computers

Article

Optimizing Water Distribution through Explainable AI and Rule-Based Control

Enrico Ferrari [1,*,†], Damiano Verda [1,†], Nicolò Pinna [1,†] and Marco Muselli [2]

[1] Rulex Innovation Labs, Rulex Inc., 16122 Genoa, Italy; damiano.verda@rulex.ai (D.V.); nicolo.pinna@rulex.ai (N.P.)
[2] Institute of Electronics, Computer and Telecommunication Engineering, Italian National Research Council, 16149 Genova, Italy; marco.muselli@ieiit.cnr.it
* Correspondence: enrico.ferrari@rulex.ai
† These authors contributed equally to this work.

Abstract: Optimizing water distribution both from an energy-saving perspective and from a quality of service perspective is a challenging task since it involves a complex system with many nodes, many hidden variables and many operational constraints. For this reason, water distribution systems need to handle a delicate trade-off between the effectiveness and computational time of the solution. In this paper, we propose a new computationally efficient method, named rule-based control, to optimize water distribution networks without the need for a rigorous formulation of the optimization problem. As a matter of fact, since it is based on a machine learning approach, the proposed method employs only a set of historical data, where the configuration can be labeled according to a quality criterion. Since it is a data-driven approach, it could be applied to any complex network where historical labeled data are available. In particular, rule-based control exploits a rule-based classification method that allows us to retrieve the rules leading to good or bad performances of the system, even without any information about its physical laws. The evaluation of the results on some simulated scenarios shows that the proposed approach is able to reduce energy consumption while ensuring a good quality of the service. The proposed approach is currently used in the water distribution system of the Milan (Italy) water main.

Keywords: rule-based control; water distribution network; classification; optimization

Citation: Ferrari, E.; Verda, D.; Pinna, N.; Muselli, M. Optimizing Water Distribution through Explainable AI and Rule-Based Control. *Computers* **2023**, *12*, 123. https://doi.org/10.3390/computers12060123

Academic Editor: Stefan Bosse

Received: 10 May 2023
Revised: 5 June 2023
Accepted: 12 June 2023
Published: 18 June 2023

1. Introduction

Machine learning is a field of artificial intelligence, which deals with systems and algorithms able to automatically detect patterns in data. These systems and algorithms try to replicate the functioning of the human brain, which is able to learn general laws about a given system starting from a limited number of related examples. Our contribution consists of the proposal of a rule-based machine learning method designed to optimize a water distribution network (WDN). More specifically, the proposed method allows us to optimize the control of the pumps in real time. Even though we will focus on this application, we would like to mention that, in principle, the proposed algorithm could be applied in any other scenario involving a real-time optimization problem. The development of a support decision tool for this application is motivated by the fact that it is very difficult for an operator to predict the effect of status changes on a set of pumps across the whole system over time. In fact, both the overall energy consumption and the system quality of service must be taken into account at the same time for the optimization. We may define quality of service according to measurements related to water quality (such as chlorine concentration on water age) but, in the following, we will focus on a definition of quality of service based on the compliance of pressure levels in the network with target pressure ranges: the relevance of this element in water main operations is discussed, for example, in [1]. Operations are usually based on simple correlations known a priori. For example, if

the aim is to improve the quality of service (QoS) in a specific point of the network, the operator may switch on a pump in a nearby pumping station that is known to be related to the pressure in that point.

It is, however, not always so easy to take other correlations into account empirically; for example, correlations related to interactions among pumping stations, or to effects associated with more distant pumping stations.

For this reason, a machine learning approach could help in making better predictions by also considering more complex correlations among variables. Most machine learning approaches provide models in the form of complex mathematical functions, which leads to a number of significant limitations: (a) it is difficult to understand the cause–effect relationships and therefore (b) it is difficult to understand which drivers should be changed to improve network behavior. For this reason, a rule-based approach is proposed, leading to a better understanding of the network behavior and its key drivers. In this paper, rule-based control (RBC) is proposed to optimize a WDN. The two steps constituting the proposed method are the following:

- The extraction of a rule-based model that allows us to predict the future status of the network (e.g., pressure constraints will not be fulfilled/energy consumption will be excessive/a good status of the network);
- The use of the extracted model to generate controls to change the status of the pumps in order to match rules that predicted the desired status of the network.

The proposed pipeline is currently used by one of the biggest water main systems in Italy (Milan water main). Its performance in the long term, in terms of energy saving, is under evaluation and is being compared with other state-of-the-art techniques.

The structure of the paper is the following: Section 2 focuses on related work in this field, Section 3.1 is dedicated to the description of the proposed modeling technique and Section 3.2 shows how the use of this modeling technique allows us to develop and apply the innovative rule-based control algorithm. Section 4 includes experimental results; conclusions follow.

2. Materials and Methods

In a water supply system, the source of water is generally a lake, a river or an underground aquifer. Water then flows through water treatment facilities so that it can be purified and is finally delivered to all the demand points. This defines, especially if the area to be covered by the water main is large, a complex network, constituted by many interconnected elements and sub-systems with different functions.

The quality of service may be determined by different criteria and parameters; for example, the quality of the water can be assessed. Another relevant objective of the water main system, which will be the target of our proposed technique in experiments, is to ensure that the water reaches every area of the network under a satisfactory pressure. In order to assess this and to optimize it, pressure sensors are distributed in the network and an optimal pressure range is defined for each of the monitored points.

When optimizing this, it must be taken into account that hydraulic pumps, which allow for the reaching of the goal, require energy to work, and the cost to power them represents a significant percentage of the overall cost of the system. Due to that, the ideal goal, widely explored in this field of research, is to ensure that the optimal ranges for pressure are met with the least possible energy consumption.

As observed in [2], there is no single optimal choice of architecture that could be recommended to fit any water distribution control due to the existence of site-specific challenges. As previously mentioned, sensors shall be distributed in the network to allow for the acquisition of field measurements, while actuators allow us to actively control the distribution process according to the measurements and the prediction of the model about the future status of the water main system. The control of the network can be either supervised or entirely automatic.

Supervised control implies the intervention of a human operator who needs to confirm or modify the controls suggested by the model. Conversely, in an entirely automatic scenario, control actions are applied without any human intermediation. Considering that ensuring the correct behavior of a water main system is a very critical task, the entirely automatic solution is hard to pursue, at least as first step. Another distinction worth mentioning concerns the level at which the optimization of a water distribution network can happen. For example, optimizing can be interpreted as choosing the best solution for planning a new distribution system or for expanding an existing one: an extensive review of the most popular approaches for addressing the problem in this way was proposed in [3]. To cite a few examples, [4] used a meta-heuristic algorithm named *Shuffled Frog Leaping* to determine optimal discrete pipe sizes for new pipe networks or for network expansions. In [5], a local search was used to find the least expensive pipe configuration that satisfies hydraulic laws and customer requirements in a quick and transparent way.

The goal of the *WDN operation optimization* is to pick the best solution for pump and/or valve control in an existing network. The WDN operation optimization generates a schedule of pump operations either for a target time interval (e.g., a day or a week), or continuously, checking and possibly updating controls for the current steps and possibly for the future steps. In case the scheduling is fixed and does not change, we can also refer to the problem as a *Pump Scheduling Optimization* problem.

Since the solution is static, i.e., it does not need to be updated on a regular basis, computationally expensive methods can also be used to solve this kind of problem. Therefore, once the problem has been formulated, it is possible to find an optimal or near-optimal solution, which can then be used for scheduling the WDN. Scheduling optimization does, however, require that some boundary conditions, such as water demand, are fixed and known at the beginning of the scheduling period (e.g., a day) while, in a real-word scenario, the demand presents fluctuations that are difficult to predict ahead of time. Differences between estimated demand and the actual one can lead to bad optimization and, potentially, to failing to satisfy constraints.

Conversely, if the optimal solution is verified and, if needed, updated at each step, taking the input gathered at the current step into account, the problem is defined as *Real Time Pump Optimization*. The solution, in this case, happens at each time step: considering that we are speaking of a real-time application, computational efficiency in the determination of a (near) optimal solution is crucial.

In both cases, the optimization process aims to minimize the quantity/cost of energy consumed by pumps while ensuring that the network can meet quality of service constraints defined by the customer. In some studies, other objectives, such as the water quality and greenhouse gas emissions (GHGs), are also considered.

A relevant example of a controlled variable is WDN pressure. In this context, controllers were proposed starting from simple physical considerations on network behavior; for example, in [6]. As an alternative, forecast techniques estimating the future state of the system can also be the input for the control layer. This kind of approach, for instance, was proposed in [7,8].

In [9], a survey of methods for operation-level control of WDN pressure was reported. The methods proposed in the literature include mixed-integer linear programming as suggested by [10] as well as a combination of optimization methods with machine learning approaches as in [11–13].

In real-world applications, the most used metaheuristic search methods are *genetic algorithms* (*GAs*), which are inspired by Darwin's theory on the biological processes of reproduction and natural selection. Since a GA simulates a biological process, it includes some randomness and different runs could lead to non-identical results; consequently, there is no guarantee that the GA reaches the optimal solution and it may stop at a local minimum. Nevertheless, some experiments have shown that, if the algorithm parameters are chosen correctly, the use of GAs could lead to near-optimal solutions, as shown in [14].

Basically, all the variants of GAs are based on both the iterative generation of possible solution vectors and their evaluation based on a specific fitness score.

The method proposed in this paper allows for the real-time optimization of a network without imposing a physical model of the network. According to the four criteria defined by [15], this is how it relates with state-of-the-art techniques:

- Application area: The application area of RBC is "pump operation", like the largest portion of the paper analyzed by Mala-Jetmarova (40%).
- Optimization model: The optimization model used in this work considers a single objective (pump energy consumption) like the vast majority of models according to Mala-Jetmatova (84%). The test case considered 3 constraints (demand satisfied, nodes pressure, tanks level) but RBC allows us to deal with any number of constraints by changing the quality indicator.
- Solution methodology: The RBC solution methodology is hybrid: the rule extraction step is stochastic, whereas the control step is deterministic. (in the literature, deterministic methods are 45.5% of the total).
- Test network: For what concerns the test network, we used a mid-size network with approximately 400 demand nodes, whereas 80% of papers used a network with fewer than 100 nodes.

2.1. Optimization Model

Optimization consists of selecting the best values for some variables (called *decision variables*) to minimize or maximize a (set of) functions (called *objectives*) according to specific *constraints*. If an optimization problem includes only one objective, it is called *single-objective programming*; otherwise, it is called *multi-objective programming*.

In multi-objective optimization, it is usually impossible to find a single solution that minimizes (maximizes) all the objectives, so the result is consequently a set of *nondominated* solutions where ulterior improvements in one objective are not possible without worsening the others. Decision makers must then select the best option from the set of solutions.

For this reason, most of the literature on WDN operation optimization focuses on single-objective programming. If more criteria need to be optimized by design, they are usually combined into a single-objective function. Usually, the decision variables are the statuses of the pumps, which could be either binary (on/off) or continuous variables, if the pump includes some mechanism to tune the pressure of outgoing water.

Almost all the models in literature include the pump operating cost in the objectives, while other frequently used costs are the maintenance, usually estimated from the number of pump switches (e.g., [16]), the water age (e.g., [17]) and the GHG emissions (e.g., [18]).

The problem is called *linear programming* if both the objectives and the constraints can be written as linear combinations of the decision variables; otherwise, it is called a *non-linear programming problem*. The WDN optimization problem in the literature is formulated both as a linear problem and as a non-linear problem.

A large variety of WDN operational constraints is present in the literature, and the most common ones concern pressure at nodes and levels of tanks. For instance, a linear regression model can be used to interpolate the relationship between pump status, tank levels, demand, pressure and energy in order to obtain an optimization model characterized by a linear objective and linear constraints (e.g., [19]). Of course, introducing a linear model could induce an excessive simplification of the behavior of the system and therefore a reduction in the performance of the control.

In the following subsections, a short description of the main optimization models and solution methodologies is proposed.

2.2. Optimization Methods

An optimization method is a technique used to find a solution for an optimization model. The most common approach in this field is referred to as *metaheuristic optimization*.

A metaheuristic algorithm is a general-purpose algorithm that can be used to find a (sub)optimal solution for both linear and non-linear problems. Even if metaheuristic approaches do not guarantee that the optimal solution is retrieved, they are usually faster than deterministic methods and, therefore, they can also be used when the optimal solution cannot be obtained in a reasonable amount of time.

Notice that metaheuristic algorithms also aim at exploring the solution space by including some random decisions; thus, different executions could lead to different solutions even if they start from the same problem and set of parameters.

Exploring the solution space through metaheuristic approaches is usually combined with a hydraulic simulator that, at each iteration of the algorithm, tests the quality of the explored solution. The most commonly used heuristic search methods are *genetic algorithms* (*GAs*), which are described in Section 2.3.

For real-time optimization, the use of hydraulic simulators is computationally expensive as shown in [12]; consequently, a machine learning method can be used to capture knowledge from the hydraulic simulator.

For example, in [13], a GA was combined with a *multilayer perceptron* (*MLP*), introduced in [20]. In this approach, the MLP retrieves the status of each pump, the level of the tank and the expected water demand at time t as inputs and returns the energy consumption, the level of each tank and the pressure of each node at time $t + \Delta t$, where Δt is the time period to be optimized. The GA then uses this information in place of the hydraulic simulator to find the best configuration according to the consumed energy and quality of the service.

The MLP is trained by means of a training set obtained by collecting a large set of possible behaviors from the hydraulic simulations of different realistic network situations. The software used for simulations is EPANET 2.0, introduced by [21], one of the most widely used hydraulic simulators, which will also be referred to in the experimental section of the present paper.

However, MLPs are black boxes, i.e., classifiers whose behavior is difficult to explain. Recent trends in AI tends to prefer white boxes, i.e., classifiers whose behavior can be understood by a human: this approach is usually called explainable AI (XAI).

2.3. Genetic Algorithms

A genetic algorithm (GA) is a metaheuristic research method inspired by Darwin's theory on the biological processes of reproduction and natural selection.

The method allows us to find high-quality solutions for complex problems, where traditional methods are not applicable or require excessive computational effort. Since a GA simulates a biological process, it includes some randomness, and different runs could lead to non-identical results; consequently, there is no guarantee that GA reaches the optimal solution and it may stop at a local minimum. Nevertheless, some experiments have shown that, if the algorithm parameters are chosen correctly, the use of GAs could lead to near-optimal solutions, as shown in [14].

Different variants of GAs are used in the literature. Basically all of them are based on the iterative generation of possible solutions by means of the combination of elementary operations performed on the values of potential solution vectors. Borrowing the terminology from biology, the value of a variable is a *gene*, which may undergo a *mutation* (i.e., a random change) or a *recombination* with another solution, giving origin to a new *offspring*. Moreover, at each iteration, only some solutions are selected to optimize a *fitness* score.

In most GA applications for solving WDN operation optimization problems, the objectives and the constraints are not computed by closed-form expressions depending on the status of pumps. Usually, objectives and constraints are either produced by a hydraulic simulator or by a model that emulates the behavior of a simulator.

3. Theory/Calculation

3.1. Modeling

As mentioned in Section 2, optimization methods require the management of several closed-form differential equations dealing with the physical laws that describe the behavior of the network or, alternatively, the use of hydraulic simulators that implicitly (and approximately) handle these relations.

In this paper, a different approach is proposed that allows us to perform optimization without the need for a physical model or a simulator. In fact, the relationships among parameters were implicitly derived from historical data by evaluating the effectiveness of network configurations used in the past in ensuring the quality of the WDS. Once the drivers leading to a good quality of the network are identified from historical data, it is possible to act on them to improve the current configuration quickly and with little computational cost. Due to this, the proposed approach does not need the definition since a set of solutions labeled as "good" or "bad" is sufficient to highlight the drivers that influence the quality of the network.

Thanks to these features, RBC could be an efficient alternative to exact optimization when it is not possible (or too computationally intensive) to predict the evolution of the network through closed differential equations or an accurate hydraulic simulator.

RBC aims at identifying and acting on a solution leading to "good configuration" and "bad configuration". The definition of a "good solution" depends, of course, on the requirements and on the boundary conditions of the network; it is, however, possible to define the current performance of the network based on historical statistics. Consider a set $\mathcal{S}_H$ of historical data:

$$\mathcal{S}_H = \{(\boldsymbol{x}_i, y_i)\}$$

where y_i is the observed cost (e.g., the energy consumption per quantity of pumped water) for the considered configuration. We will refer to a solution of the problem as a set of settings of the network constituting the parameter vector $\boldsymbol{x}_i$. In particular, the solution $\boldsymbol{x}_i$ is "good" with respect to historical data S_H if $y_i \leq Q_y(S_H, q)$, where $Q_y(S_H, q)$ is the q-th percentile (where $0 \leq q \leq 1$) of observed cost y_i in historical data S_H (the q-th percentile is the minimum observed value in data that is greater or equal to a fraction q of all the observed values). Otherwise, the solution is "bad". For example, by setting $q = 0.5$, a solution is considered good if its indicator is less than or equal to the median of the cost of past solutions, and is bad otherwise.

The goal of RBC is to produce solutions with a performance indicator smaller than $Q_y(S_H, q)$. These solutions will constitute new historical data for future runs, and future runs will aim to further decrease the performance indicator. The value of $Q_y(S_H, q)$ is then expected to decrease progressively; in other words, when iterated several times, RBC should therefore lead to better and better solutions over time until reaching a value $\hat{Q}_y(S_H, q)$, which cannot be further improved. Ideally, if y_i does not depend on external parameters and supposing that it is always possible to act on controllable variables, the optimization effect of RBC will decay over time as the steps toward an optimal solution become smaller and smaller. In a real-life situation, it may not be feasible to bring the output under the percentile acting on controllable variables.

Supposing that no other constraints are set for the WDN, the measure y computed on historical data, properly discretized, can be used as the output of a classification problem to make a prediction about the performance and its correlation with the network configuration.

However, the problem usually includes some operational constraints that define the quality of the system (i.e., pressure at nodes); therefore, a definition based only on energy efficiency is not suitable.

More specifically, in a typical WDN optimization problem, these indicators can be considered:

- The *efficiency indicator* y^{eff} that corresponds to the cost in the unconstrained case, i.e., it is related to the energy consumption.

- The *quality indicator* y^{qlt} that represents the observed value of a penalty function accounting for the fulfillment of constraints.

Since the quality indicator is the primary objective for reaching a good solution, the criterion for evaluating if a solution is efficient considers y^{qlt} as the primary objective and y^{eff} as the secondary objective.

$$y_i^{\text{qlt}} \leq Q_{y^{\text{qlt}}}(\mathcal{S}_H, q) \tag{1}$$

$$y_i^{\text{eff}} \leq Q_{y^{\text{eff}}}(\mathcal{S}_H^*, q) \tag{2}$$

where $\mathcal{S}_H^*$ is the subset of the $\mathcal{S}_H$ satisfying condition (1). In the first place, a solution must have a good quality indicator, and the best solutions will also be characterized by a good efficiency indicator.

A single discrete indicator can therefore be defined with three possible classes:

- *Bad quality of the service* (BadQoS) is the worst class and indicates that the constraints are poorly respected $y_i^{\text{st}} = \text{BadQoS}$ if $y_i^{\text{qlt}} > Q_{y^{\text{qlt}}}(\mathcal{S}_H, q)$;
- *Bad energy efficiency* (BadEff) is an intermediate class corresponding to fulfilled constraints but a high cost function $y_i^{\text{st}} = \text{BadEff}$ if $y_i^{\text{qlt}} \leq Q_{y^{\text{qlt}}}(\mathcal{S}_H, q)$ and $y_i^{\text{eff}} > Q_{y^{\text{eff}}}(\mathcal{S}_H^*, q)$;
- *Good energy efficiency* (GoodEff) is the best class in which constraints are respected and the efficiency indicator is low $y_i^{\text{st}} = \text{GoodEff}$ if $y_i^{\text{qlt}} \leq Q_{y^{\text{qlt}}}(\mathcal{S}_H, q)$ and $y_i^{\text{eff}} \leq Q_{y^{\text{eff}}}$.

The dataset produced by this *output definition* phase is made of N examples $S = \{(\boldsymbol{x}_i, y_i^{\text{st}})\}_{i=1}^N$ and can be used as a training set for the learning phase of a rule-based classification model. In this context, the use of the XAI method is crucial because it allows us to not only make predictions, but also to have a clear insight about the cause–effect relationships.

This learning phase consists of using historical data to generate the model that describes the relationship between inputs (parameters of the WDN) and the output (performance of the WDN).

For the rule extraction step, the use of the *logic learning machine* (*LLM*) method is suggested. Notice that it is possible to perform the rule generation phase periodically (e.g., every day) so that changes in the behavior of the system are included in the created models.

The LLM is a clear box method based on the *switching neural network* (*SNN*) model, introduced by [22], which generates a set of intelligible rules through Boolean function synthesis. Briefly, the LLM transforms the data into a Boolean domain where some Boolean functions (namely one for each output value) are reconstructed starting from a portion of their truth table with the method described in [23]. The Boolean functions are then converted back into intelligible rules involving the original variables:

$$\textbf{IF} < \textit{premise} > \textbf{THEN} < \textit{consequence} >$$

where $< \textit{premise} >$ is the conjunction of a set of conditions on the input variables (e.g., the pressure, the time of the day, etc.) and $< \textit{consequence} >$ contains the output value (i.e., the performance of the network) corresponding to $< \textit{premise} >$.

Competing techniques, such as decision trees, introduced by [24], usually produce complex rulesets (i.e., with many splits) that are too dependent on the initial splits. Moreover, the effectiveness of rule-based optimization depends on the "completeness" of the set of rules. In other words, as the next sections will clarify, optimization is searching for conditions leading to good configurations; thus, it is important that (almost) all the possible correlations are retrieved by the classification algorithms in order to increase the chance of having at least one available and attainable good configuration. On the contrary, due to the divide-and-conquer approach of DT, even if they are more complex, the number of rules is usually reduced and each vector $\boldsymbol{x}$ satisfies only one rule; for this reason, in several situations, it is possible that no good configurations can be achieved from the current

situation. A random forest (RF) model, proposed by [25], would produce a larger ruleset containing redundant rules, where it is also more complex for the RF to apply changes in order to improve a solution. In addition, since it is not constrained to a tree structure, the LLM is able to integrate intelligible rules based on WDN expert knowledge, increasing the level of customization of the solution. However, if a DT or RF are used, the RBC approach can be applied after having converted the tree (or the ensemble of trees) into a set of rules.

Whatever the rule generation method used, the rules can be evaluated, introducing some quality measures related to their ability to describe the available data.

A generic rule $r \in \mathcal{R}$ can be seen as a couple $(\boldsymbol{\kappa}(r), O(r))$, where $\boldsymbol{\kappa}(r)$ is a vector of conditions and $O(r)$ is the output in its consequence. Let $A(\kappa)$ be the attribute associated with a condition κ . An example $\boldsymbol{x}$ is said to be *covered* by r (denoted by $\boldsymbol{x} \leq r$) if it satisfies all the conditions in $\boldsymbol{\kappa}(r)$. Vice versa, $\boldsymbol{x} \not\leq r$ means that $\boldsymbol{x}$ is not covered by $\kappa(r)$. The *covering* $\chi(r)$ of r is the fraction of cases $(\boldsymbol{x}_i, y_i)$ in the training set having $y_i = O(r)$ and input $\boldsymbol{x}_i$ covered by r:

$$\chi(r) = \frac{N_{O(r),\boldsymbol{\kappa}(r)}}{N_{O(r)})}$$

where $N_{O(r),\boldsymbol{\kappa}(r)}$ is the number of cases in the training set having output $O(r)$ and satisfying all the conditions in $\boldsymbol{\kappa}(r)$ and $N_{O(r)}$ is the number of cases having output $O(r)$, while the *error* $\epsilon(r)$ of r is the fraction of the example in the training set having output $y_i \neq O(r)$ covered by r.

$$\epsilon(r) = \frac{N_{\overline{O(r)},\boldsymbol{\kappa}(r)}}{N_{\overline{O(r)}}}$$

where $N_{\overline{O(r)},\boldsymbol{\kappa}(r)}$ is the number of cases in the training set having an output different from $O(r)$ and covered by r and $N_{\overline{O(r)}}$ is the number of cases where the output is not $O(r)$.

Combining the covering and error, we can define the *relevance* of a rule as:

$$R(r) = \chi(r)(1 - \epsilon(r))$$

Since a new configuration $\boldsymbol{x}$ being described by rules of distinct classes may occur, the relevance of the rules can be used to determine the output associated with $\boldsymbol{x}$ by computing a score for each output class c:

$$S(\boldsymbol{x}, c) = \sum_{r \in \mathcal{R}_{\boldsymbol{x}}^{c}} R(r) \tag{3}$$

where $\mathcal{R}_{\boldsymbol{x}}^{c}$ is the subset of rules covering $\boldsymbol{x}$ with an output of $O(r) = c$. The input vector $\boldsymbol{x}$ will then be associated with the class c scoring the highest value of $S(\boldsymbol{x}, c)$.

3.2. Control

The control phase consists of continuously applying the rule-based model, taking the current solution as the input and providing a new solution that improves the quality of the network. Notice that the new solution becomes part of the historical data, thus improving the average performance of past configurations.

In general, the rationale behind RBC consists of searching for all the rules that $\boldsymbol{x}$ could match and then changing some values in $\boldsymbol{x}$ so that more good rules cover the modified version of $\boldsymbol{x}$. It is important that RBC is configured so that only changes in controllable variables are admitted. For example, if a variable representing the time of the day exists, that cannot be changed: it is not controllable.

The search for rules and solution edits is repeated until a termination condition is met and the produced solution becomes part of the historical data. Notice that the output definition and rule extraction steps are not performed at every iteration but only when the changes applied in the Rule Based Control step no longer lead to improvements or the classification model performance drops on average over a specific time period.

The description of the proposed method assumes that the classification model has been generated by means of the LLM algorithm.

Before applying RBC, a weight value should be associated with each output class so that the higher the weight, the higher the cost of the class. In particular, negative weights should be associated with target classes, whereas positive weights should be assigned to unwanted outputs. Moreover, weights should be chosen so that $w_{\text{BadQoS}} \geq w_{\text{BadEff}} \geq w_{\text{GoodEff}}$.

The goal of the optimization procedure is to find the vector $\boldsymbol{x}$ that corresponds to the minimum of the probabilities $P(\text{BadQoS} \mid \boldsymbol{x})$ and $P(\text{BadEff} \mid \boldsymbol{x})$ and to the maximum of the probability $P(\text{GoodEff} \mid \boldsymbol{x})$ for each WDN configuration $\boldsymbol{x}$. Since the probabilities $P(c \mid \boldsymbol{x})$ are usually unknown a priori, they can be estimated using the rules derived from the LLM. The generic probabilities $P(c \mid \boldsymbol{x})$ derived from an LLM model can be obtained as a function of the rules covering $\boldsymbol{x}$ relative to c and the rules covering $\boldsymbol{x}$ relative to classes other than c.

$$P(c \mid \boldsymbol{x}) = \frac{S(\boldsymbol{x}, c)}{\sum_{k \in C} S(\boldsymbol{x}, k)} \tag{4}$$

where $C = \{\text{BadQoS}, \text{BadEff}, \text{GoodEff}\}$ is the set of all possible output classes. Then, the cost function $g(\boldsymbol{x})$ that RBC must try to minimize is introduced:

$$g(\boldsymbol{x}) = \sum_{c \in C} w_c P(c \mid \boldsymbol{x}) = \frac{\sum_{c \in C} w_c \cdot S(\boldsymbol{x}, c)}{\sum_{k \in C} S(\boldsymbol{x}, k)} \tag{5}$$

which, if the score of the class $S(\boldsymbol{x}, c)$ is computed by using Equation (3), can be written as:

$$g(\boldsymbol{x}) = \frac{\sum_{c \in C} \left(w_c \cdot \sum_{r \in \mathcal{R}_{\boldsymbol{x}}^c} R(r) \right)}{\sum_{k \in C} \sum_{r \in \mathcal{R}_{\boldsymbol{x}}^k} R(r)} \tag{6}$$

Equation (6) highlights that $w_c < 0$ leads to a negative contribution to the cost function and therefore the corresponding class c is preferred if possible; otherwise, classes where $w_c > 0$ are discouraged since they increase the cost function g. A class with $w_c = 0$ does not contribute to the cost, so the associated score is not relevant. Each $P(c \mid \boldsymbol{x}_i)$ can be increased (or reduced) by making changes to $\boldsymbol{x}$ in order to add (or remove) rules to $\mathcal{R}_{\boldsymbol{x}}^c$. RBC aims to change $\boldsymbol{x}$ so that more rules for the good class and fewer rules for the bad classes are matched.

Given the set E of modifiable attributes, RBC (see Algorithm 1) builds the set $\mathcal{R}_{\boldsymbol{x},E}$ of rules that could potentially cover $\boldsymbol{x}$ after some changes in attributes belonging to E.

RBC scans $\mathcal{R}_{\boldsymbol{x},E}^+$ starting from the class with the lowest weight (i.e., the most desirable class if there is more than one class with $w_c < 0$), detecting candidate rules to cover the current configuration. The rule belonging to the best class or, in the case of a tie, that with the highest relevance, is selected. In other words, the selection is performed by finding the maximum of $\left(-w_{O(r)}, R(r)\right)$ according to lexicographic ordering. Starting from the selected rule, the algorithm makes the minimum changes in $\boldsymbol{x}$ (creating a modified vector $\boldsymbol{x}^M$) so that r covers $\boldsymbol{x}^M$. In principle, this change should improve the value of the cost function; nonetheless, due to the strong interdependency among rules, it is possible that the performed changes make other undesired rules to cover $\boldsymbol{x}^M$. For this reason, the value of $g(\boldsymbol{x}^M)$ is computed and, if it is smaller than $g(\boldsymbol{x})$, the changes are maintained; otherwise, they are dropped.

If there are only two possible output values (e.g., good and bad), then it is sufficient to choose any $w_{\mathsf{Good}} < 0$ and $w_{\mathsf{Bad}} > 0$ to achieve the expected result for any starting solution $\boldsymbol{x}_i$ with RBC, since, in all cases, good is the desirable class and bad is the undesirable class.

On the other hand, if the possible values of the output are more than two, then the choice of weights is not trivial.

Since three classes have been defined with an increasing level of performance, two different approaches could be applied aiming at (a) maximizing the probability of GoodEff or (b) minimizing the probability of BadQoS. In many situations, the two approaches

would lead to the same result, but when the probability (according to the ruleset) associated with GoodEff is null, approaches (a) and (b) may produce different solution vectors. In other words, if approach (a) is adopted, a weight vector with $w_{\text{GoodEff}} < 0$, $w_{\text{BadEff}} > 0$ and $w_{\text{BadQoS}} > 0$ can be chosen. Nevertheless, if GoodEff cannot be reached (i.e., if the associated probability is null), this choice is not able to avoid class BadQoS. In order to change the performance of the configuration from BadQoS to BadEff, the weight function should be defined so that $w_{\text{GoodEff}} < 0$, $w_{\text{BadEff}} < 0$ and $w_{\text{BadQoS}} > 0$. In this way, RBC tries to increase the probability of both class GoodEff and BadEff.

Algorithm 1: Function **RuleBasedControl** that implements the control strategy based on rules.

Data: $\boldsymbol{x}$, $\mathcal{R}_{\boldsymbol{x},E}$, $\boldsymbol{w}$, E
Result: $\boldsymbol{x}^M$
$\mathcal{R}^+_{\boldsymbol{x},E} = \{r \mid r \in \mathcal{R}_{\boldsymbol{x},E} \text{ and } w_{O(r)} < 0\}$;
while $\mathcal{R}^+_{\boldsymbol{x},E} \neq \emptyset$ **do**
  $r^* = \arg\max_{r \in \mathcal{R}_{\boldsymbol{x},E}} \left(-w_{O(r)}, R(r)\right)$;
  $\hat{\boldsymbol{\kappa}}(r^*) = \{\kappa \mid \kappa \in \boldsymbol{\kappa}(r^*) \text{ and } A(\kappa) \in E\}$;
  $\boldsymbol{x}^M = \boldsymbol{x}$;
  for $\kappa \in \hat{\boldsymbol{\kappa}}(r^*)$ **do**
    $j = A(\kappa)$;
    if $x_j \not\leq \kappa$ **then**
      $x^M_j = \arg\min_v \{\| x_j - v \| \mid v \leq \kappa\}$;
    end
  end
  if $g(\boldsymbol{x}^M) < g(\boldsymbol{x})$ **then**
    $\boldsymbol{x} = \boldsymbol{x}^M$;
  end
  $\mathcal{R}^+_{\boldsymbol{x}} = \mathcal{R}^+_{\boldsymbol{x}} \setminus \{r\}$
end

It is worth noting that, since it is possible that the optimum cannot be reached, Algorithm 1 is not the best approach. In this situation, deciding to avoid the worst configuration (i.e., BadQoS) rather than looking for the best one (i.e., GoodEff) could bring better results. This can be achieved by applying RBC in a hierarchical way, leading to the *hierarchical Rule Based Control* (*HRBC*) algorithm. In particular, in the first application, only the top class (i.e., GoodEff) has a negative weight; at the subsequent step, if the best output cannot be reached, then a new class (i.e., BadEff) is set with a negative weight.

Even if here it is applied to a three-class problem, the algorithm is easily generalized: the goal of the first application of RBC is to maximize the probability for the best class. If the goal is not achieved, the goal becomes less ambitious and consequently consists of moving the considered solution toward an acceptable class; at every step, the set of acceptable classes is enriched and the goal is less restrictive.

Each step of the algorithm consists of applying RBC; then, only if the predicted output does not reach the target classes at the last step, the weights are updated and the step is repeated. The score update consists of setting the minimum positive weight equal to the maximum between the current negative weights. In addition, all the negative weights are decreased, as shown in Algorithm 2.

The effectiveness of the different versions of RBC is tested in the next section using a synthetic network whose evolution is simulated via a hydraulic model.

Algorithm 2: Hierarchical Rule Based Control that iteratively implements RBC changing the target class.

Data: $\boldsymbol{x}$, $\mathcal{R}_{\boldsymbol{x},E}$, $\boldsymbol{w}$, E
Result: $\boldsymbol{x}^M$,
$C^- = \{c \in C \mid w_c > 0\}$;
$C^+ = C \setminus C^-$
while $C^- \neq \emptyset$ **do**
  $\boldsymbol{x}$ =RuleBasedControl ($\boldsymbol{x}$, $\mathcal{R}_{\boldsymbol{x},E}$, $\boldsymbol{w}$, E);
  $\tilde{y} = \arg\max_c \{P(c \mid \boldsymbol{x}) \mid c \in C\}$;
  if $\tilde{y} \in C^-$ **then**
    $c^* = \arg\min_c \{w_c \mid c \in C^-\}$;
    $w_{c^*} = \max\{w_c \mid c \in C^+\}$;
    for $c \in C^+$ **do**
      $w_c = w_c - 1$;
    end
    $C^- = C^- \setminus \{c^*\}$;
    $C^+ = C^+ \cup \{c^*\}$
  else
    $C^- = \emptyset$;
  end
end

4. Results

The validity of the RBC method was demonstrated by optimizing the control of a D-town network. This network was presented for the Battle of the Water Networks II, discussed by [26], whose objective was to define the best design improvement given a population's growing demand for water. The hydraulic model of D-town is available in the benchmark problems hosted by the Centre for Water Systems of the University of Exeter, which collects examples of water distribution networks used by various researchers in their studies related to water modeling and optimization (https://engineering.exeter.ac.uk/research/cws/resources/benchmarks/, accessed on 8 May 2023). The D-town hydraulic model consists of 399 junctions, 7 storage tanks, 443 pipes and 11 pumps divided into 5 pumping stations. The schema of the D-town network is presented in Figure 1.

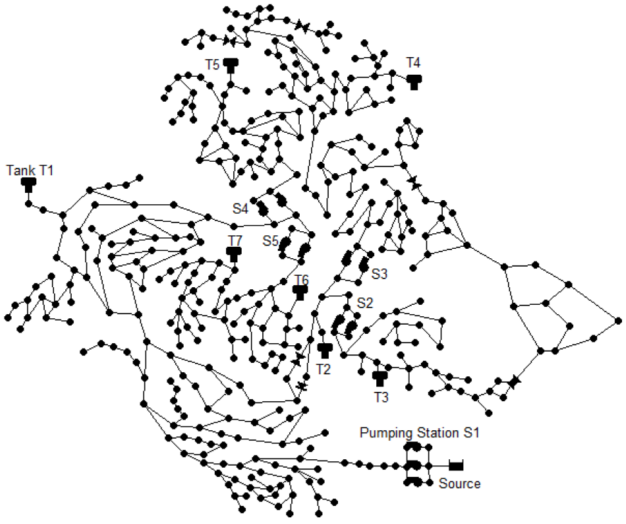

Figure 1. Schema of the D-town network.

The training dataset was generated by performing several EPANET simulations and collecting information on the elements of the network at each time step utilizing EPANET TOOLS 1.0.0, a package enabling the user to call all the EPANET programmer's toolkit functions within Python scripts. For each simulation, the input file was modified to simulate different water demands for each node and different initial states for tanks and pumps, leading to a different efficiency and quality of the service. The resulting dataset is a collection of 67K samples representing different network states. As described in Section 3.1, each sample was labeled with BadQoS, BadEff and GoodEff according to its efficiency (Eff) and quality (QoS) indicator. In particular, the efficiency indicator was computed by quantifying the energy consumed by pumping stations and the quality indicator was computed by observing whether the pressure of each node is in its optimal pressure range or measuring how much lower it is than its lower limit or greater it is than its higher extreme. More specifically, if the constraint violation for a pattern is greater than the median constraint violation in the training set, its output is set to BadQoS. If it is lower but the energy efficiency for the pattern is worse than the median energy efficiency in the training set, its output is set to BadEff. Finally, if both indicators are better than the median, the output is set to GoodEff. Afterwards, a three-fold cross-validation procedure was used to select the optimal LLM parameters for generating the rule-based model. The parameter set leading to the higher average Cohen's K in the validations sets was used to extract rules predicting the output value based on the level of the tanks and the number of active pumps in each pumping station.

To enhance the robustness of the evaluation, a separate set (named *test set*) of 30K simulated patterns was generated, and will be used for evaluation and comparison. In other words, RBC was applied, on top of the rules generated by the LLM in the training step, to drive the controls of the system (i.e., to turn on/off some pumps) in such a way that the QoS (firstly) and the efficiency (secondly) of the network are improved. The model trained by the LLM obtains an accuracy of 0.84 and a Cohen's K of 0.76 on the test set.

The QoS and efficiency measured after the control actions provided by the RBC approach were compared with the same indicators in two different scenarios. Firstly, the system indicators after applying RBC controls were compared with the system indicators if no actions had been taken. We will refer to this as the no action case. In the second scenario, RBC was compared with one of the most used control approach, where suggestions were generated by applying the genetic algorithm. The used genetic representation is the binary one, where each gene is a pump status; the fitness is a combination of efficiency and quality indicators, where their weight parameters were set in order to favor the former one; the selection method is the proportional selection; the crossover method is the single-point crossover with a probability equal to 0.9 and a mutation probability equal to 0.1. The population size was set to 100; the best solution was maintained without any change in each generation; and the termination conditions were: 40 generations were explored; 5 steps were performed without improving the best solution.

In all cases, the system indicators were evaluated by the EPANET simulator 15 min after the controls application.

It is noteworthy that the RBC approach only needs a pre-training rule-based model to generate the control suggestions. The genetic algorithms procedure, on the other hand, requires hydraulic simulations to compute the fitness used to select the best suggestions. For this reason, genetic algorithms can be used only after the formulation of a physical model representing, with some degree of approximation, the functioning of the water distribution network.

To set up the comparison, EPANET was used as a support for genetic algorithms. All the comparisons were made using 30K samples that are different from the training set but generated with the same approach. Each sample was given as an input to the control method and the effect of the control output was checked 15 min after the application through EPANET simulations. EPANET allows us to insert a demand profile as a factor that is multiplied by a standard water demand value. Since EPANET was used both to

select and to evaluate the solution generated by the genetic algorithms, Gaussian noise with an average equal to zero and standard deviation equal to 0.05 was added to the EPANET multiplier of the water demand in order to also take into account the approximation that characterizes the physical model of the system. To avoid excessive variances, the expected water demand profile used for GA optimization was simulated to differ 20% at most from the real demand.

The first comparison regards the different behaviors in terms of QoS . As shown in Figure 2, when RBC is applied, the occurrence of the BadQoS output in the next 15 minutes is avoided in 67% of cases. If no action is applied, the BadQoS is avoided in 59% of cases, whereas only 55% of patterns prevent the BadQoS output if genetic algorithms are applied.

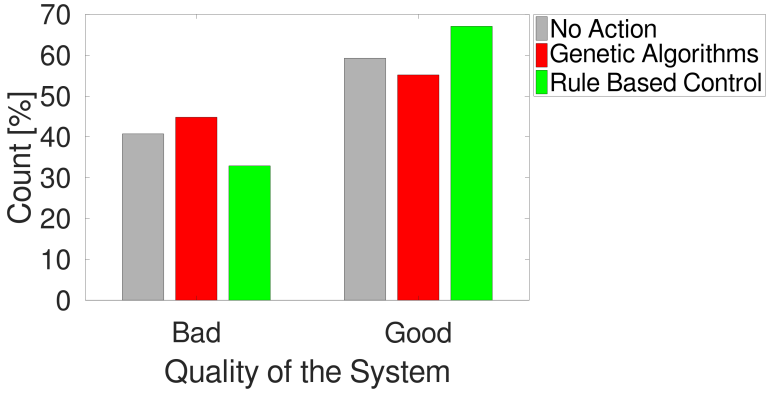

Figure 2. The performances of the three approaches concerning the quality of the system indicator.

Figure 3 reports the results of experiments when considering energy efficiency and puts the QoS in a broader perspective. As illustrated, genetic algorithms result in an improved level of consumption compared to the median in 96% of cases. If no controls are applied, this only happens in 52% of cases, whereas the RBC approach achieves 78% of good cases if energy efficiency alone is considered.

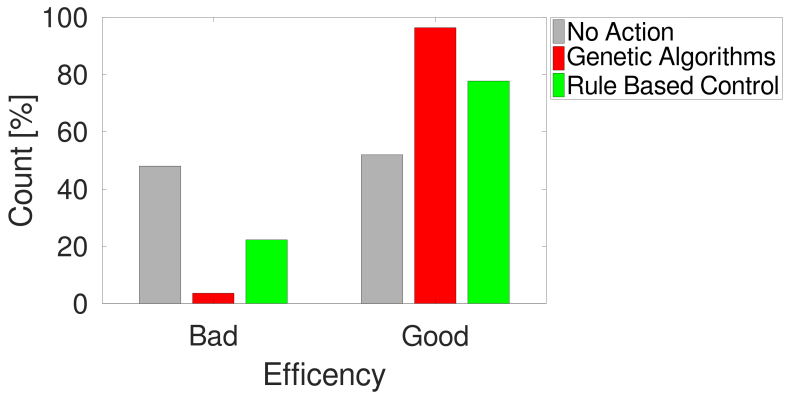

Figure 3. The performance of the different approaches regarding energy optimization.

Combining the two elements, it is possible to see (in Figure 4) that, referring to the BadQoS, BadEff and GoodEff possible output values, RBC obtains the lowest percentage of cases in the worst class (BadQoS) and the highest percentage of cases in the best class (GoodEff). Table 1 resumes the output distribution in the three cases.

Table 1. Comparison: output classes in test set.

	BadQoS	**BadEff**	**GoodEff**
No Action	40.73%	29.84%	29.43%
Genetic Algorithms	44.82%	1.77%	53.41%
Rule Based Control	32.91%	4.39%	62.70%

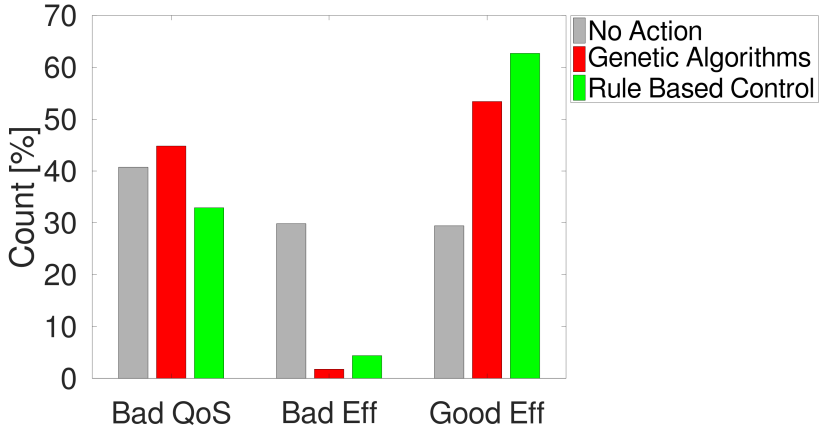

Figure 4. The performance of the three approaches considering the indicator that takes into account both quality of the service and energy optimization.

It is also worth considering that turning pumps on and off at a high rate in such a complex and critical system as a water main may not be desirable for stability reasons. To take this aspect into account, Figure 5 shows a comparison in terms of the average number of changes that are suggested to improve the performance. In this case, only RBC and GA can be compared because, in the no action case, by definition, the number of changes is 0. From the histogram, it can be seen that more than 60% of cases RBC suggest 0, 1 or 2 switches, while the frequency of a higher number of switches decreases quickly.

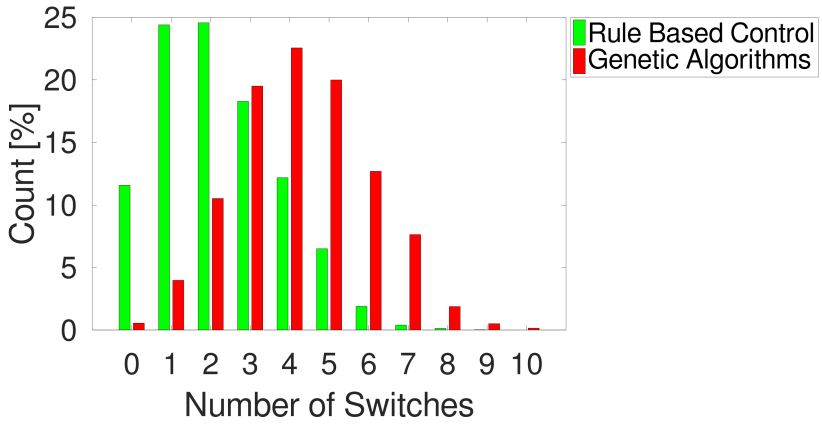

Figure 5. The number of switches performed by the two approaches at each iteration.

On the contrary, in more than 60% of cases, GAs propose more than 3 changes, almost never suggest 0 changes and suggest 1 switch in less than 5% of cases. The average number of changes per iteration for RBC is 2.25, compared to 4.24 for GAs. This means that, considering all the set of pumps during one day (i.e., 96 iterations of the control algorithm since it runs every 15 min), approximately 216 switches are expected for RBC whereas, on average, GAs suggest 407 switches.

As far as the computation time is concerned, on average, RBC requires 0.04 s for a single run, whereas a single run for the GAs requires 56 s.

The time needed for computation depends on the complexity of the network to be controlled. For this reason, the behavior of time needed for computation as a function of the complexity of the network (measured as the ratio of the number of nodes with respect to the original network) is shown in Figure 6. From this plot, it is evident that RBC is able to perform far quicker than GAs, even when the complexity of the network increases. This is not surprising as the model used by RBC is much more simple than the hydraulic simulator used by GAs. To better evaluate the time performance of RBC, we also plotted the time needed by the novel approach separately in Figure 7. For example, it takes less than 50 s to optimize a network that is 20 times more complex than the original with approximately 8000 demand nodes. This also allows for application where new control rules should be generated with a high frequency. Moreover, opposite to other approaches (such as linear programming or genetic algorithms), RBC allows us to (sub)optimize the network by also considering a portion of it, thus reducing the computational burden further.

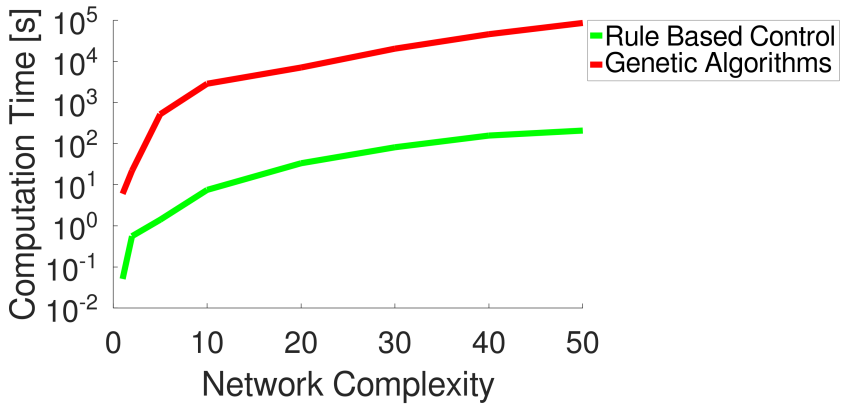

Figure 6. The time needed by RBC and GAs as a function of the complexity of the network (logarithmic scale).

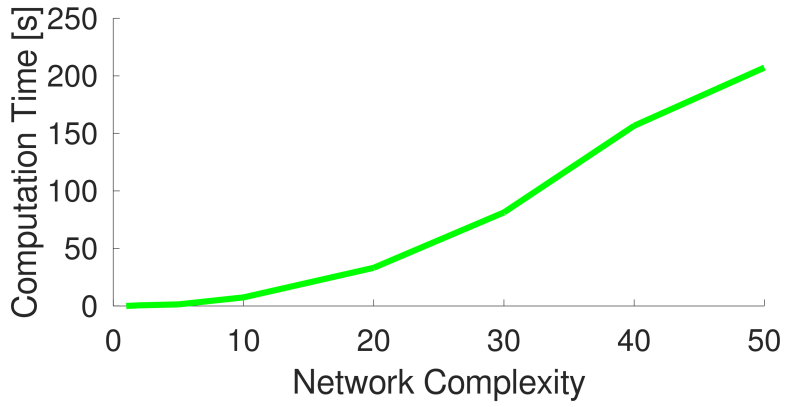

Figure 7. The time needed by RBC as a function of the complexity of the network.

Computational efficiency, combined with the fact that RBC does not require a mathematical model of the network, makes it particularly well-suited for a reactive, on-line control application.

5. Discussion

The present work introduced and described the rule-based control method, a new rule-based machine learning technique that can be used for the optimization of complex systems. Specifically, in this paper, we applied this method to a water distribution network to optimize pump controls, saving energy and ensuring a good quality of service, but it

is worth noting that the rule-based control approach can be used to optimize the control of other systems. It only requires the availability of training data so that the underlying classification algorithm can learn the behavior of the system under study. This is also a limitation, as it cannot be used to optimize a network if it does not have historical data and cannot be simulated. In order to measure how this method can improve the network quality, a test was performed on a separate test set using a hydraulic simulator. Our results suggest that rule-based control is able to not only improve the quality of service but also the energy efficiency of the network system. Furthermore, to demonstrate the effectiveness of the rule-based control methodology, it was compared with a more traditional method based on genetic algorithms. First of all, this comparison revealed that our approach allows for higher levels of network improvement; secondly, it reduces the number of switches in the pumps, which should imply lower maintenance costs for the water main; and, finally, it reduces the calculation time, thus enabling real-time implementations.

Future studies will focus on the application of RBC for network optimization, including rule extraction and the evaluation of results. New implementations of RBC will be investigated, where not just the next step time but also the subsequent steps will be considered. In this case, other factors, such as the cost of energy and demand forecasting, could also be considered. The application of RBC in optimizing the control of other types of systems will also be investigated.

Author Contributions: Conceptualization, E.F., D.V. and M.M.; methodology, E.F., N.P. and D.V.; software, N.P.; validation, N.P.; formal analysis, E.F. and D.V.; investigation, N.P.; writing—original draft preparation, N.P. and D.V.; writing—review and editing, E.F.; supervision, M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ghorbanian, V.; Karney, B.; Guo, Y. Pressure standards in water distribution systems: reflection on current practice with consideration of some unresolved issues. *J. Water Resour. Plan. Manag.* **2016**, *142*, 04016023.
2. Stinson, M.; Vitasovic, Z. Real time control of sewers: US EPA manual. In Proceedings of the World Environmental and Water Resource Congress 2006: Examining the Confluence of Environmental and Water Concerns, Omaha, NE, USA, 21–25 May 2006; American Society of Civil Engineers (ASCE): Reston, VA, USA, 2006; pp. 1–8.
3. Mala-Jetmarova, H.; Sultanova, N.; Savic, D. Lost in Optimisation of Water Distribution Systems? A Literature Review of System Design. *Water* **2018**, *10*, 307. https://doi.org/10.3390/w10030307.
4. Eusuff, M.M.; Lansey, K.E. Optimization of water distribution network design using the shuffled frog leaping algorithm. *J. Water Resour. Plan. Manag.* **2003**, *129*, 210–225.
5. De Corte, A.; Sörensen, K. An iterated local search algorithm for water distribution network design optimization. *Networks* **2016**, *67*, 187–198.
6. Creaco, E.; Franchini, M. A new algorithm for real-time pressure control in water distribution networks. *Water Sci. Technol. Water Supply* **2013**, *13*, 875–882.
7. Page, P.R.; Creaco, E. Comparison of flow-dependent controllers for remote real-time pressure control in a water distribution system with stochastic consumption. *Water* **2019**, *11*, 422.
8. Page, P.R.; Abu-Mahfouz, A.M.; Mothetha, M.L. Pressure management of water distribution systems via the remote real-time control of variable speed pumps. *J. Water Resour. Plan. Manag.* **2017**, *143*, 04017045.
9. Mosetlhe, T.; Hamam, Y.; Du, S.; Monacelli, E. A Survey of Pressure Control Approaches in Water Supply Systems. *Water* **2020**, *12*, 1732. https://doi.org/10.3390/w12061732.
10. Singh, M.K.; Kekatos, V. Optimal scheduling of water distribution systems. *IEEE Trans. Control. Netw. Syst.* **2019**, *7*, 711–723.
11. Bunn, S.M.; Reynolds, L. The energy-efficiency benefits of pump-scheduling optimization for potable water supplies. *IBM J. Res. Dev.* **2009**, *53*, 5:1–5:13.
12. Rao, Z.; Alvarruiz, F. Use of an artificial neural network to capture the domain knowledge of a conventional hydraulic simulation model. *J. Hydroinform.* **2007**, *9*, 15–24.
13. Rao, Z.; Salomons, E. Development of a real-time, near-optimal control process for water-distribution networks. *J. Hydroinform.* **2006**, *9*, 25–37.

14. Rylander, S.; Gotshall, B. Optimal population size and the genetic algorithm. In Proceedings of the World Scientific and Engineering Academy and Society (WSEAS) International Conference on Soft Computing, Optimization, Simulation, and Manufacturing Systems (SOSM 2002), Shanghai, China, 10–14 June 2002.
15. Mala-Jetmarova, H.; Sultanova, N.; Savic, D. Lost in Optimisation of Water Distribution Systems? A Literature Review of System Operation. *Environ. Model. Softw.* **2017**, *93*, 209–254. https://doi.org/10.1016/j.envsoft.2017.02.009.
16. Lopez-Ibanez, M.; Devi Prasad, T.; Paechter, B. Multi-objective optimisation of the pump scheduling problem using SPEA2. In Proceedings of the 2005 IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; Volume 1, pp. 435–442. https://doi.org/10.1109/CEC.2005.1554716.
17. Murphy, L.; Dandy, G.; Simpson, A. Optimum Design and Operation of Pumped Water Distribution Systems. In Proceedings of the 1994 Conference on Hydraulics in Civil Engineering, Brisbane, Australia, 15–17 February 1994.
18. Stokes, C.S.; Simpson, A.R.; Maier, H.R. A computational software tool for the minimization of costs and greenhouse gas emissions associated with water distribution systems. *Environ. Model. Softw.* **2015**, *69*, 452–467. https://doi.org/10.1016/j.envsoft.2014.11.004.
19. Patriksson, M.; Bagloee, S.; Asadi, M. Minimization of water pumps' electricity usage: A hybrid approach of regression models with optimization. *Expert Syst. Appl.* **2018**, *107*. https://doi.org/10.1016/j.eswa.2018.04.027.
20. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **1991**, *4*, 251–257. https://doi.org/10.1016/0893-6080(91)90009-T.
21. Rossman, L.A. *EPANET 2 Users Manual*; U.S. Environmental Protection Agency (EPA): Washington, DC, USA, 2000; Volume 38.
22. Muselli, M. Switching Neural Networks: A New Connectionist Model for Classification. In *Proceedings of the 16th Italian Workshop on Neural Nets, WIRN 2005, International Workshop on Natural and Artificial Immune Systems (NAIS 2005), Vietri sul Mare, Italy, 8–11 June 2005*; Apolloni, B.; Marinaro, M., Nicosia, G., Tagliaferri, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 23–30.
23. Muselli, M.; Ferrari, E. Coupling Logical Analysis of Data and Shadow Clustering for Partially Defined Positive Boolean Function Reconstruction. *IEEE Trans. Knowl. Data Eng.* **2011**, *23*, 37–50. https://doi.org/10.1109/TKDE.2009.206.
24. Quinlan, J.R. Induction of decision trees. *Mach. Learn.* **1986**, *1*, 81–106.
25. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. https://doi.org/10.1023/A:1010933404324.
26. Monteiro, A.J.; Salomons, E.; Ostfeld, A.; Kapelan, Z.; Simpson, A.; Zecchin, A.; Maier, H.; Wu, Z.; Elsayed, S.M.; Song, Y.; et al. Battle of the Water Networks II. *J. Water Resour. Plan. Manag.* **2014**, *140*, 04014009. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000378.