Alternating direction method and deep learning for discrete control with storage

Sophie Demassey, Valentina Sessa, Amirhossein Tavakoli

Alternating direction method and deep learning for discrete control with storage

Sophie Demassey[1][0000−0002−4272−6278], Valentina Sessa[1][0000−0002−0083−2515], and Amirhossein Tavakoli[1,2][0000−0002−5187−7844]

[1] Centre for Applied Mathematics, Mines Paris-PSL, Sophia-Antipolis, France
[2] University Côte d'Azur, Sophia-Antipolis, France
{sophie.demassey,valentina.sessa,amirhossein.tavakoli}@minesparis.psl.eu

Abstract. This paper deals with scheduling the operations in systems with storage modeled as a mixed integer nonlinear program (MINLP). Due to time interdependency induced by storage, discrete control, and nonlinear operational conditions, computing even a feasible solution may require an unaffordable computational burden. We exploit a property common to a broad class of these problems to devise a decomposition algorithm related to alternating direction methods, which progressively adjusts the operations to the storage state profile. We also design a deep learning model to predict the continuous storage states to start the algorithm instead of the discrete decisions, as commonly done in the literature. This enables search diversification through a multi-start mechanism and prediction using scaling in the absence of a training set. Numerical experiments on the pump scheduling problem in water networks show the effectiveness of this hybrid learning/decomposition algorithm in computing near-optimal strict-feasible solutions in more reasonable times than other approaches.

Keywords: Mixed Integer Nonlinear Programming · Variable splitting · Deep Learning.

1 Introduction

Operating any system governed by nonlinear laws and discrete controls leads to complex optimization problems. If memory or storage capacities are present, load shifting allows for a further reduction of operating costs. However, the problem of planning on a timeline the static operations interrelated by the storage balance and capacity limits is not only challenging to optimize but even to satisfy.

Consider the operation of a pressurized drinking water distribution network: the problem is to plan the activation of the pumps on a discrete time horizon to realize, at minimum cost, a hydraulic head/flow equilibrium meeting the demand in water on each time step. Elevated water tanks enable shifting the pumping ahead of time, resulting in substantial savings in energy consumption and cost, given the nonlinear efficiency of the pumps and a dynamic incentive electricity tariff. Nonconvex MINLPs modeling this problem are hard to optimize and even

to satisfy, as the feasible set is usually sparse and scarce in the binary search space, especially when the tank capacities are tight. Dedicated exact or heuristic algorithms attempt in various ways to simplify the hydraulic relations [14, 2] or the storage constraints [19, 8, 1], by trading off accuracy for speed. Still, even for small-size problems, they struggle to reach feasible solutions, and little seeks to fully repair the approximate solutions. The latter is mainly addressed by flipping the binary variables within a local [15, 1] or global search [8, 2]. Also, most existing approaches rely on fast simulation, using Newton methods, to compute the unique possible static hydraulic equilibrium at a time step when both the status of the pumps and the level of the tanks are known [18].

This property – fast computation of the static operations given a storage state and no condition on the resulting state – is not specific to hydraulic systems. In particular, it arises alike in other potential-flow networks in various contexts, ranging from electric circuits or thermodynamic systems to traffic congestion [16, p.350]. It also holds in long-term planning problems split into a sequence of smaller periods, where the planning subproblems become independent and easy once the initial states are fixed, and the final state conditions dropped. This paper focuses on computing feasible solutions for discrete control of systems with storage satisfying this property by combining two independent but complementary models: a data model built from deep learning to predict multiple approximate solutions, followed by a feasibility recovery phase based on the storage/control variable splitting in the MINLP formulation. The whole approach revolves around storage state profiles as partial solutions: it searches through these partial solutions to gradually match a feasible control decision.

It then differs from previous applications of machine learning to optimize combinatorial problems like TSP [13] or to predict partial discrete solutions of MILPs [5] where repairing/completing a feasible full assignment is less of an issue. Working in the continuous space of the storage state variables instead of the discrete control variables has advantages. First, our data model is regression and not classification. In particular, we combined a convolutional neural network (CNN) and bidirectional long short-term memory (Bi-LSTM) to capture local patterns in the input (loads and costs) and output (storage states) time series. Furthermore, we employ Monte Carlo dropout [6] to obtain multiple predictions. Working with continuous variables allows for smoother moves (both in the neighborhood exploration and in the multi-start mechanism) to address the feasibility recovery issue. It also enables us to develop a scaling mechanism as a substitute for missing data: if no training dataset is available for a given system, we propose to build it by solving, for a sufficient number of inputs, a tractable variant of the problem with a coarser temporal resolution. Once trained, the data model predicts coarse-grained storage state profiles, which are easily interpolated into fine-grained profiles, possibly without much loss in the prediction accuracy.

In the feasibility recovery phase, continuity allows for smoother moves in the search space, and the property above offers a mechanism to map storage solutions to control decisions. To leverage this property, we investigate an alternating direction method (ADM), restricting and solving the MINLP in these

two spaces alternatively. Precisely, we penalize the time-coupling storage balance constraints, then iterate between the control and storage subproblems: at each iteration, the penalized MINLP is restricted to the partial solution obtained from one subproblem, hence defining the next subproblem.

As we make no assumption on the analytic nature of the coupling between storage and control in the static subproblems, our algorithm differs from other ADM schemes. In particular, it differs from ADMM [20] and PADM [12], as we choose not to dualize this coupling. These algorithms have known theoretical convergence guarantees, even in the nonconvex case, but only if the coupling is linear, which is not the case in pump scheduling, for example. Instead, we can easily enforce this structural constraint in the control subproblem thanks to the property, but then we need to relax it in the storage subproblem. In doing so, we may lose some mild theoretical convergence guarantees [10], but we expect a fast and practical convergence to feasible solutions without much altering the cost. Besides, we enforce search diversification by multi-start, using the multiple data model predictions as trial initial points.

We illustrate and experiment with our approach to the pump scheduling problem and build an extended test set based on the benchmark water network *Van Zyl* [19]. Both deep learning and ADM are original approaches for this well-studied problem. While independent, they appear to be strongly complementary: ADM is able to recover feasible decisions with a reasonable optimality gap from predictions of the data model, which, conversely, provides good approximate solutions to initialize ADM.

2 MINLP for discrete control with storage

The optimal control problem is modeled as a sequence of steady states over a discretized time horizon $\mathcal{T} = \{0, 1, ..., T-1\}$ as follows:

$$(P): \min_{x,y,s} \sum_{t\in\mathcal{T}} f_t(x_t, y_t, s_t, C_t) \tag{1a}$$

$$\text{s.t.: } g_t(x_t, y_t, s_t, L_t) = 0 \qquad \forall t \in \mathcal{T} \tag{1b}$$

$$s_{t+1} = s_t + y_t^I \qquad \forall t \in \mathcal{T} \tag{1c}$$

$$s_t \in [\underline{S}_t, \overline{S}_t] \subseteq \mathbb{R}^I \qquad \forall t \in \mathcal{T} \cup T \tag{1d}$$

$$x_t \in \mathcal{X}_t \subseteq \{0,1\}^N, y_t \in \mathbb{R}^M \qquad \forall t \in \mathcal{T}. \tag{1e}$$

Vector $s_t \in \mathbb{R}^I$ denotes the state variables figuring the state/level of the I storage devices at $t \in \mathcal{T}$ or at the end of the horizon $t = T$. Vector x_t represents N discrete control variables at $t \in \mathcal{T}$. W.l.o.g. we assume they are binary variables figuring the on/off status of N controllers, and $\mathcal{X}_t$ represents the allowed combinations. Vector $y_t \in \mathbb{R}^M$ includes all other continuous (control or implied) variables. Together with variables x_t under Constraints (1b), they model the steady operation of the system on period $t \in \mathcal{T}$ to serve a given load $L_t \in \mathbb{R}^J$ starting from the storage state s_t. Note that y_t includes, as a subvector

denoted $y_t^I \in \mathbb{R}^I$, the positive or negative contribution of the system operation to every storage at $t \in \mathcal{T}$, as modeled by Constraints (1c). Constraints (1d) define the minimum $\underline{S}$ and maximum $\overline{S}$ storage capacities. These limits are allowed to vary in time, and $\underline{S}_0 = \overline{S}_0$, i.e., the storage state is fixed at $t = 0$. Finally, the objective (1a) is to minimize the global operation cost, where $C_t \in \mathbb{R}^\ell$ denotes exogenous price signals for the different elements operated at $t \in \mathcal{T}$. We make no assumption on the analytic nature of the real-valued functions in objective f_t and in constraints g_t, but the following property must hold.

Property 1. For all time $t \in \mathcal{T}$, solving or optimizing over $\{(x, y) \in \mathcal{X}_t \times \mathbb{R}^M : g_t(x, y, S, L_t) = 0\}$ is computationally easy, given any initial state $S \in \mathbb{R}^I$.

With the above property, we assume the subproblems can be solved quickly enough to iterate on them.

3 A concrete model: pump scheduling

The standard nonconvex MINLP for the pump scheduling problem in water distribution networks without maintenance constraints [2] is a concrete implementation of (P) through the following identification:

$$f_t(x_t, y_t, s_t, C_t) \equiv C_t^0 x_t + C_t^1 y_t^Q \text{ (pumping electric cost)},$$

$$g_t(x_t, y_t, s_t, L_t) \equiv \begin{cases} E_I y_t^Q - y_t^I & \text{(tank inflow)} \\ E_J y_t^Q - L_t & \text{(load satisfaction)} \\ s_t - B y_t^H & \text{(tank head)} \\ (E^\top y_t^H + \phi(y_t^Q)) x_t & \text{(head loss)} \end{cases}$$

where variables x represent the activity of the network arcs (i.e., pumps and pipes with or without valves), y^Q the flow through the arcs, y^H the pressure (hydraulic head) at the network nodes (i.e., tanks and service nodes), and s the tank levels. Given E the network incidence matrix, the operational constraints g_t include, in order: flow conservation at tanks, then at service nodes, tank volume/head linear relation B, nonlinear relations ϕ between head loss and flow on each arc.

For fixed $s_t = S \in \mathbb{R}^I$ and $x_t = X \in \{0,1\}^N$, the system of equations $g_t(X, y, S, L_t) = 0$, known as the *network analysis problem*, admits at most one solution y, easy to compute with Newton methods [18]. Furthermore, it breaks into independent subsystems following the network partition along the tanks [1]. Usually, each subnetwork (resp. subsystem) $p \in P$ contains a number N_p of controllable arcs (resp. binary variables), small enough to envisage to enumerate all possible combinations $X \in \{0,1\}^{N_p}$ and compute the unique corresponding solution denoted $y^p(X, S)$. The solutions of system $g_t(X, y, S, L_t) = 0$ are thus any product of such vectors $(X_p, y^p(X_p, S))$ over $p \in P$ with $X_p \in \{0,1\}^{N_p}$. Hence, Property 1 holds for pump scheduling.

Finally, note that the system $g_t(x_t, y_t, s_t, L_t) = 0$ defined above in the context of hydraulic networks, is an instance of the *nonlinear network equilibrium*

problem [16, p.350] arising in many contexts, ranging from thermodynamic systems to traffic congestion. This provides insight into the broad applicability of model (P) with Property 1.

4 ADM for local optimization by decomposition

Model (P) admits a time-decomposition after dualizing or penalizing the storage balance constraints (1c) as in the following model, given ℓ_1 penalty and multiplier $\rho_{ti} \geq 0$ for each $d_{ti}(y,s) = |s_{(t+1)i} - (s_{ti} + y^I_{ti})|$ at storage i and time t:

$$(L_\rho): \ \min_{x,y,s}\{l(x,y,s,\rho) : (1b),(1d),(1e)\}, \ \text{with}$$
$$l(x,y,s,\rho) = \sum_{t\in\mathcal{T}} f_t(x_t,y_t,s_t,C_t) + \sum_{i=1}^{I} \rho_{ti} d_{ti}(y,s).$$

Even if separable as T independent subproblems, such relaxations remain difficult to solve because of the complexity of each system (1b). However, they become tractable for fixed values of s according to Property 1.

We thus propose solving approximately (L_ρ) following the storage/control variable split: we first solve (L_ρ) with respect to the control variables (x,y), given a storage state profile s, then solve (L_ρ) with respect to the state variables s for fixed (x,y), and iterate, using the solution of one subproblem to define the restriction in the next subproblem. Contrarily to other alternating direction methods, like ADMM or PADM, we do not dualize the variable-coupling constraints (1b) but keep them as structural constraints in the control subproblems to leverage Property 1, and relax them from the storage subproblems to ensure feasibility. As a consequence, the algorithm loses the mild guarantee of [10] to converge to a stationary point, not even a local minimum, of the relaxed problem. Still, it generates a sequence of solutions closer to being feasible for (P).

Algorithm 1 Partial storage/control splitting for (P)

1: choose storage profile $s^0 \in \mathbb{R}^{TI}$, penalty $\rho \in \mathbb{R}^{TI}_+$ and set $k = 0$
2: given (s^k, ρ), solve $(x^{k+1}, y^{k+1}) \in \arg\min_{(x,y)}\{l(x,y,s^k,\rho) : (1b),(1e)\}$
3: given (x^{k+1}, y^{k+1}, ρ), solve $s^{k+1} \in \arg\min_s\{l(x^{k+1}, y^{k+1}, s, \rho) : (1d)\}$
4: If $\|d(y^{k+1}, s^{k+1})\|_\infty < \varepsilon$ then $(x^{k+1}, y^{k+1}, s^{k+1})$ is ε-feasible for (P), then STOP. Otherwise, if $\|s^{k+1} - s^k\|_\infty < \varepsilon'$, then update ρ. Increment k and go to Step 2.

Conceptually, Algorithm 1 searches through the s-space of the storage state profiles satisfying the capacity limits and attempts to derive a matching control profile (x,y) by gradually reconciling the storage states $s_t + y^I_t$ and s_{t+1} at each time t. Following the framework for PADM [7], the penalty ρ is updated when the algorithm stalls or after a given number of iterations.

The efficiency of this algorithm is highly dependent on the initial storage state profile s^0. We design a deep learning model to predict near-feasible near-optimal profiles. This model is capable of deriving multiple predictions, allowing us to

add diversification to the local optimization algorithm: starting from various initial points s^0 increases the chance of reaching a feasible solution.

5 The deep learning model

Our deep learning algorithm aims at building a hypothesis function $\mathcal{H}$ mapping to each input $(L, C) \in \mathbb{R}^{T(j+\ell)}$ (loads and costs), the storage profile $s(L, C) \in \mathbb{R}^{TI}$ in an optimal solution of problem (P) with input (L, C). The map is built, given a precomputed collection $\{((L_l, C_l), s_l)\}_{l=1}^{N_D}$ of input/target tuples, by approximately minimizing a loss function $\mathcal{L}_{loss}(\mathcal{H}(L, C), s(L, C))$, which is the mean square error in our case, as commonly used in regression problems.

To compute $\mathcal{H}$, we choose to combine a Convolutional Neural Network (CNNs) and a Bidirectional Long Short-Term Memory Network (Bi-LSTM) with dropout layers, as found in several works, e.g., in [3]. The capability of Bi-LSTM lies in its ability to handle information in both forward and backward temporal directions. This attribute is crucial in our application due to the dynamic relations spanning the entire planning horizon. On the other hand, our CNN comprises convolutional (conv1D) layers of different sizes, located in parallel to each other with zero padding, and with their outputs exclusively linked to neighboring areas within the input (i.e., loads and costs). This arrangement is accomplished by moving a filter (i.e., a weight matrix) along the input vectors, and this design enables the model to acquire filters to discover distinct patterns within the dataset [9]. To handle the temporal dimension of the target, the output of the CNN layers, implied as extracted features, are passed through a hidden layer with ReLU activation functions, and their outputs are concatenated, reshaped, and fed into the Bi-LSTM. The final prediction $\mathcal{H}(L, C)$ results from a fully connected layer with a linear activation function placed after the Bi-LSTM unit.

Finally, we implement the Monte Carlo dropout technique [17] to force our model to make multiple predictions. While this technique has been proposed to estimate the prediction uncertainty by randomly masking neurons in the neural network, we employ it for a diversification purpose. It is particularly suitable in our context, given the epistemic uncertainty due to the model complexity and the relatively small size of the training dataset. The dropout layers are placed just before and after the Bi-LSTM unit, and, in alignment with [6], we apply dropout (i.e., random neuron masking) not only during training but also during the testing phase. As a result, for a given input (L, C), the dropouts yield as many distinct solution samples $\mathcal{H}(L, C)$.

6 Experimental evaluation

We run experiments in the context of pump scheduling on *van Zyl*, a benchmark hydraulic network. We first evaluate the predictive accuracy of our deep learning model (denoted CNN-LSTM below) and compare it to a conventional feedforward neural network (FFW) having a similar number of parameters. We then evaluate the performance of our hybrid algorithm (HA) and compare it

to the open-source branch-and-check global optimizer (BC) from [2] dedicated to pump scheduling, and the enhanced variant from [4] (BCpre). BC solves the MILP relaxation of (P) with an LP-Branch-and-Bound combined with hydraulic simulation to check the actual feasibility and cost of the integer solutions. BCpre implements a heavy preprocessing to tighten the MILP formulation before running BC.

All algorithms are implemented in Python and experiments are executed on an Intel(R) Xeon(R) 6148 2.40GHz and 128 GB memory. The deep learning models CNN-LSTM and FFW are built using Tensorflow API v2.12.0 on Jupyter notebook in Google Colab with GPU A100, and the Adam optimizer [11] with the default initial learning rate of 10^{-3}. Algorithms BC and BCpre are implemented on top of the MILP solver Gurobi v10.0.1.

6.1 Benchmark, dataset generation, and scaling

The *Van Zyl* network [19] is characterized by $I = 2$ water tanks of different capacities, $J = 2$ service nodes with individual loads, and $N = 4$ controllable elements (2 symmetrical pumps and a boost pump operating parallel to a check valve). This is a medium-sized but difficult network, often used for empirical evaluations. However, the available test set is small.

We present a new dataset for implementing deep learning algorithms. It consists of a collection of 2113 unique daily observations (L, C, S), each given as $T = 12$ time steps profiles: the input features are the water demand and the electricity tariff profiles $(L, C) \in \mathbb{R}^{T(J+1)}$, and the target is the storage state profile $S \in \mathbb{R}^{TK}$ in the optimal solution of (P) for input (L, C) obtained with algorithm BCpre. Tariffs C are taken from the Belgian spot market data, considering a reference period from 2007 to 2013. Loads L are drawn from a 3-year highly seasonal history of actual consumption sourced from a network based in a touristic zone in Brittany, France. The data is split into 75 percent training, 15 percent validation, and 10 percent evaluation. Given $T = 12$, pumps and valves are scheduled on 2-hour periods, aligned with the resolution of the load and cost forecasts. Because of the nonlinear behavior and the limited capacity of the tanks, it is advantageous to enforce a finer-grained schedule on periods of 1 hour ($T = 24$) or 1/2 hour ($T = 48$). However, when the time horizon length increases, it quickly becomes difficult to compute even feasible solutions in a reasonable time. This prevents the creation of a dataset of reasonable size to train our data model on daily instances for $T = 24$ or $T = 48$. To apply our algorithm to such instances, we then use the data model trained for $T = 12$ and scale down and up the input and output time series, using linear interpolation, to get the predictions for starting ADM. We expect that such a simple transformation does not deteriorate the quality (feasibility and optimality) of the predicted storage state profiles. We generate the two benchmark sets of 50 instances for $T = 24$ and $T = 48$ also by resampling instances with coarser-time discretization $T = 12$ and high variability. The three benchmark sets (denoted VZ12, VZ24,

and VZ48), together with the training set and the code of the algorithms, are available online[3].

6.2 Hyperparameters of the deep learning models

To train CNN-LSTM, we set the batch size to 32, and the maximum number of epochs to 1350. For all conv1D and feedforward layers, we use l_2 regularization with a coefficient equal to 0.02. The conv1D layers have 32 filters, and their kernel sizes range from 4 to 10. The number of hidden layers of Bi-LSTM is set to 16. Due to the complexity of the model and the relatively small training set, and for more diversification, we select a high dropout rate of 0.75, then generate 80 Monte Carlo dropout samples.

For comparison, we design a FFW network with 8 hidden layers, $ReLU$ activation functions, and a pyramid architecture. We set the maximum number of epochs to 250, and the regularization coefficient to 0.01. We also add dropout layers with a rate between 0.2 and 0.5 after each hidden layer to mitigate overfitting and generate multiple starting points for ADM. In the end, CNN-LSTM and FFW have 45k and 63k parameters, respectively.

6.3 Parameters of the optimization algorithms

We run Algorithm 1 sequentially, although it could be parallelized, from at most 35 trial points s^0: the first trial is the component-wise mean prediction, other trials are just picked randomly from the 80 predictions samples. Both ADM and BC algorithms are stopped when reaching a first feasible solution, given the feasibility tolerance $\varepsilon = 10^{-6}$, or a global time limit.

All other parameter values were chosen after a brief numerical evaluation process. In Algorithm 1, we set $\varepsilon' = 10^{-3}$. For a given trial s^0, the penalty vector ρ is updated at most four times, and the number of iterations k is limited to 85 between each update. At each update, penalties are increased according to the update number $u \in \{0, \dots, 4\}$ and to the constraint violations. Precisely, for each time $t \in \mathcal{T}$ and storage $i \in \{1, \dots, I\}$, we update ρ_{ti} to $1 + 5a$ if $d_{ti}(y^{k+1}, s^{k+1}) > \varepsilon$, and to $1 + 2a$ otherwise, given a random value $\xi \in [0.75, 1]$ and $a = \xi e^{(\frac{-u}{10})} \rho_{ti}$. We initialize the penalty terms ρ_{ti}, uniformly, either to value 2 or to value 50, corresponding to algorithms denoted HA2 and HA50 below.

6.4 Performance of deep learning

Prediction error. To assess the effectiveness of the CNN-LSTM, we first compute the prediction error between the outcome $\mathcal{H}(L, C)$ (the component-wise mean of the 80 prediction samples) and the target $s(L, C)$ over the test set, using the classical metrics for time series in regression problems. With CNN-LSTM, the Mean Square Error between predictions and targets is MSE= 0.64, and the Mean Absolute Error is MAE= 0.53 or nMAE= 9.2% after normalization (divided by

[3] https://github.com/sofdem/gopslpnlpbb

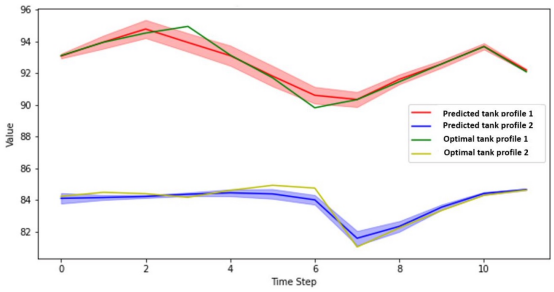

Fig. 1. Storage state profiles: prediction, credible interval, and optimal target.

the range of target values). The Pearson coefficient, which measures the linear correlation between predictions and targets, is $R = 0.772$. In comparison, FFW yields MSE= 1.56, MAE= 0.89, nMAE= 12.7%, and $R = 0.724$. These metrics show that CNN-LSTM performs well and better than FFW. Figure 1 depicts the storage state profiles in each tank for one random instance in the test set, showing the proximity of the optimal target and the CNN-LSTM prediction with its uncertainty interval.

Distance to a feasible solution. Since the optimal solutions of (P) are not unique, these metrics only show the ability of the deep learning model to predict the target one. We now estimate the distance of the predictions to feasible solutions by measuring the number of predictions that actually lead to feasible solutions through ADM (with penalty terms ρ initialized to 50). When starting from CNN-LSTM predictions, ADM finds a feasible solution for 49 out of the 50 instances in benchmark VZ12, but only half (27 instances out of 50) from FFW predictions. For the 26 instances where both algorithms reach a feasible solution, the associated objective function value derived with CNN-LSTM is smaller for all but two instances, with an average 4.8% decrease.

6.5 Performance of the hybrid algorithm

The previous experiment provides a first insight into the efficiency of algorithm HA. It hints that better predictions increase the chance of ending up with a feasible solution or a better upper bound. We now validate the capability of HA to compute good feasible solutions for the daily instances of the Van Zyl network with $T = 12, 24, 48$. We evaluate the hybrid algorithm with the penalty terms ρ in ADM initialized to 50 (HA50) or to 2 (HA2), and we compare it with the two variants of branch-and-check, with (BCpre) and without (BC) advanced preprocessing. Each algorithm stops when reaching one first feasible solution or the CPU time limit: 1800 seconds for $T = 12$, 3600 seconds for $T = 24$, and 7200 seconds for $T = 48$. Table 1 reports, for each benchmark set (column 1) and for each algorithm (column 2), the number of instances solved (column 3) among the 50 instances in the set, then, on the subset of instances solved by

Table 1. Instances solved, computation time, and optimality gap.

bench	algo	#solved	avg time	max time	avg gap	max gap
VZ12	HA50	*49*	*254s*	*1570s*	*6.6%*	*21.2%*
	HA2	*44*	*305s*	*1577s*	*4.6%*	*11.3%*
	BC	*48*	*121s*	*681s*	*5.4%*	*12.5%*
	BCpre	**50**	124s	137s	4.3%	12.4%
VZ24	HA50	**50**	285s	1257s	9.5%	23.4%
	HA2	**50**	279s	1711s	8.4%	16.3%
	BC	*5*	*1097s*	*3117s*	*11.1%*	*12.6%*
	BCpre	**50**	809s	2430s	7.5%	39.6%
VZ48	HA50	**50**	776s	7069s	9.8%	21.0%
	HA2	*49*	*1014s*	*5548s*	*10.3%*	*19.7%*
	BC	*1*	-	-	-	-
	BCpre	*32*	*2517s*	*6404s*	*6.4%*	*8.9%*

the algorithm: the average (column 4) and maximum (column 5) computational time in seconds, and the average (column 6) and maximum (column 7) estimated optimality gap, measured as the deviation in % of the solution cost to the best known lower bound computed with BCpre.

The results of the global solver BC illustrate well the computational complexity of this problem, particularly as the horizon length T increases. With a tailored preprocessing (running for a minimum of 100 seconds included in the time reported in the table), BCpre quickly reaches one first feasible solution of high quality for all instances with $T = 12$. The performance falls dramatically for the largest instances with $T = 24$ and $T = 48$. Without preprocessing, BC cannot even compute one feasible solution for most of the instances within the allowed time. Preprocessing strongly improves the performance, still, BCpre cannot reach feasible solutions in two hours for 18 instances with $T = 48$.

While the hybrid algorithm is globally outperformed on the instances with $T = 12$, it is well suited to the hardest instances with $T = 24$ and $T = 48$, as HA50 can compute a feasible solution for all these instances, within an average 10% estimated optimality gap (which is overestimated as the lower bound computed by BCpre is probably far below the optimum value). Because of the scaling mechanism, the prediction error is probably worse for these instances, compared to $T = 12$ where the deep learning predictions are directly used as starting points for ADM. However, the space of the feasible solutions is also less scarce, and the optimal storage state profiles for the coarse and fine discretization problems are probably not too dissimilar. This should explain the good performance of the hybrid algorithm on the hardest benchmark sets VZ24 and VZ48 and how this approach overcomes the difficulty of solving larger dimensional problems as the scheduling horizon increases.

Using a lower penalization of the time coupling constraints, HA2 is less robust than HA50, as it solves fewer instances, but the computed feasible solutions have a slightly better cost, at least when $T = 12$ and $T = 24$. On average, for these instances, the first feasible solutions computed by BCpre are still better, but

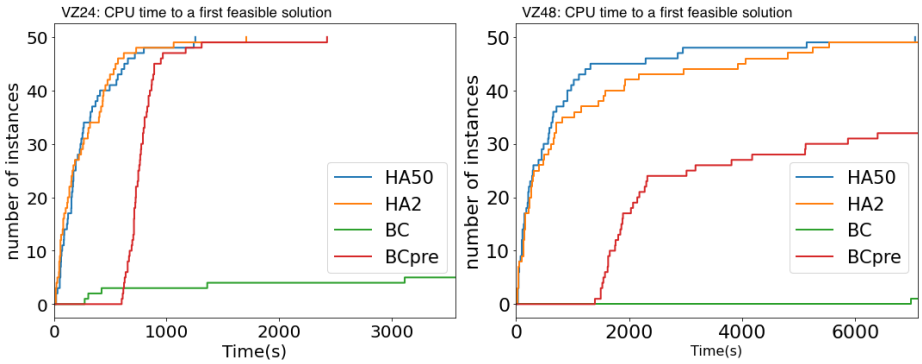

Fig. 2. Number of instances solved to feasibility over time on benchmarks VZ24 (left) and VZ48 (right): hybrid algorithm (HA2, HA50) vs branch-and-check (BC, BCpre).

the average computational time is also about three times larger. For $T = 48$, HA50 finds feasible solutions in 13 minutes on average for all instances and in 8 minutes on the 18 instances for which BCpre does not acquire any solution in 2 hours.

Figure 2 depicts the number of instances solved in VZ24 and VZ48 as a function of time for each algorithm. It clearly illustrates the speed of the hybrid algorithm.

7 Conclusions

The proposed strategy is to address a certain broad class of control problems through the mapping between the discrete control decisions and the continuous storage state profiles. Searching in this continuous space allowed us to derive an original combination of deep learning and variable splitting, linked with scaling and multi-start mechanisms to overcome the lack of training data and the prediction errors, respectively. The approach proves to be effective in tackling the feasibility issue in hard combinatorial and nonconvex problems, such as pump scheduling. A major asset is its robustness to the lengthening of the scheduling horizon, which results from the storage/control decomposition strategy. A perspective of this work is to investigate different ways to drive the search, e.g., to learn also dual starting points (penalty terms) for ADM or to apply numerical algorithms (black-box, nonsmooth) to the augmented Lagrangian relaxation that could exploit the same decomposition and mapping.

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

References

1. Bonvin, G., Demassey, S.: Extended linear formulation of the pump scheduling problem in water distribution networks. In: 9th Int. Network Optim. Conf. pp. 13–18 (2019)
2. Bonvin, G., Demassey, S., Lodi, A.: Pump scheduling in drinking water distribution networks with an LP/NLP-based B&B. Optim. & Engin. **22**, 1275–1313 (2021)
3. Borovykh, A., Bohte, S., Oosterlee, C.W.: Conditional time series forecasting with convolutional neural networks. preprint arXiv:1703.04691 (2017)
4. Demassey, S., Sessa, V., Tavakoli, A.: Strengthening mathematical models for pump scheduling in water distribution. In: 14th Int. Conf. Applied Energy (2022)
5. Ding, J.Y., Zhang, C., Shen, L., Li, S., Wang, B., Xu, Y., Song, L.: Accelerating primal solution findings for mixed integer programs based on solution prediction. In: AAAI Conf. Artif. Intell. vol. 34, pp. 1452–1459 (2020)
6. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: 33rd Int. Conf. Machine Learning. vol. 48, pp. 1050–1059 (2016)
7. Geißler, B., Morsi, A., Schewe, L., Schmidt, M.: Penalty alternating direction methods for mixed-integer optimization: A new view on feasibility pumps. SIAM J. Optim. **27**(3), 1611–1636 (2017)
8. Ghaddar, B., Naoum-Sawaya, J., Kishimoto, A., Taheri, N., Eck, B.: A Lagrangian decomposition approach for the pump scheduling problem in water networks. Eur. J. Oper. Res. **241**(2), 490–501 (2015)
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
10. Gorski, J., Pfeuffer, F., Klamroth, K.: Biconvex sets and optimization with biconvex functions: a survey and extensions. Math. Meth. Oper. Res. **66**, 373–407 (2007)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. preprint arXiv:1412.6980 (2014)
12. Kleinert, T., Schmidt, M.: Computing feasible points of bilevel problems with a penalty alternating direction method. INFORMS J. on Computing **33** (2020)
13. Kool, W., Van Hoof, H., Welling, M.: Attention, learn to solve routing problems! preprint arXiv:1803.08475 (2018)
14. Mackle, G.: Application of genetic algorithms to pump scheduling for water supply. In: Int. Conf. Genet. Algorithms Eng. Syst. Innov. Appl. pp. 400–405 (1995)
15. Naoum-Sawaya, J., Ghaddar, B., Arandia, E., Eck, B.: Simulation-Optimization Approaches for Water Pump Scheduling and Pipe Replacement Problems. Eur. J. Oper. Res. **246** (Apr 2015)
16. Rockafellar, R.T.: Network Flows and Monotropic Optimization. Athena Scientific (1999)
17. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
18. Todini, E., Pilati, S.: A gradient algorithm for the analysis of pipe networks. In: Computer Applications in Water Supply, vol. 1. Research Studies Press Ltd. (1988)
19. Van Zyl, J.E., Savic, D.A., Walters, G.A.: Operational optimization of water distribution systems using a hybrid genetic algorithm. J. Water Res. Plan. Mgmnt. **130**(2), 160–170 (2004)
20. Wang, Y., Yin, W., Zeng, J.: Global convergence of admm in nonconvex nonsmooth optimization. J. Sci. Comput. **78**, 29–63 (2019)