SCHEDULING PUMPS AND RESERVOIRS WITH INTEGER NONLINEAR PROGRAMMING AND DEEP LEARNING

Sophie Demassey[1], Valentina Sessa, Amirhossein Tavakoli
CMA, Mines Paris - PSL, CS 10 207, 06904 Sophia Antipolis, France
{sophie.demassey, valentina.sessa, amirhossein.tavakoli}@minesparis.psl.eu

KEY WORDS

mathematical optimization, demand-response, hydraulic network operation, drinking water distribution

ABSTRACT

The pump scheduling problem is to shift the pumping in a hydraulic network ahead of time of the forecasted water demand, so as to minimize the energy cost of pumping by leveraging the dynamic electricity tariff, together with the storage capacities of the reservoirs and the nonlinear flow dynamics. This optimization problem has a challenging mathematical nature, discrete and nonconvex. It also has a complex structure, comprising three interleaved decision levels: the control of pumps and valves, the flow-head equilibrium, and the storage level at all time periods. Many solution approaches approximate some of the coupling or complicating constraints, but the results of these procedures are hard to evaluate. In this paper, we focus on computing feasible solutions to the pump scheduling problem, and present a cooperative approach making use of deep learning to predict the reservoir level trajectories over the time horizon, integer programming to get the optimal combinations of pumps, and Todini's gradient algorithm to simulate the hydraulic equilibria independently on all periods. We implement an alternating direction method to synchronize these components. The decoupling enables a fast computation of each component, some could additionally be run in parallel. The synchronization algorithm, although it lacks a theoretical guarantee of convergence, succeeds in computing strictly-feasible nearly-optimal solutions on the well-known Van Zyl benchmark set.

1. INTRODUCTION

In the current trend towards decarbonizing the energy sector, hydraulic utilities, which combine energy-intensive pumps and energy-storage reservoirs, offer great opportunities for implementing efficient demand-response strategies, resulting in both substantial energy savings for these utilities and flexibility services for the power system. Load shifting in pressurized drinking water distribution networks is one emblematic example. There, pumping can profitably be scheduled ahead of time of delivering water, for three reasons: (i) the elevated tanks act as storage for both water and gravitational energy, (ii) the dynamic tariff of the electricity gives direct incentives for pumping when the power production is in surplus, and (iii) distributing the pumping effort allows reaching better efficient points given the nonlinear nature of the flow-pressure relation. Pump scheduling refers to anticipating the activation of pumps and valves according to a water demand forecast in order to minimize the energy cost of pumping.

In the static case, the demand and initial tank levels are fixed, and the problem is to find the most efficient combination of active pumps and valves, along with the associated network-wide flow-head equilibrium to serve the demand. The mathematical model is already challenging for conventional global optimization tools [1], as it involves both nonconvex head loss equations and combinatorial control decisions, formulated either as integer variables or complementary constraints. Note that this static problem has the same nature as the strategic design problem of pipe sizing in gravity-fed networks [2]. In the dynamic case considered in this paper, the water demand forecast is provided as an hourly time series for every service point on the day ahead. The time horizon is discretized accordingly to handle the pump and valve switch decision. The dynamic case then relies on solving a sequence of 24 static problems, interleaved pairwise by the storage conservation constraints in the water tanks. Its algorithmic complexity is still one magnitude higher.

[1] Corresponding author

An abundant literature is dedicated to methods for solving different variants of this problem, experimenting with various techniques. In the standard deterministic variant, pumps and valves have binary states – on/off and open/close – and all data, including water demand and electricity tariff, are known. The few published global optimization approaches [3][4], even when tailored to a specific case [5], quickly become computationally intensive as the network size or time discretization grows. Alternatively, most proposed algorithms trade off accuracy for speed, usually by truncating the search (e.g., in genetic algorithms [6] [7]) and by approximating some of the complicating constraints (e.g., using a piecewise linear hydraulic model with mathematical programming [8]). The quality of the computed approximate solutions cannot be properly evaluated using heuristic approaches, especially in the case of pump scheduling, where it is challenging to estimate how far an approximate solution is from complying with a reliable model of this problem. Indeed, the solution space (i.e., binary vectors of pump and valve configurations) is sparse: applying a local move in this space relies on turning a pump on or off for the duration of one time period, which strongly impacts the dispatch of water in the network from this time period and later. In the process of recovering the feasibility of an approximate solution, it likely leads to exceeding the tank capacity if the level is already close to it, as in the case when the tanks are well dimensioned and the approximate solution has been optimized accordingly. This sensitivity is an issue for both exact and heuristic algorithms, including end-to-end machine learning approaches, as they mostly rely on global or local search through this discrete space.

In this paper, we adopt an alternative strategy by searching through the continuous state space of the tank level trajectories. This strategy was previously applied in [9], where, based on numerical experiments and industrial experience, it was observed that a simple linear programming relaxation provides a good approximation of the optimal tank level trajectories. The authors argued that experienced network operators could manually turn this prediction into a practical pump schedule. We adopt the same two-step scheme but implement it with some elaborate and complementary decision tools. Namely, we build a deep learning model instead of a linear program to predict tank levels. Then, we apply an iterative splitting algorithm to a reliable mathematical model of the problem to refine this approximation and derive a feasible solution. The mathematical model is a mixed integer nonlinear nonconvex program (MINLP) based on an accurate analytical representation of the dynamics that govern the hydraulic system, thus it allows measuring the theoretical quality of the computed solutions. The more accurate, the more difficult to solve, so the complementary deep learning (DL) model, built from historical data or observations, serves as a preprocessing step to initialize the MINLP algorithm with approximate predictions.

The main motivation behind our algorithm comes from the multi-level structure of the dynamic problem and how it decomposes into independent and easy static problems once the tank levels are fixed. We exploit this by implementing a Douglas-Rachford splitting scheme, akin to ADMM tailored for this discrete nonconvex problem: each iteration consists in solving the two subproblems obtained by splitting the (pump) control variables and the (tank) state variables, where the control part splits itself into parallelizable static problems, after dualizing the intertemporal storage conservation constraints. This relates then to the Lagrangian decomposition model of [10] or to the recent ADMM adaptation of [11], but with one major difference: in our decomposition, the tank levels are not only decoupled in time, but they are fixed in each static control subproblem. This allows us to decompose these subproblems also spatially, following the decomposition of the network along the tanks, and to use a fast Newton method, namely Todini-Pilati algorithm [12] to compute the hydraulic equilibria for each static configuration on each network component independently. Observing that the number of pumps and valves is often limited in each component, we expect that enumerating the configurations and computing the hydraulic equilibria to be much more efficient than relying on a global optimization tool, as in [10][11]. Our approach also relates to the matheuristic of [13], which employs this enumeration as a preprocessing step to populate an extended linear programming approximation before running a final step to repair the hydraulic inaccuracies. In comparison, our iterative scheme can be seen as a dual alternative to column generation, where configurations are computed on the fly. This eliminates the need for that final repair step; however, in theory, it is not guaranteed to converge because the MINLP is discrete and nonconvex. We thus implemented simple meta techniques to make it work in practice, in particular: a restart mechanism to diversify the search and a penalization technique to drive the search, or a scaling mechanism to alleviate the DL training process.

This paper presents an original hybridization of machine learning and mathematical programming for tackling optimization problems with hard feasibility issues, using DL for handling the optimization objective and MINLP for recovering feasibility. The approach is flexible as the two components are independent, and one could be replaced without impacting the other. This approach is also original for the pump scheduling problem for drinking water distribution as it fully leverages the problem separability in elementary tasks – using temporal decomposition and network partition, and handling independently the hydraulic/nonconvex

constraints with simulation and the combinatorial decisions with controlled enumeration – while using a mathematical programming framework to coordinate the tasks and enforce the strict feasibility of the solutions. It is also flexible in the sense that it can be applied to other hydraulic or hydrologic planning problems involving tight storage conservation constraints at reservoirs, possibly on different time scales, and with any kind of discrete control decisions. The algorithm is not even restricted to water systems, as the principle of flow-head equilibrium occurs in many other physical or economic networks, such as power distribution or routing. We actually developed the methodology in [14] on an abstract system with storage capacities. The present paper details its application to the pump scheduling problem, specifically, and describes a new physics-informed deep learning model along with the corresponding experimental results.

The paper is organized as follows: Section 2 describes the mathematical model of the pump scheduling problem; Section 3 describes two deep learning models to forecast the tank level trajectories; Section 4 describes the splitting algorithm to repair the forecast trajectory and compute a feasible pump schedule; Section 5 provides experimental results on the benchmark network known as *Van Zyl* network.

2. MATHEMATICAL MODEL FOR THE PUMP SCHEDULING PROBLEM

We briefly present a standard mathematical model for the pump scheduling problem; more details can be found, e.g., in [3]. The notation used in this paper is described in Table 1.

t, T	time step index, horizon length
r, s	network node indices: reservoirs (tanks and sources), service nodes
$E, a = (i,j), n$	network arcs: incidence matrix, arc indices, number of controllable arcs (pumps, valves)
π	network partition (along the reservoirs): partition index
$x_{at}, q_{at}, h_{it}, v_{at}, l_{rt}$	variables: binary arc control, arc flow, node head, arc head loss, tank filling level
D_{st}, C_t	data: demand at service nodes and electrical cost
φ_a, f_a	arc characteristics: head loss function, electrical consumption (for pumps)

Table 1: Notations

The water distribution network is formalized as a directed graph where each node represents either a junction, a service node, a source, or a water tank, and each arc represents either a pipe or a pump. Some arcs are controllable: the fixed-speed pumps can be turned on or off, and the gate valves equipping some pipes can be open or closed. For ease of presentation, we consider such controllable devices with only two states (active/inactive), but our method can also be adapted to networks with variable speed drive pumps or pressure reducing valves, following either approach in [3] or [13].

The scheduling horizon is discretized in T time steps, so as the water demand, which is fixed over each time step, and the network operation is to decide the configuration of pumps and valves to activate on each time step (the noncontrollable arcs, such as the passive pipes are considered to be always active). This decision is formulated using boolean variables: $x_{at} = 1$ if arc a is active at time step t (i.e., flow can pass) and $x_{at} = 0$, otherwise. The transitory effect is usually neglected, and steady flow is assumed at each time step. Hence, the hydraulic equilibrium at time t is characterized by the flow q_{at} and head loss $v_{at} = h_{it} - h_{jt}$ through the active arcs $a = (i,j)$, and it derives from flow analysis in the current state of the network given by: the current arc configuration x_{at}, the demand D_{st} at service nodes s, and the filling level l_{rt} of the water tanks r. It is well known that such a steady flow q_t is uniquely determined by this input when each node has either a known demand (D_{st}) or a known head (assimilated to l_{rt}), and when the resistance function $v_{at} = \varphi_a(q_{at})$ in each arc a – corresponding to friction-induced head loss in pipes and to discharge in pumps – is strictly increasing. In [12], it is proved to be the unique solution of the following equation system for some head vector h_t :

$$h_{rt} = l_{rt} \ \forall r, \qquad E^T{}_s q_t = D_{st} \ \forall s, E \ {}_a h_t + \varphi_a(q_{at}) = 0 \ \forall a: x_{at} = 1, q_{at} = 0 \ \forall a: x_{at} = 0.$$

We denote by $g(q_t;\, x_t, l_t, D_t) = 0$ this system where the function g of q_t is nonconvex. Alternatively, the feasible steady flow q_t can be seen as the solution of a linearly-constrained strictly convex minimization problem, called the Content Model in [15] in the hydraulic case, or the distribution problem in [16] in general nonlinear flow networks. In [12], an efficient gradient algorithm is proposed to optimize this strictly convex program, then to solve the equilibrium system. It is implemented in hydraulic simulators, like EPANET [17].

The equilibrium problem is separable. Indeed, given a partition of the network along the reservoir nodes, the steady flow can be computed for each component independently, as the equation system breaks into as many components:

$$g(q;\, x, l, D) \;\; = 0 \;\; \Leftrightarrow \;\; q = (q^{\pi})_{\pi} \;\; with \;\; g(q^{\pi};\, x^{\pi}, l^{\pi}, D^{\pi}) \;=\; 0 \;\; \forall \pi$$

This characteristic is particularly useful in our approach as the number n_{π} of controllable arcs in each component π is usually small enough to envisage enumerating all $2^{n_{\pi}}$ possible combinations x^{π} [13].

The pump scheduling problem can then be stated as the following Mixed Integer NonLinear Problem, referred to as (MINLP) hereafter:

$$min \sum_t C_t . f(x_t, q_t) : \qquad (0)$$

$$\begin{aligned}
& g(q_t; x_t, l_t, D_t) = 0\,, && \forall t = 1,\dots,T && (1)\\
& l_{t+1} \;\; = \; l_t \; + \; E^T q_t\,, && \forall t = 1,\dots,T && (2)\\
& \underline{L_t} \; \le l_t \le \underline{L_t}\,, && \forall t = 1,\dots,T+1 && (3)\\
& x_t \;\; \in \{0,1\}^A, && \forall t = 1,\dots,T. && (4)
\end{aligned}$$

As shown in (2), the tank levels l_{t+1} at the next time step depend linearly on the incoming flows q_t; moreover, in (3), tank levels are bounded by the tank capacity and fixed at the beginning and at the end of the horizon. Finally, in (0), the energy consumption can be reduced to a linear function f of the activity x and flow q through the pumps, where coefficients are obtained from the pump characteristics, and the energy cost is defined by the dynamic electricity tariff C_t. As mentioned, the function g is nonconvex in q_t, with x_t and l_t being additional variables of the system $g(q_t;\, x_t, l_t, D_t) = 0$ in the above formulation. Even for small-size problems, off-the-shelf global optimization solvers usually fail to find a feasible solution after hours of computation, and specialized branch-and-bound methods based on relaxing [1] or approximating [7] the nonlinear part all struggle in closing the optimality gap. This model makes explicit the time-separable structure of the problem after dualizing the time-coupling constraint (2), as implemented in the Lagrangian decomposition-based approaches of [10] and [11]. However, in these approaches, the tank levels remain unknown in the subsystems (1), preventing the application of Todini's gradient algorithm to solve them. We propose instead to adapt the Douglas-Rachford scheme to the dualized problem by splitting the state variables l_t and the control variables (x_t, q_t). Conceptually, our algorithm searches through the space of tank level trajectories l and attempts to derive a matching pump schedule x and network flows q by gradually repairing the relaxed constraint (2). To initialize the algorithm, a good prediction of the optimal tank level trajectories is required. To compute such a prediction, Ulanicki et al [9]. proposed to solve the continuous relaxation of the mathematical model above (without the integrality constraints (4)). In our approach, we opt for a numerical model, learned from data, to complement the analytical model presented above.

3. DATA-DRIVEN MODELS OF THE OPTIMAL TANK TRAJECTORIES

The evolution of drinking water consumption at the local scale of a distribution network exhibits high daily cyclicality and often significant seasonal variability. Still, it changes little from one year to the next. This motivates the use of the history of a network operation to predict the future daily operation using supervised machine learning methods. One data-driven approach to tackling combinatorial optimization problems is known as end-to-end learning, which consists of learning the solution to mathematical optimization problems directly from data [18]. Such a data-driven model does not offer a guarantee of optimality, nor even feasibility. When applied to pump scheduling, in place of the MINLP model above, to predict the optimal vector of binary control variables x, it is not even trivial to turn such an approximate prediction into an actual solution (x, q, l) satisfying the tight and intricate constraints (1)-(4) simultaneously. Instead of predicting the discrete control variables x, we propose predicting the continuous storage state variables l. We expect that the feasible solution set is denser when projected into the l-space or, at least, that feasible solutions exist at a short distance of the prediction in this continuous space, and, in the next section, we present an effective local search approach based on lagrangian decomposition, to repair this prediction and reach such a feasible solution. This section describes the deep learning architectures we devised for building the numerical model, and two original components we implemented, either for managing scalability or predicting multiple outcomes.

3.1 Deep Learning Architecture

Our machine learning problem is to build a hypothesis function, denoted as Θ_0, mapping an input given as a set of time series representing the time-of-use energy price and the demand profiles at each service node, to the target, that is, the time series representing the optimal tank trajectories, one per tank. The hypothesis is built during the training phase from a precomputed collection of input/target tuples, by approximately minimizing a standard loss function: the mean squared error between the targets and the mapping outcomes. To minimize the loss, the hypothesis function needs to capture both spatial and temporal local dependencies in the input and target data. However, a feedforward architecture requires flattening the input data, which might fade these patterns. Following the lead of previous works, e.g. [19], we choose to integrate the capabilities of Bidirectional Long Short-Term Memory (Bi-LSTM) networks for sequence understanding and Convolutional Neural Networks (CNNs) for spatial pattern recognition [20]. LSTMs are a type of Recurrent Neural Networks (RNNs) that is well-suited for tasks involving sequential data, including time-series forecasting. RNNs are neural networks with feedback loops to detect sequential patterns, and LSTMs use, in addition, a gating mechanism to train the network on long sequences of data. A BiLSTM can handle information in both forward and backward temporal directions, which is crucial in our application due to the dynamic relations spanning the entire scheduling horizon. On the other hand, CNNs are primarily recognized for their efficacy in computer vision tasks. Yet, they can also be utilized on time series to extract temporal features.

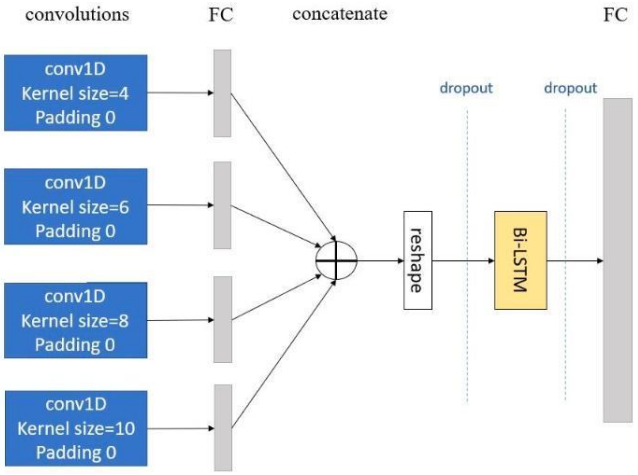

Figure 1: Scheme of the proposed CNN-LSTM architecture for learning hypothesis Θ_0.

In our architecture, depicted in Figure 1, the CNN consists of parallel one-dimensional convolutional layers (conv1D) with different kernel sizes located parallel to each other with zero padding. The kernel strategy enables sliding along the input sequence to identify local temporal patterns through an element-wise inner product operation between kernel weights and input data. Varying the number and size of kernels enables the model to recognize various patterns in the sequence. In relation to the time horizon, we considered 32 kernels in each layer with kernel sizes of 4, 6, 8, and 10. To handle the temporal dimension of the target (tank level trajectories), the output of the layers is passed through a hidden layer with ReLU activation functions, and their outputs are concatenated, reshaped, and fed into a Bi-LSTM. In the output layer, the prediction results are from a fully connected layer with a linear activation function placed after the Bi-LSTM unit.

3.2 Physics-Informed Neural Network

Hypothesis Θ_0 is solely learned from training data and the approximate minimization of the loss function, a distance between predicted and ground truth solutions. Thus, its outcome is not guaranteed to satisfy the physical constraints of the pump scheduling problem. Some simple variable domain restrictions (e.g., box constraints) can be enforced into the deep learning architecture or in a post-processing step. But, in general, there is no straightforward way to specify analytical constraints as modeled by (1) and (2), not even individually. Supervised penalty (see e.g., [21]) can be seen as a soft approach for handling the constraint violations by adding penalty terms to the loss function. The penalty terms are weighted by parameters that control the trade-off between minimizing loss and infeasibility. In our experiments, we set them to 0, 0.05, and 0.1. Minimization remains tractable, while the penalized constraints are convex. In our case, equations (1)

and (4) reveal two different types of nonconvexity. We propose addressing this issue, first by separating the two nonconvexities, then by resorting to convex relaxations, taking the continuous relaxation of equations (4) and penalizing equations (3).

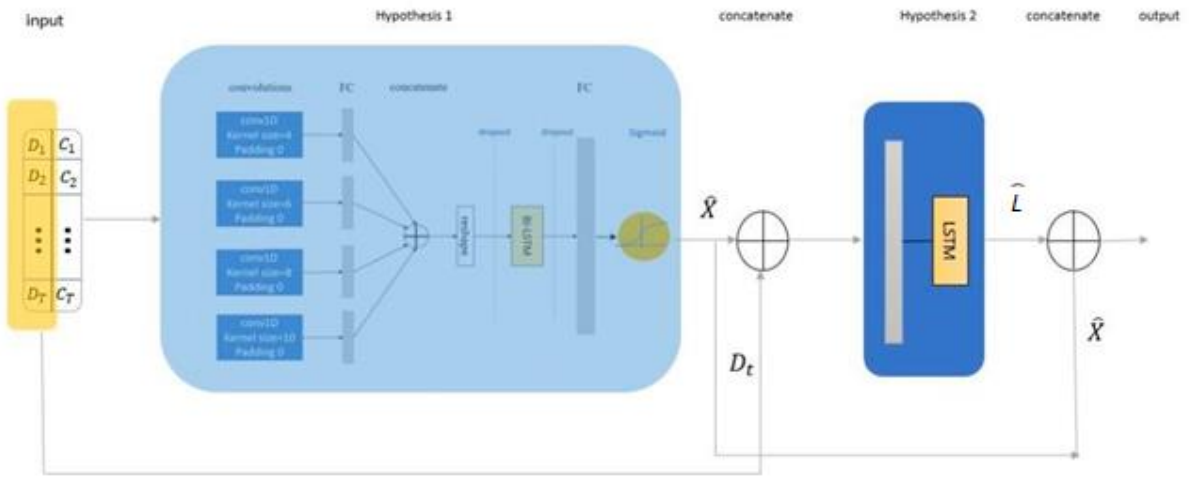

Figure 2: Scheme of the proposed DL architecture for learning the combined hypothesis $\Theta_1 + \Theta_2$.

Hence, our second DL architecture is made of two blocks, as depicted in Figure 2. The first block builds the hypothesis $\Theta_1$to predict the binary variables via classification, given the cost and demand profiles. The following block creates a map Θ_2 from the predicted binary values, together with the demand profile, to the tank profile. It is in this second block that physical constraints are considered.

The first block is a slight modification of the CNN-BiLSTM architecture developed to build the previous hypothesis Θ_0. In this case, the goal is to predict the optimal (binary) pump and valve control trajectories. Therefore, we replace the shape of the output layer with a sigmoid activation function to produce fractional values inside the interval [0,1]. The second hypothesis $\Theta_2$is a surrogate model of the feasibility subproblem (known as the extended analysis problem) defined by equations (1) and (2), i.e., Θ_2 maps the pump and valve control trajectories x to the unique tank level trajectories l satisfying these two constraint sets. This block consists of LSTM units and feedforward layers with ReLU activation functions.

The second hypothesis Θ_2 can be learned offline, as a preprocessing step. At this aim, we generate a training set of input/target pairs (x, l) by solving the extended analysis problem for each binary matrix input x: the target l is built iteratively, starting from the initial tank levels l_1, then computing for each period t, the hydraulic equilibrium satisfying $g(q_t, x_t, l_t, D_t) = 0$ (using Todini's algorithm), then the tank levels at the next period according to $l_{t+1} = l_t + E^T q_t$. Within the whole architecture, the hypothesis Θ_2 receives as input the output of the mapping Θ_1, which are fractional values in the interval [0,1]. We propose to use *data augmentation* (see e.g., [22]) to train Θ_2 also on fictitious fractional inputs: we populate our training data set by simply adding, for each input/target pair (x, l) in the collection (where x is binary), 200 new pairs(x', l) where x' is obtained by perturbing the matrix x by adding to each 0 entry and subtracting from each 1 entry a random variable from a uniform distribution in the interval [0,0.5].

Once Θ_2 is learned, we use *fine-tuning* to train the whole architecture [22]. This means that the weights of the model Θ_2 are frozen, so that only the weights in the first block are updated. We do not consider the tank capacities (3) in this process. However, when training the whole architecture, we penalize the violation of constraints (3) when evaluating the loss function:

$$\Lambda_{loss} = \Lambda_{CE}(X, \widehat{X}) + p|\hat{L} - \underline{L}|^- + p|\hat{L} - \underline{L}|^-$$

where Λ_{CE} denotes the classical binary cross-entropy loss (to evaluate the accuracy of predicting the binary variable), and the parameter p is a hyperparameter to control the compromise between minimizing the binary prediction and violating the physical constraints (3). Note that, unlike the approach described in the previous

architecture, we do not track the optimal tank profile here; instead, we only require feasibility. The shape of the tank profile is dictated by the map Θ_2 built to approximate the extended analysis problem.

3.3 Scaling Mechanism

DL models have a data-intensive nature, requiring a large dataset for training. This means that for training the two DL architectures described above, we need to collect the solutions of a large number of pump scheduling problems. For a real-world network, the training dataset could be built from the history of daily operations. On the other hand, if no history is available or if we assume that the historical operating decisions are not optimal, we must generate the training set from the optimal solutions of the analytical model (MINLP). However, even computing a feasible solution may require minutes or hours when considering a fine-grained time discretization of the scheduling horizon, e.g., for half an hour time steps ($T = 48$). To generate a training set that is large enough, the problem must be drastically simplified, without losing the structure of the optimal solutions. In this paper, we propose to act on the time resolution rather than the complicating constraints. Exploiting the elasticity of the scheduling horizon, we generate the training set by solving the problem for a coarse-grained time discretization of 2-hour duration steps ($T = 12$). By reducing the number of time steps, we also reduce the number of binary variables in (MINLP), consequently increasing its tractability. On medium-size networks, specialized algorithms can then compute a (feasible) solution in seconds and even prove optimality. This scaling mechanism allows reducing the computational cost of dataset generation, without directly compromising feasibility or optimality, as constraint relaxation would do. Reducing the size of the input and output also allows for reducing the computational cost for training the DL models. Finally, when running the learned model on a fine-time resolution instance, we simply need to scale the input and output time series up and down, e.g., through resampling and linear interpolation. The inaccuracies resulting from both the learning model and the scaling mechanism are expected to be corrected by the splitting algorithm in the second step of our procedure.

3.3 Diversified Predictions

Our two-step predict/repair procedure can be viewed as an original hybrid local search heuristic, using DL to generate a candidate solution driven by optimality, and MINLP to search for an actual solution, driven by feasibility, in the neighborhood of the candidate (without guarantee of finding one). The second step thus corresponds to the *intensification phase* of the heuristic, and the *diversification phase* could be simply implemented by generating more than one candidate. We propose generating multiple outcomes from the DL step by using the Monte-Carlo dropout method, originally developed for quantifying the uncertainty of neural networks, by approximating Bayesian variational inferences [23]. Simple and efficient, it works by randomly dropping out a fraction of neurons during training. In our architecture, the dropout layers are placed before the Bi-LSTM unit and just before the last fully connected layer that yields the outcomes (see Figure 1 and Figure 2). This approach yields a collection of distinct models, arising from masked neurons. As a result, for one input (tariff and demand profiles), we obtain multiple outcomes (level profiles), one from each model. The repair step of our procedure can then be applied to each outcome, independently, thus possibly in parallel.

4. SPLITTING ALGORITHM TO RESTORE FEASIBILITY

This section presents an iterative algorithm, related to the Alternating Direction Method (ADM) [24], based on the partial splitting of the control (x, q) and state l variables in (MINLP). ADM for separable problems and its variant ADMM [25] for handling linear coupling constraints, are very popular methods in large-scale convex optimization. The theoretical convergence to a stationary point is also established in some nonconvex cases, but under conditions that are not met in our case. As such, a splitting scheme would offer no theoretical guarantee of convergence, so we present a specific adaptation and its practical motivation.

First, we duale the time coupling constraints (2) with multipliers μ as follows:

$$\mathcal{L}(\mu) = min_{(x,q,l)}\ \mathcal{L}(x,q,l,\mu)\text{: (1), (3), (4) } with\ \mathcal{L}(x,q,l,\mu)$$
$$= \sum_t\ (C_t.f(x_t,q_t)\ + \mu_t(l_{t+1}\ -\ l_t\ -\ E^T q_t))$$

This allows leveraging the temporal separability of the resulting Lagrangian subproblems:

$$\mathcal{L}(\mu) = \sum_{t=1..T}\mathcal{L}_t(\mu)\ + \mu_{T+1}\ l_{T+1}\quad with$$
$$\mathcal{L}_t(\mu) = min_{(x_t,q_t,l_t)}\ C_t f(x_t,q_t) + (\mu_{t-1} - \mu_t)l_t\ -\ \mu_t E^T q_t :\ g(q_t;x_t,l_t,D_t) = 0, \underline{L}_t \le l_t \le \underline{L}_t,\ x_t \in \{0,1\}^A$$

The maximum of the dual function $\mathcal{L}$ usually provides a good lower bound for (MINLP). Furthermore, $\mathcal{L}$ is piecewise linear concave, so it can be efficiently maximized with either subgradient, bundle, or cutting-plane algorithms. During the proposed iterative process, each evaluation at a trial point μ returns a nearly feasible solution for (MINLP), which can potentially be turned into a feasible solution, then an upper bound for (MINLP). This decomposition is exploited in the Lagrangian relaxation approach of [10]. More recently, in [11], a standard ADMM is applied to the continuous model of the pump scheduling problem (replacing the integrality constraints with complementary constraints), based on a decomposition conceptually similar to the one we propose: they duplicate the state variables and dualize these new linking constraints. However, in such decompositions, the static subproblems $\mathcal{L}_t(\mu)$, while much smaller, remain hard to solve as they keep the same discrete nonconvex nature of the original problem.

Our first proposition is to evaluate $\mathcal{L}(\mu)$ through ADM, that is to enforce a control/state variable splitting, then to solve alternatively the two created subproblems: a control subproblem (update (x,q) by solving $\mathcal{L}(\mu)$ with fixed l), and a state subproblem (update l with fixed (x,q)). This approach can not only leverage the temporal separability, but also, as discussed in Section 2, the spatial separability and computational simplicity of the hydraulic equilibrium system $g(q_t;\ x_t,l_t,D_t) = 0$ in constraints (1) once the tank levels l_t are fixed. Since constraints (1) are coupling, then the variable split can be enforced by dualizing them (as in ADMM). However, this iterative procedure may not converge as the function g in (1) is not biconvex in x and l, and the integrality constraints (4) are not qualified. Furthermore, constraints (1) are structuring, and their dualization impoverishes the subproblems.

Hence, our second proposition is to enforce the variable split, not by dualizing constraints (1), but by keeping them as hard constraints in the control subproblem (as it conserves the property of spatial separability) and relaxing them in the state subproblem. Our adaptation of ADM leads to the following algorithm to evaluate $\mathcal{L}(\mu)$:

0. Initialize k=0 and l^0 with tank level trajectories (not necessarily feasible). Choose a positive tolerance ε and a maximum number of iterations K.
1. Fix $l = l^k$, solve $min_{(x,q)}\ \mathcal{L}(x,q,l^k,\mu)$: (1), (4), and update (x^{k+1},q^{k+1}) with the solution.
2. Fix $(x,q) = (x^{k+1},q^{k+1})$, solve $min_l\ \mathcal{L}(x^{k+1},q^{k+1},l,\mu)$: (3), and update l^{k+1} with the solution.
3. If (l^{k+1},q^{k+1}) solves (2), then STOP: solution $(x^{k+1},q^{k+1},l^{k+1})$ solves (MINLP).
4. Otherwise, if $\|l^{k+1} - l^k\|_\infty < \varepsilon$ or $k > K$, then STOP.

Algorithm 1: Adapted ADM to evaluate a solution to $\mathcal{L}(\mu)$ and (MINLP) heuristically

The two optimization subproblems solved at Steps 1 and 2 of Algorithm 1 can be solved efficiently. The first one is the control subproblem. It is separable in time and space and consists, for each time step t and each component π of the network partition, of enumerating the possible configurations ${x_t}^\pi$ of pumps and valves in the partition, then to generate the unique associate hydraulic equilibrium ${q^\pi}_t$ (given that the tank levels are fixed) and the corresponding cost that we denote $c({x^\pi}_t)$. The optimal solution is obtained by considering, for each time t, the configurations ${x^\pi}_t$ in every component π that altogether minimize $\Sigma_p c({x^\pi}_t)$. The second state subproblem is a small linear program.

Our application of ADM is specific as it focuses on feasibility rather than optimality: conceptually, Algorithm 1 searches through the l-space of the tank level profiles satisfying the capacity limits, and attempts to derive a matching control profile (x,q) by gradually reconciling the states l_{t+1} and $l_t + q_t$ at each time t, which is the stopping test at Step 3. As this adaptation does not allow for recovering the theoretical convergence of ADM, the stopping test at Step 4 limits the number of iterations. This algorithm remains a heuristic, so the computation of a solution to (MINLP) is not guaranteed. In compensation, we act on the input to the algorithm by running it from various promising initialization points l^0. Indeed, as described in Section

3.3, the initialization points l^0 are taken as outcomes of our deep learning model. They correspond to nearly feasible and optimal tank level trajectories. Therefore, we expect that actual feasible solutions exist in their neighborhood, and that our local repairing procedure in Algorithm 1 is able to reach them. The restart procedure with different candidates increases this expectation.

Finally, the lack of theoretical convergence allows taking freedom in the proposed scheme, as it is not restricted to convexity, smoothness, or regularity properties. In particular, there are many possible implementations of this scheme, depending on (i) the choice of the dualization, penalization, or partition of the variable-coupling constraints, and (ii) the policy for updating the multipliers or penalties. For example, a loyal adaptation to ADMM would be to limit Algorithm 1 to just one iteration, then update the multipliers μ with some gradient information derived from the result (x^1, q^1, l^1). For the experiments presented in the next section, we opted for another approach, more similar to the penalized ADM proposed in [26]. First, we manage constraints (2) with penalization, instead of dualization, using the L1 regularization form $p_{rt}(l,q) = |l_{r,t+1} - l_{rt} - E_r{}^T q_t|$. Hence, in the model, $\mathcal{L}(\mu)$ we redefine the objective function as:

$$\mathcal{L}(x,q,l,\mu) = \sum_t C_t.f(x_t,q_t) + \mu_t p_t(l,q),$$

where μ now represents nonnegative penalties instead of multipliers. When Algorithm 1 terminates at Step 4, we restart it after increasing the penalties to push towards satisfying (2). Note that this objective function, and then problem $\mathcal{L}(\mu)$, are not separable in time, but our variable splitting procedure maintains this separability: it only changes, at Step 1, the way to combine the configurations $x^\pi{}_t$ over all components π, to minimize the nonlinear cost (and not the way to compute the equilibrium for each configuration individually).

5. COMPUTATIONAL RESULTS

This section presents computational results for the experimental evaluation of the proposed hybrid DL-MINLP algorithm. It takes up and completes the results presented in [14]. A complete description of the dataset generation procedure, and of the DL architecture parameters, is available in the PhD thesis [27] and data and generation codes are available at:
https://github.com/amirhtavakoli94/bench_pmpscheduling.

5.1 Experimental Protocol

We run experiments on generated instances of the pump scheduling problem for the benchmark hydraulic network [7], known as *van Zyl*.

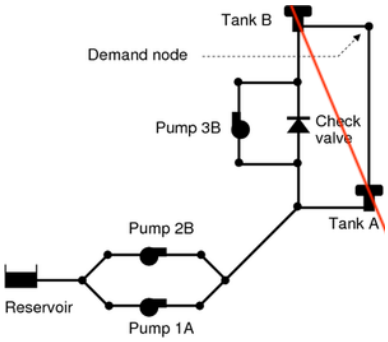

Figure 3: Schema of network *van Zyl* [7] and, in red, a cut through the two tanks which creates a partition of two subgraphs: the one on the left has 4 controllable arcs and no service node, the other on the right has no controllable arcs and one service node.

The network, depicted in Figure 3, consists of two water tanks of different capacities, two service nodes with individual loads, and four controllable elements: two symmetrical pumps and a boost pump operating parallel to a check valve. This is a fictitious, medium-sized but difficult network, often used for empirical evaluations. A collection of 2113 instances has been generated to train (75% of the instances), validate (15%), and test (10%) the DL model. Demands are drawn from a three-year highly seasonal history of real consumption sourced from a public network based in a tourist zone in Brittany, France, and electricity tariffs taken from the Belgian spot market data for the reference period from 2007 to 2013. Demands and tariffs are given as daily profiles of 2-hour time steps (i.e., $T = 12$) and optimal solutions have been computed for these instances using the specialized branch-and-cut algorithm of [3], denoted *BC* hereafter. The DL architectures are trained on

these data, and sized accordingly. They are built using Tensorflow API v2.12.0 in Google Colab with GPU A100.

The hybrid DL-MINLP algorithm, denoted *HA*, is tested on 50 instances, also using finer-grained temporal discretizations: with $T = 24$ (1-hour time steps), and $T = 48$ (30-minute time steps). As said in Section 4, implementing this variable-splitting scheme can be done in various ways. In these experiments, we tested the approach with penalization (not dualization) of the time coupling constraints (2). Algorithm 1 is run from at most 35 initialization points l^0 computed using the DL model. Penalties are updated at most 4 times for each run, and the maximum number of iterations is set as $K = 85$ for each penalty update. We initialize the penalty terms μ_{rt}, uniformly for all tanks r and times t, either to value 2 or 50; we denote *HA2* and *HA50* the corresponding algorithms. The penalties are updated/increased according to the update number u ∈ {0,..., 4} and the constraint violation. Precisely, given a random value ξ ∈ [0.75, 1] and $a = \xi e^{(-u/10)}$, the penalty μ_{rt} is updated to $(1 + 5a\mu_{rt})$ if $p_{rt}(l, q) > 10^{-6}$ and to $(1 + 2a\mu_{rt})$ otherwise.

We evaluate the quality of the returned solutions and the computation time, in comparison with the branch-and-cut algorithm *BC* and its variant, denoted *BCpre* [28], which implements an enhanced preprocessing. For the sake of comparison, the algorithms are all run on one thread of the same machine, i.e., without parallelization, (a personal laptop with x86 microprocessor at 2.40GHz and 128 GB memory). They employ the same implementation of Todini's algorithm (in Python, adapted to a quadratic form of the head loss functions) and Gurobi v10.0.1 to solve MILP. They are stopped at the first computed feasible solution found (within a 10^{-6} feasibility tolerance) or when reaching the computation time limit to 1 hour and 2 hours, for $T = 24$ and $T = 24$, respectively.

5.2 Evaluation of the Prediction Step

We first evaluate the prediction accuracy of the proposed DL models: the initial hypothesis Θ_0 and the physics-informed hypothesis $\Theta_1+\Theta_2$, as well as the two internal models Θ_1 and Θ_2, independently. We also compare it to a conventional feedforward (FW) neural network with a proportional depth and number of parameters (Θ_0 and FFW have 45k and 63k parameters, respectively).

We considered classical metrics to evaluate accuracy in regression problems, namely the mean absolute and squared errors (MAE and MSE) between the predicted output and the target. For comparison, we also report the normalized MAE (nMAE), i.e., MAE divided by the range of the ground truth, and the Pearson coefficient (R) to measure the linear correlation between targets and predictions. The prediction accuracy of Θ_0 is good: MAE=0.53, MSE=0.64, nMAE=9.2%, R=0.772. In comparison, the simpler architecture FW gives MAE=0.89, MSE=1.56, nMAE=12.7%, R=0.724. We can visualize the performance of our DL model Θ_0 on one random test instance in Figure 4. It depicts the filling levels of the two water tanks in the target/optimal solution and the prediction.

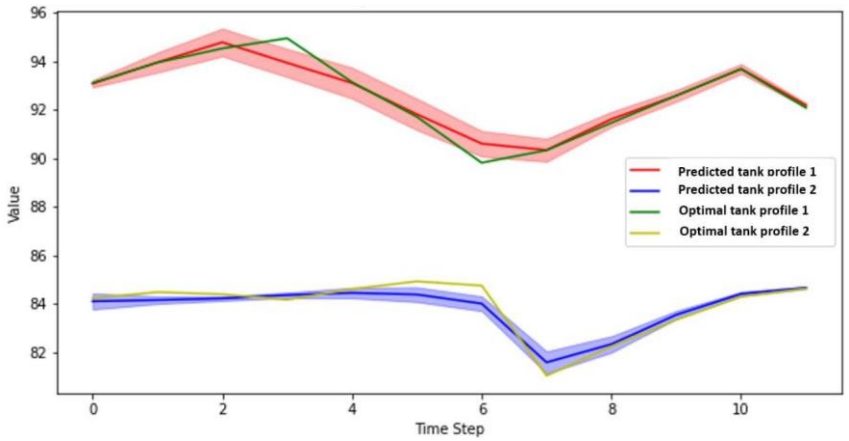

Figure 4: Comparing optimal and Θ_0 predicted solutions on one random test instance.

We now evaluate the accuracy of our second, physics-informed model $\Theta_1+\Theta_2$ described in Section 3.2. This is composed of the binary classification model Θ_1 to predict the optimal profiles of the configuration of pumps

and valves, and the regression model Θ_2 to derive the tank level profiles. The complete model $\Theta_1+\Theta_2$ aims to minimize a penalized loss function. In the experiments, we considered the penalty values $p = 0,0.05,0.1$. Model Θ_2 mimics the extended analysis problem and is also trained on fractional inputs, via data augmentation. Its accuracy is high, as indicated by MAE=0.11 and MSE=0.027. We evaluate the performance of Θ_1 according to the value of the binary classification accuracy, which is the rate of correct predictions. Increasing the penalty parameter decreases the accuracy value: from 0.84 for p=0 to 0.75 for p=0.05 and 0.74 for p=0.1. This seems natural as the penalty cost reduces the weight of accuracy in the loss function. In return, we expect that the predictions of the physics-informed model are closer to being feasible than the predictions of Θ_0.

There is no trivial way to evaluate this distance, however, we can evaluate the computational effort required to repair the infeasibility when applying our splitting algorithm directly to the outcomes of DL (without scaling). Hence, Table 2 shows the computational times of the hybrid algorithm when applied to the 50 test coarse-grained instances with T=12, using either DL models Θ_0 (*HA*) or Θ_1+Θ_2 (*PHA*), initial penalty value of the splitting algorithm $\mu \in \{2, 50\}$, and penalty coefficient of the loss function $p \in \{0, 0.05, 0.1\}$. We also report the time required for the global optimization algorithms *BC* and *BCpre* to reach a first feasible solution.

T	algorithm	p	μ	# solved	avg CPU (s)	max CPU (s)	avg GAP (%)	max GAP (%)
12	HA		2	44	*305*	*1577*	*4.6*	*11.3*
	HA		50	49	*254*	*1570*	*6.6*	*21.2*
	PHA	0	50	49	*131*	*510*	*4.8*	*12.8*
	PHA	0.05	50	**50**	**33**	392	11.1	22.3
	PHA	0.1	50	49	*45*	*438*	*10.9*	*23.7*
	BC			48	*121*	*681*	*5.4*	*12.5*
	BCpre			**50**	124	**137**	**4.3**	**12.4**

Table 2: Computational results on the 50 instances of van Zyl T=12 for different algorithms and parameters p and μ: number of instances solved (to feasibility) in 30 minutes, average and maximum computation time in seconds to reach a feasible solution, and average and maximum gap of the solution values with the optimum values (over the set of instances for which a solution has been computed: the statistics are shown in italic when this does not correspond to the whole set of instances, best results are typed in bold face).

The time limit is set to 30 minutes. These problems are small, however, we see that the state-of-the-art solver *BC* fails to compute feasible solutions within this time limit for 2 instances. At least one instance is also challenging for all hybrid algorithms except for the physics-informed version *PHA0.05*, which is able to find a feasible solution for all instances very quickly, in a maximum of 7 minutes and in 33 seconds on average. The comparison with *HA50* and *PHA0* tends to confirm the impact of penalization on predicting solutions close to being feasible. However, the predicted solutions appear to be less optimal, as indicated by the average and maximal optimality gaps reported in the last two columns of Table 2.

5.3 Evaluation of the Repair Step

We now evaluate the ability of the whole hybrid algorithm to find feasible solutions, and apply it to the 50 instances with T=24 and T=48. As stated in Section 3.3, we then rescale the output of the DL model (12-steps profiles) to get the input of the splitting algorithm.

T	algorithm	μ	# solved	avg CPU (s)	max CPU (s)	avg GAP (%)	max GAP (%)
24	HA	2	**50**	**279**	1711	8.4	**11.3**
	HA	50	**50**	**285**	**1257**	9.5	21.2
	BC		5	*1097*	*3117*	*11.1*	*12.5*

T	algorithm	μ	# solved	avg CPU (s)	max CPU (s)	avg GAP (%)	max GAP (%)
	BCpre		**50**	809	2439	**7.5**	12.4
48	HA	2	49	*1014*	*5548*	*10.3*	*19.7*
	HA	50	**50**	**776**	**7069**	**9.8**	**21.0**
	BC		1	-	-	-	-
	BCpre		32	*2517*	*6404*	*6.4*	*8.9*

Table 3: Computational results on the two sets of 50 instances of van Zyl with T=24 and T=48 for different algorithms and parameter μ: number of instances solved (to feasibility) in 30 minutes, average and maximum computation time in seconds to reach a feasible solution, and average and maximum gap of the solution values with the optimum values (over the set of instances for which a solution has been computed: the statistics are shown in italic when this does not correspond to the whole set of instances, best results are typed in bold face).

Table 3 provides the same information as in Table 2 for algorithms *HA2*, *HA50* to estimate the performance of the splitting algorithm. We also report results on *BC* and *BCpre*, to illustrate the complexity of this problem. Precisely, the *BC* algorithm is not able to compute a feasible solution for almost all instances in the two sets T=24, T=48 in 1 and 2 hours, respectively. The efficient preprocessing of *BCPre* allows finding a feasible solution for all the smallest instances, but not for a third of the largest instances, and the computation times increase up to 40 minutes on average on the 32 solved instances. In comparison, the hybrid algorithm appears to be very stable as the discretization precision increases. Configuration HA50 computes one feasible solution for all instances in the two sets in a short time, respectively, less than 7 and 13 minutes on average. Furthermore, the computed solutions are of good quality in terms of electricity cost, as they all are within a 10% tolerance range of the optimal value.

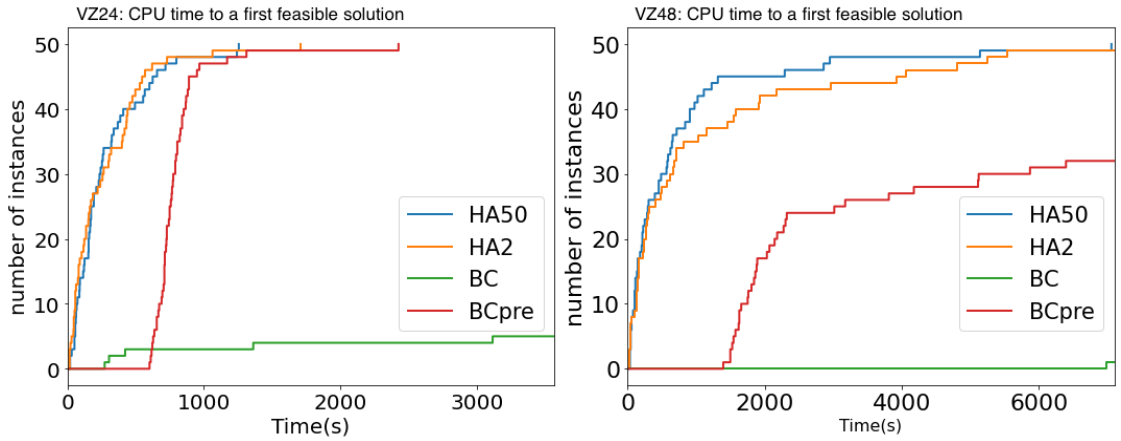

Figure 5: Cumulated number of instances solved as a function of the time needed to compute the first feasible solution by hybrid algorithm *HA* (with $\mu \in \{2, 50\}$) and by the two variants of branch-and-cut (*BC*, *BCpre*) the two sets of 50 instances of Van Zyl with T=24 (left) and T=48 (right).

Figure 5 compares the computation times cumulated over the 50 instances in the two sets T=24 and T=48. We observe that the hybrid algorithm is very fast on the majority of the instances: it takes less than 2 minutes on half of the instances in both sets. The algorithm is efficient even on the most difficult instances: for 18 of the largest instances, we were not able to close the optimality gap after running *BCpre* for several hours. The hybrid algorithm computed a feasible solution for half of these difficult instances in less than 10 minutes.

6. CONCLUSIONS

This paper presents an original hybridization of machine learning and mathematical programming to compute strictly-feasible nearly-optimal solutions to a reliable model of pump scheduling in drinking water distribution networks. Experiments on the benchmark network van Zyl show that the method manages to compute good solutions on all instances and that the computational times remain unusually low as the problem complexity increases dramatically when the scheduling horizon becomes more precise. This performance mainly results

from the deep problem decomposition offered by our splitting method. In return, the nature of the problem makes it unlikely to guarantee the theoretical convergence of iterative methods based on such a decomposition. In future work, we aim to evaluate the practical limits of the proposed approach by exploring other networks and investigate which algorithmic components need to be adapted to push these limits. In particular, we are interested in studying various methods to enforce the splitting scheme, using dualization or penalization. More generally, we are interested in applying our approach to other planning problems with storage conservation constraints, especially for the operational and strategic management of water systems with reservoirs.

ACKNOWLEDGEMENTS

This work is supported by the French government, through the 3IA Côte d'Azur Investments in the Future project ANR-19-P3IA-0002 managed by the National Research Agency (ANR).

REFERENCES

[1] A. M. Gleixner, H. Held, W. Huang, and S. Vigerske, "Towards globally optimal operation of water supply networks," Numer. Algebra Control Optim., vol. 2, no. 4, pp. 695–711, 2012.

[2] A. U. Raghunathan, "Global optimization of nonlinear network design," *SIAM J. Optim.,* vol. 23, nº 1, pp. 268–295, 2013.

[3] G. Bonvin, S. Demassey and A. Lodi, "Pump scheduling in drinking water distribution networks with an LP/NLP-based B&B," *Optim. & Engin.,* vol. 22, p. 1275–1313, 2021.

[4] B. Tasseff, R. Bent, C. Coffrin, C. Barrows, D. Sigler, J. Sticke, A. S. Zamzam, Y. Liu and P. V. Hentenryck, "Polyhedral relaxations for optimal pump scheduling of potable water distribution networks," *INFORMS J. Comput.,* vol. 36, nº 4, pp. 1040-1063, 2024.

[5] G. Bonvin, S. Demassey, C. Le Pape, N. Maïzi, V. Mazauric, and A. Samperio. "A convex mathematical program for pump scheduling in a class of branched water networks," Applied Energy 185 (2017): 1702-1711.

[6] G. Mackle, "Application of genetic algorithms to pump scheduling for water supply," in *Int. Conf. Genet. Algorithms Eng. Syst. Innov. Appl.,* 1995.

[7] J. Van Zyl, D. Savic and G. Walters, "Operational optimization of water distribution systems using a hybrid genetic algorithm," *J. Water Resour. Plan. Manag. ,* vol. 130, nº 2, pp. 160-170, 2004.

[8] R. Menke, E. Abraham, P. Parpas, and I. Stoianov, "Exploring optimal pump scheduling in water distribution networks with branch and bound methods," *Water Res. Mgmnt.,* vol. 30, no. 14, pp. 5333–5349, 2016.

[9] B. Ulanicki, J. Kahler and H. See, "Dynamic optimization approach for solving an optimal scheduling problem in water distribution systems," *J. Water Resour. Plan. Manag. ,* vol. 133, nº 1, pp. 23-32, 2007.

[10] B. Ghaddar, J. Naoum-Sawaya, A. Kishimoto, N. Taheri and B. Eck, "A Lagrangian decomposition approach for the pump scheduling problem in water networks," *Eur. J. Oper. Res.*, vol. 241, nº 2, p. 490–501 , 2015.

[11] A. J. Ulusoy and I. Stoianov, "Distributed solution of the day-ahead pump and valve scheduling problem for dynamically adaptive water distribution networks with storage," *Eur. J. Oper. Res.*, vol. 323, nº 1, pp. 267-275, 2025.

[12] E. Todini and S. Pilati, "A gradient algorithm for the analysis of pipe networks," in *Computer Applications in Water Supply*, Research Studies Press Ltd., 1988, pp. 1-20.

[13] G. Bonvin. and S. Demassey, "Extended linear formulation of the pump scheduling problem in water distribution networks," em *9th Int. Network Optim. Conf.*, 2019.

[14] S. Demassey, V. Sessa, and A. Tavakoli "Alternating direction method and deep learning for discrete control with storage," in *International Symposium on Combinatorial Optimization*, Cham: Springer Nature Switzerland, 2024, pp. 85-96.

[15] M. Collins, L. Cooper, R. Helgason, J. Kennington and L. LeBlanc, "Solving the pipe network analysis problem using optimization techniques," *Manag. Sci.*, vol. 24, nº 7, pp. 747-760, 1978.

[16] R. Rockafellar, Network Flows and Monotropic Optimization, Athena Scientific, 1999.

[17] L. A. Rossman et al., "Epanet 2: users manual," 2000.

[18] Y. Bengio, A. Lodi and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. of Op. Res.*, nº 2, p. 405–421, 2021.

[19] A. Borovykh, S. Bohte and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," arXiv:1703.04691v5, 2018.

[20] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[21] R. Nellikkath and S. Chatzivasileiadis, "Physics-informed neural networks for AC optimal power flow," *Electric Power Systems Research*, vol. 212, p. 108412, 2022.

[22] A. Shroyer and D. M. Swany, "Data Augmentation for Code Analysis," em *International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, 2022.

[23] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *33rd Int. Conf. Machine Learning*, 2016.

[24] J. Gorski, F. Pfeuffer and K. Klamroth, "Biconvex sets and optimization with biconvex functions: a survey and extensions," *Math. Meth. Oper. Res.* , vol. 66, nº 2007, p. 373–407, 2007.

[25] S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, nº 1, pp. 1-122, 2011.

[26] T. Kleinert and M. Schmidt, "Computing feasible points of bilevel problems with a penalty alternating direction method," *INFORMS J. on Computing*, vol. 33, nº 1, pp. 1-418, 2020.

[27] A. Tavakoli, "Hybrid combinatorial optimization and machine learning algorithms for energy-efficient water networks," http://www.theses.fr/2023COAZ4121, PhD thesis, 2023.

[28] A. Tavakoli, S. Demassey and V. Sessa, "Strengthening mathematical models for pump scheduling in water distribution," in *14th Int. Conf. Applied Energy*, 2022.