
- Water cost: the amount paid for the consumed water according to the applied tariff.

These data represent the economic layer of the simulation, where the consumption information produced by the hydraulic model is translated into financial transactions. This log is eventually used to evaluate the economic implications of the proposed system, including payments for water usage and the operational costs introduced by blockchain transaction system.

Chapter 4

Demand Optimization

For optimizing WDNs it is necessary to define which water consumption patterns can be applied to various users and which tariffs are adopted. For the evaluation of the proposed approach, we use three representative consumption patterns that capture typical variations in water demand throughout the day and can be alternatively selected from the proposed system as user behavior. These patterns, illustrated in Fig. 5.3, are characterized by a peak consumption period in one-third of the 24-hour cycle, while the remaining two-thirds exhibit lower demand levels. The three patterns are differentiated by the timing of their peak: morning, afternoon, or evening. These consumption patterns serve as the basis for our optimization framework, where each node could be assigned a specific consumption behavior.

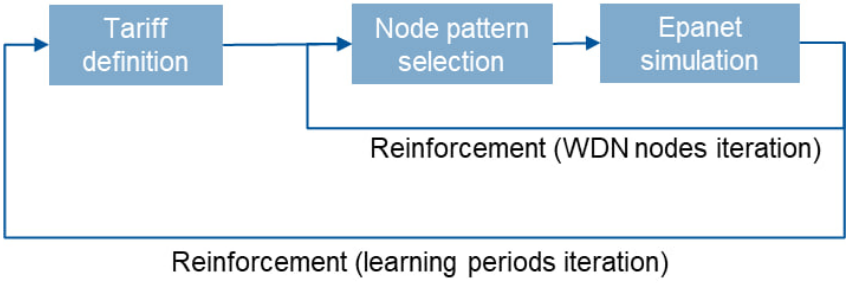

Figure 4.1. Schematic representation of the DQN framework. The process consists of two nested reinforcement loops: the inner loop selects consumption patterns and simulates their effects within the EPANET, while the outer loop integrates tariff definition.

In addition to consumption patterns, this optimization framework incorporates tariffs that can potentially influence the user pattern selection strategy. By aligning storage tank filling and consumption patterns with lower tariffs, the system promotes a more balanced distribution of demand throughout the day. We will focus on two tariff (shown in fig. 5.1) models that incentivize water consumption at specific times. Specifically, we consider a time-of-use pricing structure with two distinct price periods that resembles a two-hourly pricing scheme commonly used in utility pricing. Pricing functions are defined as square wave patterns, where higher rates apply during peak hours and lower rates encourage consumption during off-peak periods. The time slots of higher price are chosen as representative examples of a pricing scheme that incentivizes consumption shifting, encouraging users to fill storage tanks or adjust demand patterns to reduce costs while alleviating peak loads on the network.

4.1 MDP Formalization

Before proceeding to the DQN implementation, it is necessary to define the underlying Markov Decision Process (MDP) framework in which the agent operates. Reinforcement Learning problems are formally modeled as MDPs because they provide the mathematical structure required to describe sequential decision making under uncertainty. The main property of an MDP is the Markov property, which states that the probability of transitioning to the next state depends only on the current state and action, and not on the full history of previous states and actions. Formally:

$$P(S_{t+1} = s' | S_t = s, A_t = a, S_{t-1}, \ldots, S_0) = P(S_{t+1} = s' | S_t = s, A_t = a)$$

Defining the problem as an MDP is essential because modern reinforcement learning approaches are built assuming an MDP structure.

In general an MDP is defined as a tuple (S, A, P, R, γ) where:

- S is the State Space, consisting of the set of possible state of the environment;
- A is the Action Space, representing all the possible action that an agent can perform in the environment;
- $P(s'|s, a)$ is the transition probability function which, given the current state s, an action a and a possible next state s', returns the probability of transitioning to that next state from the current state performing that specific action;
- $R(s, a, s')$ is the reward function which returns the reward that the agent will receive when performing action a and transitioning from state s to state s';
- γ is a discount factor used to weight future rewards so that it is possible to determine how much future rewards are valued relative to immediate rewards, in other words, how long term based the decisions of the agent should be.

In the context of this work the MDP models the WDN environment in order to make it suitable to perform sequential decision making on it. Following the definition used in this experiment:

- A state s represents the current configuration of the water distribution network during the sequential assignment of consumption patterns to nodes
  - At the initial state the water network has no node consuming water (all demand values are set to zero);
  - At each decision step, the state evolves as nodes are progressively assigned to a consumption pattern.

  The observation space is represented by:
  - The water pressure values at all nodes in the network (contained in the vector p_k),
  - The currently applied tariff (vector τ_k),
  - The ID of the current node for which a consumption pattern must be selected (scalar i_k).

So formally, the observation fed as DQN input at step k can be expressed as:

$$o_k = (p_k, \tau_k, i_k)$$

In this way the agent has full observability of the hydraulic variables and economic context before selecting an action.

- The action space consists of all possible water consumption patterns available for assignment. At each step:
  - Performing an action corresponds to assigning one of the predefined consumption patterns to the current node.
  - Once assigned, the environment performs a new hydraulic simulation to recompute pressures that will be used in the next iteration, and transitions to the next node.
- The transition function P defines the dynamics of the environment which in this case is deterministic as when an action is taken, the state transition will reflect the physical dynamics of the water distribution network. We must note that this is only true because the agent has full observability of the environment.

  When a new action is taken, what happens is:

  1. The demand of the current node is set to the pattern correspondent to the action;
  2. The new WNTR simulation is performed producing as output the new state;
  3. The index i moves to the next node.

  A possible way of making this environment non deterministic at each step could be to randomly pick the index i and not simply increasing it, however this point was not covered in this work.
- The reward function is derived from a cost function that captures both the economic cost for water consumption and the penalty associated with unmet water demand.

  The cost associated with performing action a in state s is defined as:

  $$C(s, a) = \beta * PaidPrice + \delta * Penalization(s, a)$$

  where β and δ are weighting parameters, $PaidPrice$ is the amount actually paid for the supplied water (which can be fewer than the one requested) and the penalization term is calculated as:

  $$Penalization(s, a) = notSatisfiedWater_{s'} - notSatisfiedWater_s$$

  this second term therefore penalizes actions that increase the amount of unsatisfied water demand in the network.

  Since the objective of the reinforcement learning agent is to maximize cumulative reward, the reward function R is defined as the negative value of the cost:

  $$R(s, a) = -C(s, a)$$

- As said before, the discount factor $\gamma \in [0, 1]$ determines the relative importance of future rewards compared to immediate ones. In this experiments, it is set to $\gamma = 0.95$.

The WDN environment has been implemented using the Gym [29] library, a widely used tool in reinforcement learning because it provides a standardized interface for defining environments as Markov Decision Processes.

Following, an example of how can a sample WDN environment be implemented, showing only the main functions to create a basic MDP logic in Gym:

```
import gym

class WaterNetworkEnv(gym.Env):
    def __init__(self):
        self.nodes = get_nodes()
        self.num_users = len(self.nodes)
        self.patterns = [-1 for _ in range(self.num_users)]

    def reset(self):
        self.patterns = [-1 for _ in range(self.num_users)]
        self._parse_state(simulation_step(self.patterns))
        return self.state

    def step(self, i, pattern):
        self.patterns[i] = pattern
        ...
        self._parse_state(simulation_step(self.patterns))
        ...
        cost = ... # compute cost function
        return self.state, cost
```

4.2 DQN Approach

Deep Q-Networks (DQN) is an extension of the classical Q-learning. The main difference is that it uses a neural network to approximate the value function Q(s,a). The reason for doing this is that when the observation space is not discrete, having a Q-Table for value estimation is impracticable. On the other hand, a neural network can learn effective estimation policies directly from high dimensional state representations. More practically, with the neural network we want to approximate the Q-function Q(s,a) representing the expected cumulative discounted reward that an agent can obtain by taking action a in state s and then following the optimal policy in the following states.

$$Q(s, a) = \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n r_{t+n} | s_t = s, a_t = a\right]$$

Moreover, to ensure a better learning curve DQN introduces the following mechanisms:

- An experience replay buffer, a data structure in which past interactions with the environment are saved. This way, instead of updating the network using only the most recent transitions, the algorithm samples random batches from this buffer during training. This process breaks the strong temporal correlations between consecutive experiences and improves the efficiency and stability of learning;

- A target network, representing a separate neural network used to compute the target Q-values during training. The parameters of the target network are periodically updated by copying the weights of the main network. The reason for using this additional network is that by keeping the target network fixed for a number of training steps, the stability of training results higher.

4.2.1 Neural Network Architecture

The neural network used in this approach is designed to process different kind of information describing the current state of the water distribution network. As previously introduced in the MDP formulation, the input to the network consists of three components:

- Water pressures at all nodes, normalized to the $[0, 1]$ range using min–max normalization in order to improve training stability and ensure consistent scaling across inputs;
- A tariff vector, representing the currently applied tariff configuration and is composed exclusively of binary values (0 or 1);
- User (node) ID, represented as a scalar identifying the node for which the consumption pattern must be selected.

Given the different nature of these inputs, the network is structured with multiple processing branches before combining the informations.

First, a pressure processing branch is used to handle the vector of water pressures at the nodes of the network. This branch consists of four fully connected (linear) layers, each followed by ReLU activation functions. The primary purpose of this branch is to reduce the dimensionality of the pressure vector, extracting a more compact representation of the hydraulic state of the network that can be combined with the other input features in a more effective manner.

Second, the user ID is transformed into a dense representation through an embedding layer. This allows the model to learn a continuous vector representation of the node identifier, enabling the model to capture potential structural or behavioral similarities between different nodes in the network.

After these two steps, the inputs are now compressed and represented in a way that lets us concatenate them in a single vector that will serve as input for the last component of the network which is the one that computes Q-Values for each action.

The final stage of the model is anther MLP composed of three fully connected layers separated by ReLU activations. The output layer has a dimensionality equal to the size of the action space so that each output will represent the estimated Q-value associated with selecting that specific consumption pattern.

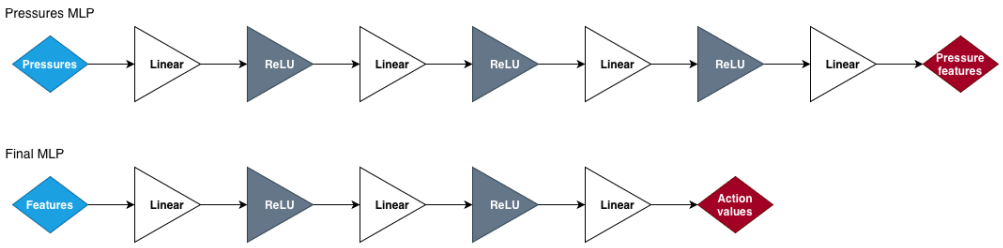

Figure 4.2. MLPs structures.

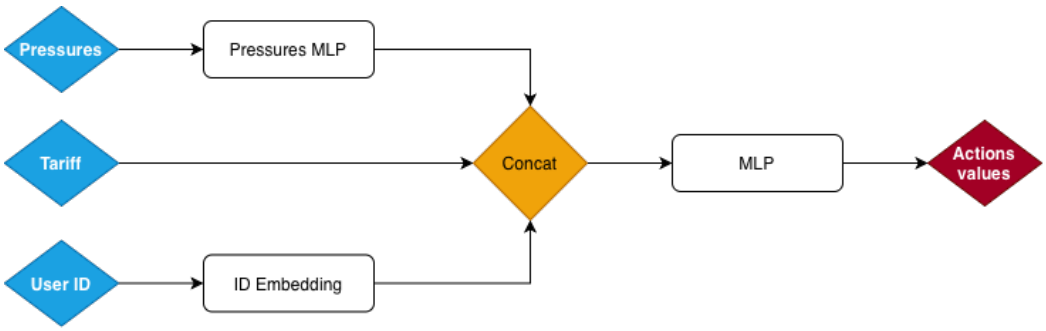

Figure 4.3. Neural network architecture.

4.2.2 DQN Implementation

As generally happens in Reinforcement Learning methodologies, exploration and exploitation have to be balanced in a certain manner in order to make the agent learn when it still doesn't have knowledge of the environment. The ϵ-greedy strategy is responsible of managing this trade off. Under this strategy, at each decision step the agent selects an action according to the following rule:

- with probability ϵ, the agent selects a random action from the action space (exploration);
- with probability $1-\epsilon$, the agent selects the action with the highest estimated Q-value according to the neural network (exploitation).

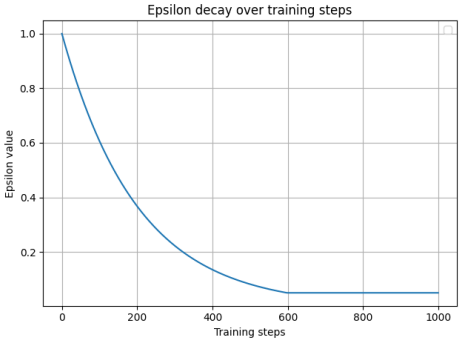

Figure 4.4. Epsilon decay over training steps

In other words this mechanism allows the agent to explore different consumption pattern assignments during training. At the beginning of training, a high value of ϵ encourages exploration of the action space. As training progresses, ϵ is gradually reduced so that the agent increasingly relies on the learned Q-value estimates to select actions that maximize the expected reward. The decay of ϵ is performed by multiplying it for a decaying constant γ_ϵ after each train step until it reaches a predefined minimum value, in this way some exploration is granted even after the model has learned a first strategy.

$$\epsilon = max(\epsilon * \gamma_\epsilon, \epsilon_{min})$$

The effective evolution of ϵ is shown in figure 4.4. After around 600 training steps the value of ϵ_{min}, set to 0.05 in this case, is reached. This way the agent will almost always rely on what it has learned but keeping to find new strategies in some scenarios.

With the ϵ-greedy strategy, actions can be selected with a logic as highlighted in the following code block:

```
def predict_action(self, ...):
    if np.random.rand() <= self.epsilon:
        return random.randint(0, self.num_patterns - 1)
```

```
    else:
        q_values = self.model(...)
        return np.argmax(q_values)
```

4.2.3 Training

The DQN agent is trained over multiple episodes where each episode corresponds to a full assignment of consumption patterns to all nodes in the water distribution network. At the beginning of each episode, the environment is reset to the initial state s_0 where no node has any assigned consumption and a new tariff is applied. The agent then iteratively follows:

- Observes the current state;
- Select an action following the ϵ-greedy precedure;
- Applies the new pattern to the current user, which will then start using water from the next iteration;
- Receives the correspondent reward;
- Observes the new state and updates the replay buffer, which in this work is implemented as a queue with max length of 2000 elements, by inserting the new transition as last item;
- Samples a batch from replay buffer and uses it to update the network weights, using the target Q-values computed from the target network.

The goal is to make the agent learn a policy that minimizes the overall cost so that both hydraulic and economic constraints are considered.

Once every 500 weights update steps, the neural networks weights are synchronized:

```
def update_target_network(self):
    self.target_model.load_state_dict(self.model.state_dict())
```

Moving on, the loss function used for training is the Mean Squared Error (MSE) between the predicted Q-value and the target value computed for each sampled transition. Given the mini-batch of the considered transitions (s_i, a_i, r_i, s_{i+1}) the loss is computed as:

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i))^2$$

where N is the batch size, $Q(s_i, a_i)$ is the Q-value predicted by the current network, and y_i is the target value computed as:

$$y_i = r_i + \gamma max_a Q_{target}(s_{i+1}, a)$$

Here, γ is the discount factor and Q_{target} is the target network, which, as shown before, is periodically updated from the main network to improve training stability.

The optimization of the network parameters is performed using the Adam optimizer, a stochastic gradient based optimization algorithm widely used in deep learning due to its ability to adapt the learning rate for each parameter during training. Adam combines the advantages of momentum and adaptive learning rate methods, providing faster convergence and improved stability when training deep neural networks.

4.3 Water Demand Patterns

Although relevant studies in the literature show that water demand in urban systems tends to follow statistically recurrent daily patterns characterized by multiple peaks associated with typical human activities, for this part of the work a simplified representation of consumption behavior has been adopted. Specifically, three demand patterns are considered, each characterized by the same total water consumption over the day and by an identical profile shape. The only difference among them is a temporal shift.

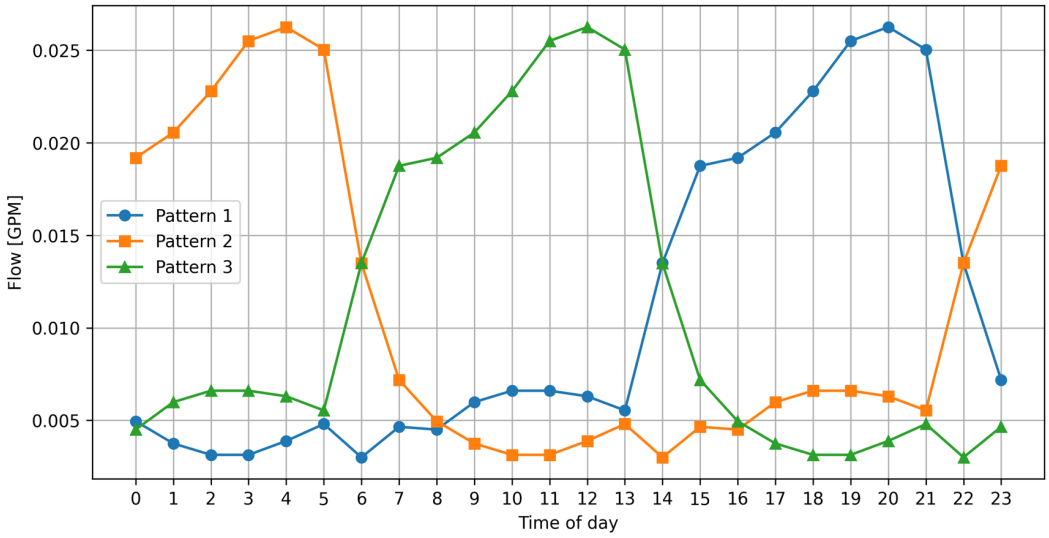

Figure 4.5. Daily water consumption patterns used for the evaluation of the proposed DQN model.

In particular, the three patterns present their maximum consumption at different times of the day: approximately at 04:00, 12:00, and 20:00. This simplification allows the neural network to focus on the effects of temporal demand distribution within the water network while maintaining comparable overall water usage across users. Finally, it provides a controlled scenario for evaluating the behavior of the proposed framework and the impact of different demand timing on the hydraulic simulation.

Chapter 5

Results

The evaluation of the proposed approach is carried out using the Water Distribution Network shown in Figure 3.2. This network represents the physical infrastructure on which the simulations are performed and includes the set of nodes, pipes, and reservoir that define the hydraulic composition of the system. All experiments are conducted on this network in order to assess how the proposed method performs in a realistic distribution scenario.

In addition to the network structure, another important component of the evaluation setup is the tariff profile used during the simulations. Specifically, two different tariff configurations are considered in the experiments. It is important

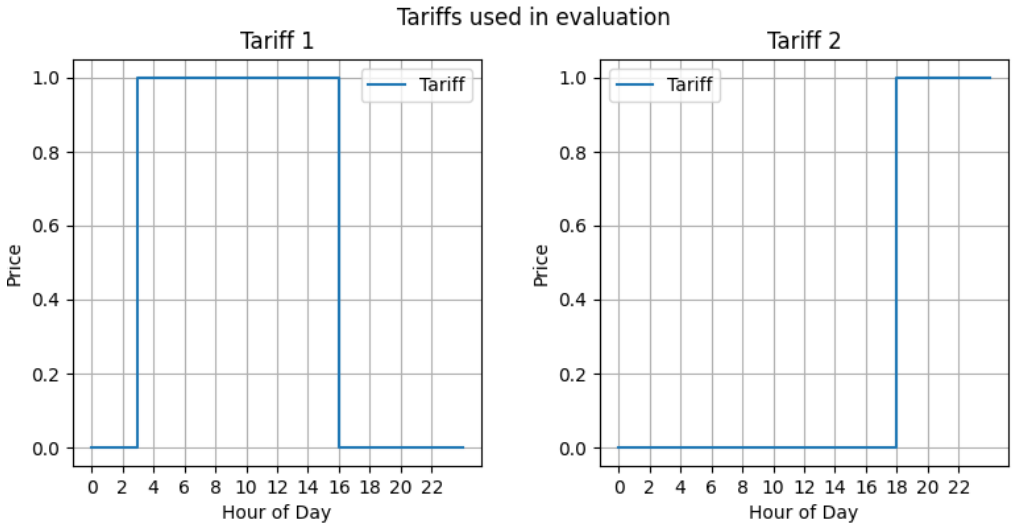

Figure 5.1. Time varying tariffs used to evaluate the DQN agent.

to note that these two tariffs are not simply translations of each other. The first tariff has a longer high price period (12 hours compared to 6 hours in the second), which will influence the cost function, however, this behavior is expected and for this reason it is normal to see higher cost function values for tariff 1.

5.1 Baselines

In optimization problems such as the one addressed in this work, it is common practice to evaluate the performance of a proposed method by comparing it with

baseline solutions. A baseline represents a reference strategy whose performance is already known or easily computable. These reference approaches are not necessarily optimal, but they provide a meaningful benchmark that allows researchers to assess whether a new method actually improves system performance.

In this context, the baseline solutions represent predefined operational strategies for the water distribution network. By measuring their performance, we obtain reference values that characterize how the system behaves under conventional or non learning based control policies. These values serve as a comparison point for evaluating the effectiveness of the proposed Deep Reinforcement Learning approach.

The objective of the DQN method is therefore to surpass the performance of these baselines. Performance is evaluated using the cost function previously defined, which captures the overall system objective by accounting for factors such as operational efficiency and service quality. Since the goal of this problem is to operate the network in the most efficient way possible, the objective is to minimize this cost function, which was defined directly to create a measure of this.

In many practical applications, especially when dealing with real world systems, defining meaningful baseline strategies is not always straightforward. Unlike well studied benchmark problems where standard reference solutions already exist, real operational environments can more easily lack of accepted control policies that can serve as clear comparison metrics. For this reason, in this case I adopt a simple but informative baseline design. Specifically, two types of baseline strategies based on static demand assignments are considered. The first baseline consists of uniform assignments, where all nodes in the network follow the same demand pattern. Although this is not a realistic scenario, it represents the worst case scenario so it can be used to better understand the other measures. The second baseline consists of random assignments, where demand patterns are assigned randomly to the nodes. This other strategy provides a reference corresponding to non optimized allocations, indeed as shown in figure 5.2, it is equal to the performance that the DQN model had in the initial part of the training.

The following table contains the values of the cost function calculated as references:

Table 5.1. Baseline Values for Different Tariff Schemes

Assignment Type	**Tariff 1**	**Tariff 2**
Random Assignment	**1007**	**861**
Fixed 0	1957	2021
Fixed 1	1999	1881
Fixed 2	2106	1763

As anticipated in the previous section, the values are almost always higher for tariff 1 because it maintains a high price for more time.

When analyzing the baseline performance, we observe a clear difference between the two strategies. The random assignment baseline produces cost function values that are within an acceptable range and can be reliably used as a reference for comparison. This happens because by randomly distributing demand patterns among nodes the system tends to avoid extreme simultaneous peaks, allowing the network to satisfy a larger portion of the demand without incurring in excessively high cost function values.

On the other hand, the uniform assignment baseline results in significantly higher cost values. The reason for this lies in the physical limitations of the WDN: the

network is unable to satisfy all nodes requesting water at the same time. Since the cost function is composed of two components (one representing the paid price and the other proportional to the amount of unsatisfied water) and, in this formulation, the one associated with unsatisfied water has a higher weight, when many nodes request water simultaneously under the same pattern, the network cannot fully meet the demand, and the cost function values increase a lot by the high penalty for unsatisfied water. This explains why the uniform baseline produces much higher costs compared to the random baseline. This behavior is expected as in this specific application case there's not only the need of reducing costs but the most important part is to balance water flows. These results highlight the importance of defining an appropriate cost function that is adapted to the specific case.

5.2 Training Results

The experiments were performed on a computational environment with an Intel Xeon w7-3455 processor (2.5 GHz, 24 cores) and 187 GB of RAM. A key factor affecting the overall execution time of the experiments is the duration of each training episode, which is largely determined by the hydraulic simulation performed through WNTR. Since each candidate solution must be evaluated through a full network simulation, the computational cost of this step dominates the training process. In the used environment, each episode requires on average approximately 39 seconds to complete.

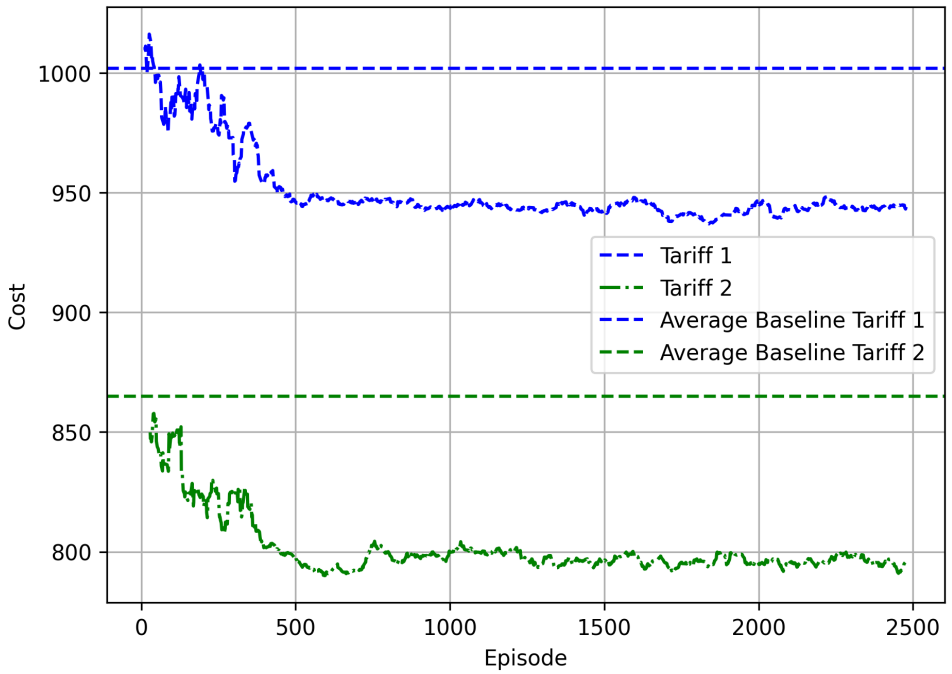

Figure 5.2. Evolution of the cost function over training episodes and for the random baselines.

The training process is illustrated in Figure 5.2, which shows the evolution of the cost function across training episodes for the two considered tariff scenarios along with random baselines. At the beginning of the training phase, the Reinforcement Learning agent explores the solution space starting from essentially random assignments of

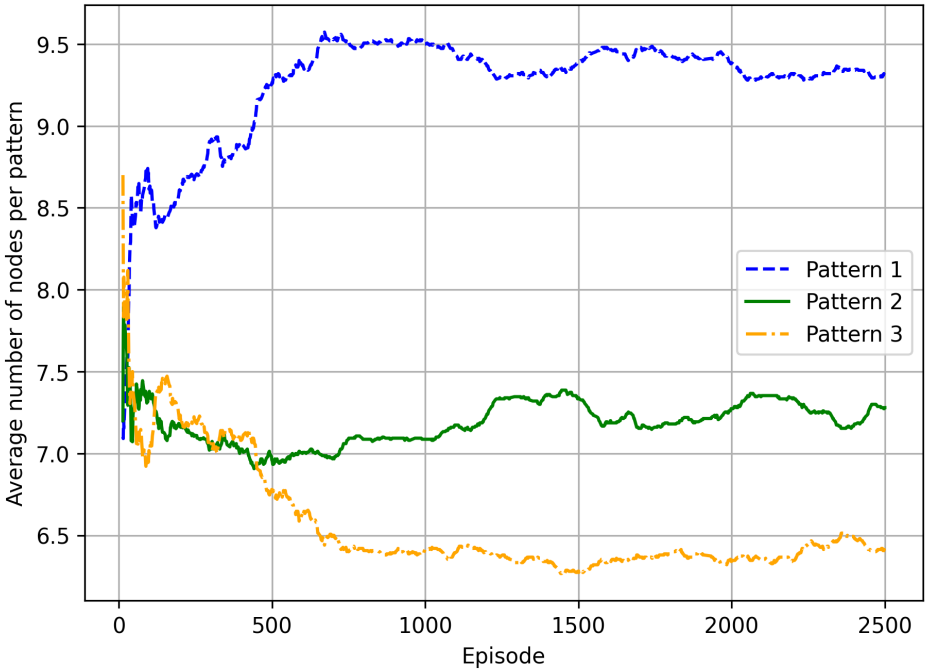

Figure 5.3. For a specific tariff, the evolution of the average number of nodes assigned to each consumption pattern across training episodes. Highlighting how the optimal pattern is preferred.

consumption patterns to the nodes. As expected, this results in relatively high initial cost values. As the number of episodes increases, the agent progressively improves its decisions by learning which pattern assignments lead to lower costs. The cost function shows a good decreasing trend until the learning process stabilizes after approximately 500 episodes, indicating that the agent has converged to a solution. Eventually, the algorithm achieves minimum cost values slightly below 800 for tariff 2 and just below 950 for tariff 1.

More information on the learned policy is provided in Figure 5.3, which shows for one of the tariff scenarios, the evolution of the average number of nodes assigned to each consumption pattern during training. As learning progresses, the distribution of nodes tends to stabilize and shows a preference for one specific pattern. This behavior is consistent with the characteristics of the tariff structure, as the preferred pattern corresponds to the one that concentrates a larger portion of water consumption during the hours with lower prices. However it is important to also note how some node is still assigned to the other patterns in order to balance flows and avoid unsatisfied water requests.

Moreover, figures 5.4 and 5.5 show how satisfied water increases and the paid price decreases as the policy is learned. This final tradeoff that the agent is able to reach is the result of how the two contributions in the cost function are weighted, this is important because this means that even with different needs, this approach can adapt by minimizing specific metrics.

As last, the proposed approach is able to identify which specific nodes should follow each pattern. This selective assignment enables the RL agent to exploit the structural characteristics of the network, leading to improved system performance. Indeed, after the training process, each node will have its own embedding representing user specific features used by the neural network in order to assign the pattern that

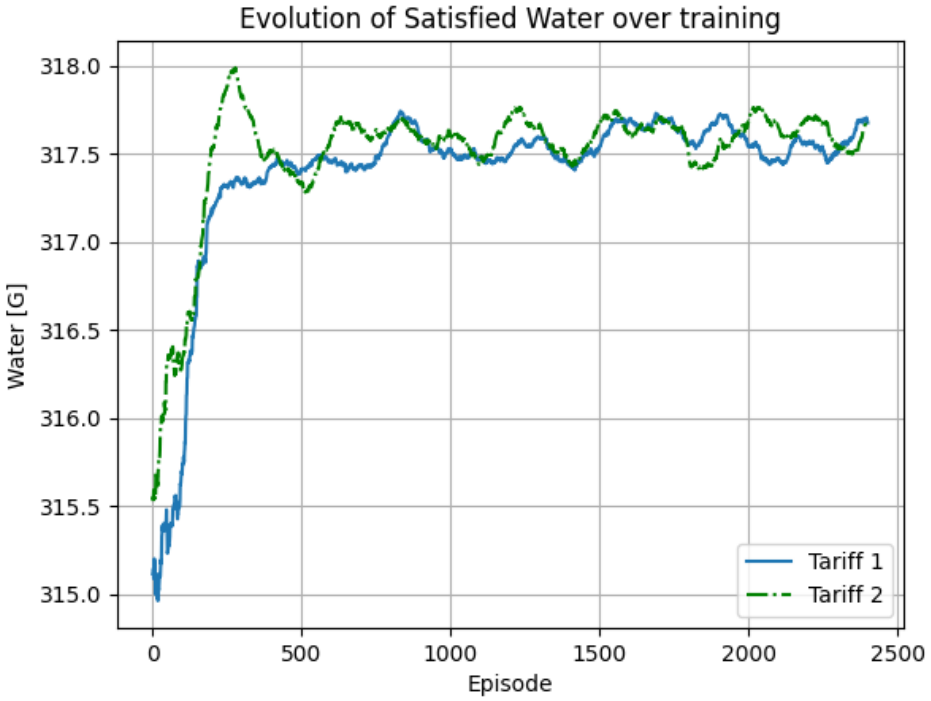

Figure 5.4. For a specific tariff, the evolution of satisfied water among all nodes.

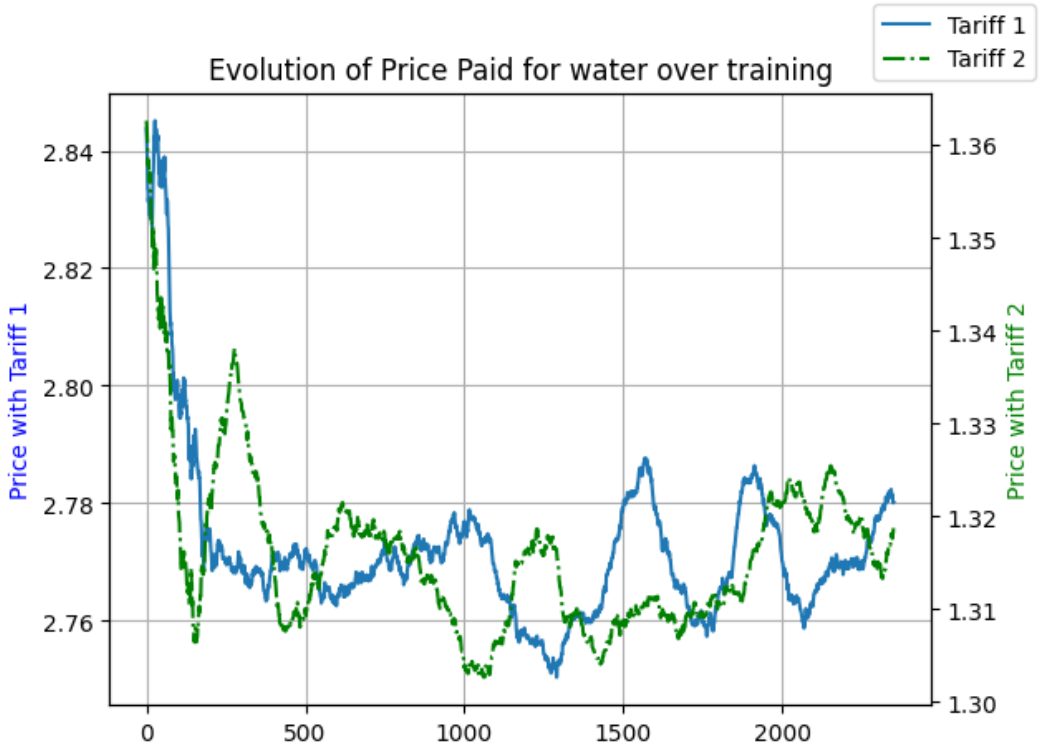

Figure 5.5. For a specific tariff the evolution of price paid by all nodes.

best harnesses the structural network capabilities.

Table 5.2. Baseline and DQN Agent Cost Values.

Assignment Type	Tariff 1	Tariff 2
Baseline	1007	861
Agent	**950**	**800**

The results also show that the proposed approach successfully outperforms the baseline strategies introduced earlier. In particular, the cost values achieved by the Reinforcement Learning agent are consistently lower than those obtained with both the random and the uniform assignments, demonstrating the effectiveness of the learned policy. Table 5.2 contains the results of the DQN agent along with the best baseline values.

This confirms that the RL based method is capable of identifying efficient consumption pattern allocations across the network nodes, leading to a reduction of the overall cost function compared to the reference baseline configurations.

Chapter 6

Conclusions

6.1 Overview

In this thesis I addressed the problem of efficient water allocation in Water Distribution Networks under conditions of uncertain and dynamic demand. Ensuring fair distribution of water is a critical challenge, especially when available resources are limited due to various reasons, in general environmental. Different allocation strategies that rely on predefined rules or static optimization approaches may struggle to adapt to changing conditions and complex network dynamics.

To tackle this problem, the water distribution process was formulated as a Markov Decision Process, in order to enable the use of reinforcement learning techniques to learn adaptive allocation policies. In particular, a Deep Q-Network approach was adopted to approximate the optimal action-value function and determine efficient allocation strategies over time. The learning environment was implemented using WNTR as simulation framework, which allows realistic modeling of hydraulic behavior in water networks. The agent interacts with the simulated environment by observing the system state ad applied tariff, selecting allocation actions, and receiving rewards based on the level of price paid and demand satisfaction achieved in the network.

All this goes together with a blockchain-based mechanism made to support transparency and accountability in the distribution process. Smart contracts were designed to implement automated compensation mechanisms based on met water demand, while IoT sensors were assumed to provide real time measurements of water usage and network conditions.

6.2 Results and limitations

The evaluation demonstrated that the proposed reinforcement learning approach is capable of learning effective allocation strategies within the simulated environment. Through the training process, the Deep Q-Network progressively improved its policy, gradually leading to higher levels of demand satisfaction eventually reaching a balanced assignment across the network that surpasses the baseline strategies considered.

From a methodological point of view, this work contributes by formulating the water allocation problem in Water Distribution Networks as a Markov Decision Process and by applying a Deep Q-Network to learn allocation policies directly from interaction with a hydraulic simulation environment. The integration of the WNTR simulator made the evaluation of the learning algorithm in a realistic hydraulic

setting possible, allowing the agent to account for the physical behavior of a realistic network. On the other hand, the integration of blockchain technologies within the water distribution framework was enhanced starting from previous works. This contribution focused mostly on dynamic smart contracts application to define policies enforced by suppliers.

A possible limitation of the proposed approach concerns its scalability when applied to large scale WDNs. In the implemented formulation, the state representation provided to the neural network includes the pressures of all nodes in the network. While this choice allows the agent to capture detailed information about the hydraulic state of the system, it also causes the size of the input space to grow proportionally with the size of the network. As the number of nodes increases, the input vector becomes significantly larger, which in turn increases the complexity of the neural network (which may also require additional layers) and the amount of training data required to learn an effective policy. This can lead to higher computational costs and slower training, potentially making the approach impractical for large WDNs with hundreds or thousands of nodes. For this reasons, although the proposed method performs well in the considered experimental setting, additional strategies would likely be required to ensure scalability when dealing with larger and more complex water distribution systems.

6.3 Future Directions

As a possible direction for future works, the scalability limitations of the current approach could be addressed by adopting neural network architectures that better exploit the graph structure of water distribution networks. In particular Graph Neural Networks such as Graph Convolutional Networks could provide a more suitable representation for this type of problem.

Water distribution networks are naturally modeled as graphs, where nodes represent junctions and edges represent pipes. In this setting, node pressures could be used as features associated with each node, allowing the model to process local hydraulic information while also capturing the structural relationships between different parts of the network. After some graph convolution the information would be aggregated across neighboring nodes, enabling the model to interpret meaningful local representations of the state.

The resulting node embeddings could then be used as input for a decision layer responsible for selecting the most appropriate allocation pattern. This approach would indeed leverage much more the topology of the water network, allowing the model to scale more effectively to larger systems while preserving relevant structural information.

Bibliography

[1] F. Ansalone, I. Chatzigiannakis, V. Taormina, D. Garlisi, "*A Reinforcement Learning Approach to Demand Balancing and Tariff Optimization in Blockchain-Based Smart Water Networks*", IFIP, 2025.

[2] A. Campisano and C. Modica, "*Optimal sizing of storage tanks for domestic rainwater harvesting in sicily*", Resources, Conservation and Recycling, vol. 63, pp. 9–16, 2012.

[3] "*Potential for peak flow reduction by rainwater harvesting tanks*", Procedia Engineering, vol. 89, 2014, 16th Water Distribution System Analysis Conference, WDSA2014.

[4] Y. Chang, G. Choi, J. Kim, and S. Byeon, "*Energy cost optimization for water distribution networks using demand pattern and storage facilities*", Sustainability, vol. 10, 2018.

[5] https://finance-commerce.com/2024/07/record-drought-tests-sicilys-water-infrastructure-and-tourism-industry/

[6] https://www.euronews.com/2024/08/07/sicilys-population-fed-up-of-water-shortages-as-rationing-starts-to-bite

[7] https://www.theguardian.com/environment/article/2024/aug/19/the-land-is-becoming-desert-drought-pushes-sicilys-farming-heritage-to-the-brink

[8] D. Amaxilatis, I. Chatzigiannakis, C. Tselios, N. Tsironis, N. Niakas, and S. Papadogeorgos, "*A smart water metering deployment based on the fog computing paradigm*", Applied Sciences, vol. 10, 2020.

[9] M. Zecchini, A. A. Griesi, I. Chatzigiannakis, D. Amaxilatis, and O. Akrivopoulos, "*Identifying water consumption patterns in education buildings before, during and after covid-19 lockdown periods*", in 2021 IEEE International Conference on Smart Computing (SMARTCOMP).

[10] M. Zecchini, A. A. Griesi, I. Chatzigiannakis, I. Mavrommati, D. Amaxilatis, and O. Akrivopoulos, "*Using iot data-driven analysis of water consumption to support design for sustainable behaviour during the covid-19 pandemic*", in 2021 6th SEEDA-CECNSM. IEEE, 2021.

[11] M. A. Wister, E. Leon, A. Alejandro-Carrillo, P. Pancardo, and J. A. Hernandez-Nolasco, "*Using iot for cistern and water tank level monitoring*", Applied System Innovation, no. 6, 2024. [Online]. Available: https://www.mdpi.com/2571-5577/7/6/112

[12] A. Pagano, D. Garlisi, F. Giuliano, T. Cattai, R. J. L. Taloma, F. Cuomo, "*Introducing and evaluating SWI-FEED: A smart water IoT framework designed for large-scale contexts*", Computer Communications, Volume 237, 2025, 108146, ISSN 0140-3664.

[13] G. Restuccia, D. Garlisi, F. Giuliano, A. Pagano, I. Tinnirello, "*The Synergy of Digital Twins and Machine Learning in Sustainable Water Management: SWIM*", Computer, Volume 58, no. 12, pp. 46-54, Dec. 2025, doi: 10.1109/MC.2025.361039.

[14] T. Sarantakos, D. Amaxilatis, A. Pagano, D. Garlisi, R. Laceda Taloma, T. Cattai, I. Chatzigiannakis, V. Vythoulka and C. Zaroliagis, "*A tool to facilitate the design of smart contracts in smart water distribution networks*", in IFIP, 2024.

[15] A. Bracciali, I. Chatzigiannakis, A. Vitaletti, and M. Zecchini, "*Citizens vote to act: Smart contracts for the management of water resources in smart cities*", in 2019 First International Conference on Societal Automation (SA). IEEE, 2019.

[16] M. Zecchini, A. Bracciali, I. Chatzigiannakis, and A. Vitaletti, "*On refining design patterns for smart contracts*", in European Conference on Parallel Processing. Springer, 2019.

[17] R. J. L. Taloma, F. Cuomo, D. Comminiello, and P. Pisani, "*Machine learning for smart water distribution systems: exploring applications, challenges and future perspectives*", Artificial Intelligence Review, vol. 58, no. 4, p. 120, Jan 2025.

[18] Puterman, M. L. (1994). "*Markov Decision Processes: Discrete Stochastic Dynamic Programming.*", John Wiley & Sons

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., "*Human-level control through deep reinforcement learning*" nature, vol. 518, 2015.

[20] Hado van Hasselt, Arthur Guez, David Silver "*Deep Reinforcement Learning with Double Q-learning*", 2015.

[21] Ziyu Wang, Nando de Freitas, Marc Lanctot "*Dueling Network Architectures for Deep Reinforcement Learning*", 2015.

[22] Tom Schaul, John Quan, Ioannis Antonoglou, David Silver "*Prioritized Experience Replay*", 2016.

[23] G. Hajgat´ o, G. Pa´ al, and B. Gyires-T´ oth, "*Deep reinforcement learning for real-time optimization of pumps in water distribution systems*", Journal of Water Resources Planning and Management, vol. 146, no. 11, p. 04020079, 2020.

[24] S. Hu, J. Gao, D. Zhong, R. Wu, and L. Liu, "*Real-time scheduling of pumps in water distribution systems based on exploration-enhanced deep reinforcement learning*", Systems, vol. 11, no. 2, p. 56, 2023.

[25] J. Xu, H. Wang, J. Rao, and J. Wang, "*Zone scheduling optimization of pumps in water distribution networks with deep reinforcement learning and knowledge-assisted learning*", Soft Computing, vol. 25, pp. 14 757–14 767, 2021.

[26] Rossman, L., H. Woo, M. Tryby, F. Shang, R. Janke, AND T. Haxton. "*EPANET 2.2 User Manual*". U.S. Environmental Protection Agency, Washington, DC, EPA/600/R-20/133, 2020.

[27] K. A. Klise et al., "*A software framework for assessing the resilience of drinking water systems to disasters with an example earthquake case study*", Environmental Modelling & Software, vol. 95, 2017.

[28] Klise, K.A., Hart, D.B., Bynum, M., Hogge, J., Haxton, T., Murray, R., Burkhardt, J. (2020). "*Water Network Tool for Resilience (WNTR) User Manual: Version 0.2.3*". U.S. EPA Office of Research and Development, Washington, DC, EPA/600/R-20/185, 82p.

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