Hybrid combinatorial optimization and machine learning algorithms for energy efficient water networks

Amirhossein Tavakoli

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Algorithmes hybrides d'optimisation combinatoire et d'apprentissage automatique pour l'efficacité énergétique des réseaux d'eau potable

Amirhossein TAVAKOLI

Centre de mathématiques appliquées, Mines Paris-PSL

20 décembre 2023

Présentée en vue de l'obtention du grade de docteur en Contrôle, Optimisation, Prospective **d'**Université Côte d'Azur

Dirigée par : Sophie DEMASSEY, maître-assistante, Mines Paris-PSL
Co-dirigée par : Valentina SESSA, Chargée de recherche, Mines Paris-PSL
Soutenue le : 20 décembre 2023

Devant le jury, composé de :
Dominique QUADRI, Professeure, Université Paris Saclay
Ambros GLEIXNER, Professeur, HTW Berlin
Laurent VERCOUTER, Professeur, INSA Rouen Normandie
Viet Hung NGUYEN, Professeur, Université Clermont Auvergne

ALGORITHMES HYBRIDES D'OPTIMISATION COMBINATOIRE ET D'APPRENTISSAGE AUTOMATIQUE POUR L'EFFICACITÉ ÉNERGÉTIQUE DES RÉSEAUX D'EAU POTABLE

Hybrid combinatorial optimization and machine learning algorithms for energy efficient water networks

Amirhossein TAVAKOLI

Jury :

Rapporteurs

Dominique QUADRI, Professeure, Université Paris Saclay
Ambros GLEIXNER, Professeur, HTW Berlin

Examinateurs

Laurent VERCOUTER, Professeur, INSA Rouen Normandie
Viet Hung NGUYEN, Professeur, Université Clermont Auvergne

Directeur de thèse

Sophie DEMASSEY, maître-assistante, Mines Paris-PSL

Co-directeur de thèse

Valentina SESSA, Chargée de recherche, Mines Paris-PSL

Université Côte d'Azur

Résumé

Les réseaux de distribution d'eau potable sont des systèmes énergivores, en raison principalement du pompage. Cependant, ils offrent des opportunités de réduction et de report de charge, grâce aux châteaux d'eau et à leur capacité de stockage. La gestion opérationnelle optimisée du pompage et du stockage dans les réseaux d'eau, dite aussi « ordonnancement du pompage », est donc est un levier avantageux pour les réseaux électriques, mais c'est aussi un problème d'optimisation mathématique complexe. L'objet de cette thèse est la conception d'algorithmes de résolution efficaces pour un modèle mathématique détaillé, discret et non-convexe. Contrairement à l'essentiel de la littérature sur le sujet, l'accent est mis sur le calcul de solutions strictement réalisables, éventuellement optimales, pour le modèle mathématique. Par ailleurs, l'étude s'efforce de lever la complexité algorithmique du problème engendrée spécifiquement par les contraintes couplantes de stockage et présente différentes approches pour opérer et exploiter la décomposition temporelle et spatiale du modèle. Une première contribution porte ainsi sur la conception de techniques de prétraitement par renforcement de bornes et génération de coupes. Bornes et coupes sont obtenues par optimisation de relaxations détaillées, mais partielles (sur un horizon temporel tronqué ou une partie du réseau), et permettent de renforcer une relaxation plus simple (continue et linéaire), mais générale, sur laquelle est construit un algorithme d'optimisation globale. Une seconde contribution porte sur le développement d'un algorithme original d'optimisation locale, de type « Alternating Direction Method », consistant à raffiner progressivement un profil de stockage jusqu'à aboutir à un ordonnancement du pompage valide associé. En fixant les contraintes couplantes de stockage, à chaque itération, le modèle discret non-convexe restreint peut en effet être résolu de manière exacte, par décomposition temporelle et spatiale. L'algorithme consiste ainsi à reconstruire une solution (un plan de pompage) réalisable à partir d'une solution (un profil de stockage) approchée quasi optimale. Si de nombreuses heuristiques de la littérature peuvent être invoquées pour générer cette solution approchée initiale, nous proposons de l'obtenir en construisant un modèle de données. La troisième contribution de la thèse porte ainsi sur le développement d'un modèle d'apprentissage profond pouvant s'appuyer sur l'historique des opérations d'un réseau donné et résultant en un modèle de données complémentaire au modèle mathématique auquel il est hybridé. Une originalité de l'approche porte sur son potentiel de mise à l'échelle permettant d'apprendre une solution pour une discrétisation temporelle fine à partir d'un jeu de données pour une discrétisation temporelle grossière, et remédier ainsi à la difficulté de génération des données d'apprentissage. À noter enfin que cet algorithme hybride d'optimisation combinatoire et de machine learning trouve des applications à d'autres problèmes de contrôle optimal discret de systèmes avec stockage. L'évaluation empirique a donné

lieu à la génération de jeux de données étendus d'apprentissage et d'expérimentation sur des réseaux de la littérature et a mis en évidence la performance des algorithmes exact et approché.

Mots-clés: *optimisation mathématique, apprentissage, réseaux hydrauliques*

Abstract

Drinking water distribution networks are energy-intensive systems, mainly due to pumping. However, they offer opportunities for load reduction and shifting, thanks to water towers and their storage capacity. Optimized operational management of pumping and storage in water networks, also known as "pump scheduling", is therefore an advantageous lever for electricity networks, but it is also a complex mathematical optimization problem. The object of this thesis is the design of efficient resolution algorithms for a detailed, discrete, and non-convex mathematical model. Unlike most of the literature on the subject, the emphasis is placed on the calculation of strictly feasible, possibly optimal, solutions of the mathematical model. Furthermore, the study strives to mitigate the algorithmic complexity of the problem due specifically to the coupling storage constraints and presents different approaches to operate and exploit the temporal and spatial decomposition of the model. A first contribution thus concerns the design of preprocessing techniques for bound tightening and cut generation. Bounds and cuts are obtained from detailed partial (on a truncated time horizon or a part of the network) relaxations and make it possible to reinforce a simpler (continuous and linear) general relaxation, the basis of a global optimization algorithm. A second contribution concerns the development of an original local optimization algorithm, of the "Alternating Direction Method" type, which progressively refines a storage profile until reaching the associated valid pump schedule. Indeed, by fixing the coupling storage constraints at each iteration, the restricted non-convex discrete model can be solved exactly, by temporal and spatial decomposition. The algorithm thus recovers a feasible solution (a pumping plan) from a near-feasible near-optimal solution (a storage profile). If many heuristics from the literature can be invoked to generate this initial solution, we propose to obtain it by building a data model. The third contribution of the thesis thus concerns the development of a deep learning model, relying on the history of operations of a given network, and resulting in a data model complementary to the hybridized mathematical model. Scalability is an original feature of the approach, making it possible to learn a solution with a fine temporal discretization from a dataset for a coarse temporal discretization, thus remedying the difficulty of dataset generation. Finally, note that this hybrid combinatorial optimization and machine learning algorithm applies to other discrete optimal control problems of systems with storage. The empirical evaluation went through the generation of extensive training and experimentation datasets on networks from the literature and highlighted the performance of the exact and approximate algorithms.

Keywords: *mathematical optimization, machine learning, water networks*

Acknowledgments

Bewilderingly fast my time for this journey has been over and I am still in the state of awe and wonder.

I am deeply grateful to Sophie and Valentina for their trust and for offering me this wonderful position. Words cannot express how fulfilling this job has been for me, nor can I imagine being as fortunate in the future to find a similar opportunity. Beyond their intellectual and scientific guidance, I owe them a debt of gratitude for their unbounded support and patience from the very beginning to the end of my PhD journey, Thank you!

My appreciation extends to Prof. Dominique Quadri and Prof. Ambros Gleixner for their invaluable feedback and encouraging words about my PhD manuscript. I also wish to thank Prof. Laurent Vercouter and Prof. Viet Hung Nguyen for the interesting and inspiring discussions during my dissertation.

I am thankful to 3IA Côte d'Azur for their generous financial support, which greatly facilitated our research endeavors. The intellectually vibrant environment at our institution has been nothing short of inspiring.

My heartfelt gratitude goes out to all the senior researchers at CMA, the director, Prof. Nadia Maïzi, the administration, Alice and Amel, the informatics unit, Sébastien and Damien, and my fellow PhDs and Post-Docs for creating such a splendid environment. The meetings and gatherings were invigorating throughout these years.

A special thanks to my friends, my parents, Hamid and Monireh, and my sister, Fatemeh, for their unconditional support and for being my haven during difficult times.

Contents

List of Figures

List of Tables

List of Abbreviations

ADM	Alternating Direction Method
ADMM	Alternating Direction Method of Multipliers
BB	Branch and Bound
CNN	Convolutional Neural Network
DL	Deep Learning
DWDN	Drinking Water Distribution Network
EAA	Extended Analysis Algorithm
FC	Fully Connected layer
FFW	Feed ForWard
HA	Hybrid Algorithm
KKT	Karush–Kuhn–Tucker
LSTM	Long-Short Term Memory
MAE	Mean Absolute Error
MSE	Mean Square Error
MINLP	Mixed Integer Nonlinear Programming
MIR	Mixed Integer Rounding
ML	Machine Learning
NN	Neural Network
OBBT	Optimization-based Bound Tightening Techniques
PADM	Penalty Alternating Direction Method
PWL	PieceWise Linear

Notation

$\mathcal{A}$	Arcs
$\mathcal{A}_L$	Pipes
$\mathcal{A}_K$	Pumps
$\mathcal{A}_V$	Valves
$\dot{\mathcal{A}}$	Controllable arcs ($\dot{\mathcal{A}} = \mathcal{A}_K \bigcup \mathcal{A}_V$)
$\mathcal{J}$	Nodes
$\mathcal{C}$	Reservoirs
$\mathcal{S}$	Service nodes
$\dot{\mathcal{C}}$	Tanks
$\mathcal{T}$	Discretized time horizon ($\mathcal{T} = \{0, ..., T-1\}$)
T	Number of periods ($T = \lvert\mathcal{T}\rvert$)
$\bar{\mathcal{T}}$	Extended time horizon ($\bar{\mathcal{T}} = \mathcal{T} \cup T$)
E_{aj}	Incidence matrix of a graph defined for all $a \in \mathcal{A}$ and $j \in \mathcal{J}$
q_{ta}	Flow through $a \in \mathcal{A}$ at $t \in \mathcal{T}$
$q_{tj}^{\mathcal{J}}$	Inflow at $j \in \mathcal{J}$ and $t \in \mathcal{T}$ ($q_{tj}^{\mathcal{J}} = \sum_{a \in \dot{\mathcal{A}}} E_{aj} q_{ta}$)
h_{tj}	Hydraulic head at $j \in \mathcal{J}$ and $t \in \bar{\mathcal{T}}$
H_{tj}	Hydraulic head at $j \in \mathcal{C}$ and $t \in \bar{\mathcal{T}}$
v_{ta}	Head loss through $a \in \mathcal{A}$ at $t \in \mathcal{T}$ ($v_{ta} = -\sum_{j \in \mathcal{J}} E_{aj} h_{tj}$)
D_{tj}	Forecasted water demand at $j \in \mathcal{S}$ and $t \in \mathcal{T}$
x_{ta}	Status of the controllable arcs $a \in \dot{\mathcal{A}}$ and $t \in \mathcal{T}$ ($x_{at} \in \{0, 1\}$)
C_t	Tariff at $t \in \mathcal{T}$

Chapter 1

Introduction

1.1 Mitigation of energy consumption in water supply networks

Given that water demand is expected to increase up to 50% globally by 2050 based on UN estimation, the water utilities are under great strain to supply the increasing demand. Growing population and climate change are the main factors driving the continued growth in demand. The sustainability of such infrastructure requires an optimal design and operation of the water network. Approximately 4% of all electricity consumption in OECD areas is merely devoted to water network facilities [2]. As a result, a slight improvement in the operation of water networks leads to a substantial reduction in energy consumption and cost reduction.

The primary focus of this thesis is the optimal operation of the pumps in drinking water distribution networks. The aim of this decision-making problem is to plan the pumping operations to minimize the energy bill over a future period, e.g., a day ahead. The complexity of the problem increases when considering an electricity tariff and water demands varying over the period. Dynamic tariffs are designed to flatten the global energy consumption peaks and to mitigate the induced carbon emissions. Indeed, they provide a major incentive for load shifting in flexible energy-intensive industries, i.e., they motivate the operators to shift the energy consumption to low tariff periods.

Since the demands for drinking water and electricity share the same dynamic of

the human local activity, load shifting in drinking water distribution networks means decoupling pump operation and water supply. It is made possible by the storage capacity provided by the elevated water tanks. The water tanks provide an opportunity to leverage the low tariff periods and avoid pumping during peak hours when the demand for water and the cost of electricity are both high. Therefore, planning the pumping regarding dynamic tariffs and demand forecasts with the objective of minimizing the electricity bill yields an implementation of optimal load shifting in drinking water distribution networks. As a result, it has both economic and ecological impacts.

Historically, the pumping operations were driven by the two-rates day/night electricity tariffs. In this scheme, pumps are operated at night to fill in the water tanks, but they need to be operated at a higher day rate if the tank capacities do not cover the water demand for the whole day. The optimal management relies then on a different automated strategy based on the use of trigger levels: the decision bears on the minimum and maximum levels of water in the tanks to activate and stop the pumps [3]. This strategy does not apply to more dynamic electricity tariffs, which involves renewable energy sources integrated into electricity production [4].

In a non-automated setting, human operators rely on their expertise and on-demand forecasts to operate the pumps in real-time. Their decision can be assisted or simulated in real-time, e.g., with a Reinforcement Learning approach [5], or in anticipation, given a plan of pumping over a future period, typically the day ahead. Such a plan results from solving an optimization problem named the *pump scheduling problem*. Depending on the accuracy of the model and of the forecasts, pump scheduling optimization could also be attached to a controller and integrated into a fully automated system. However, the computational complexity of an accurate representation of this optimization problem jeopardizes this strategy.

1.2 The pump scheduling problem

Namely, the pump scheduling problem is defined as an optimal control problem over a discrete-time finite horizon. The activity of the pumps, and also of the valves, must be decided at every step of the time horizon in order to satisfy the forecasted demand on each period and to minimize the pumping electricity cost over the whole horizon. The decision is discrete, as pumps and gate valves are either on or off at every time instant. The activity of a variable-speed pump or a pressure-reducing valve also depends on a continuous characteristic (the variable speed or the pressure reduction), but these types of elements are not considered in the present thesis (neither in the

major part of the literature on pump scheduling).

The transport of water in the network is assumed stationary on each period. It is represented as a flow through the network arcs (pipes, pumps, valves) induced by the potentials at the network nodes (sources, tanks, service nodes). The potential also called the *hydraulic head*, is a measure of energy. Due to frictions, a pipe provides resistance to the water passing through it and induces a *head loss* between its bounds. On the contrary, a pump (if on) converts electrical energy into hydraulic energy, inducing a *head gain* between its bounds. The relationship between the flow and the head loss/gain through an arc is usually nonlinear.

A demand profile is attached to each service node, fixing the inflow supplied to the node at each time step. The water tanks are not mere passive entities, but dynamic elements with variable stored volumes, subject to lower and upper limits. It is commonly assumed that the hydraulic head at a tank is linearly dependent on the volume stored.

Many other physical concerns can be integrated into the problem definition, like pump aging prevention, water quality, water leakage, limited sources, etc. However, the pump scheduling problem is already hard in its simplest definition.

Clearly, the dimension of the tanks plays a significant role in the difficulty of this problem: if tanks are oversized compared to the demand, they can be considered as infinite buffers and loaded at low-price periods. Still, even if realistic (which it is not), the uncapacitated variant of pump scheduling is challenging because of the nonlinear/nonconvex behavior of the pumps. The limited capacity of the tanks also interferes with the dimension of the pumps, their binary status (on/off), and the granularity of the time discretization. Indeed, the capacity of a small tank could be systematically exceeded when switching on a powerful pump for the duration of a time step. A fine-grained time resolution is then required to give enough flexibility to the schedule, but the scale of the problem increases proportionately. Finally, storage management at tanks yields temporal interdependencies, which prevent the decomposition of the problem into single-period static subproblems.

In summary, the pump scheduling problem appears as a large-scale discrete nonconvex problem intertwined through both temporal and spatial dimensions. This is further emphasized in networks with abundant physical elements (pumps, pipes, tanks, etc.), each of which brings its layer of complexity and, collectively, exacerbates the computational challenge. We described how the water tanks act as pivotal bottlenecks in the optimization schema due to their inherent capacity limits and induced temporal dependency. Yet, the central role of the tanks in the seeming complexity and underlying structure of the pump scheduling problem unveils opportunities to

help solve the problem. One purpose of this thesis is to explore, after others, these opportunities.

1.3 Mathematical challenges and opportunities

Despite recent impressive advances in mathematical optimization, simultaneous handling of nonlinearity/nonconvexity and integrality remains intractable for relatively small-scale problems. The pump scheduling problem allows a formulation as a nonconvex mixed-integer nonlinear program (MINLP). Off-the-shelf nonconvex MINLP solvers (or global optimization solvers, when exact) could tackle only toy instances [6] and quickly struggle to even find feasible solutions as the instance size increases. As a result, to realize the optimization of the pump scheduling problem, some machinations must be taken. In general, the approaches to tackling this problem can be categorized into two major classes.

In mathematical programming approaches, the computational complexity is reduced by approximating the nonlinear resistance relations (typically as linear or piecewise linear functions) or by ignoring the integrality constraints. More accurate approximations induce a higher possibility of ending up with an operable solution, but simultaneously, less efficient algorithms. An acceptable trade-off between accuracy and complexity may not always be found.

In metaheuristics approaches, the nonlinear constraints are handled accurately with a stand-alone hydraulic simulator, but the search in the discrete space of the solutions is incomplete and is unaware of the global structure of the problem. Previous works in this line also often overcome the feasibility issue by softening the tank limits, i.e., they relax and penalize the violations in the objective. Hence, the computed solutions are provided without a certificate of optimality or even feasibility.

Still, the metaheuristic approaches exploit an important characteristic of the pump scheduling problem: when the binary scheduling decisions are all fixed (i.e., the status of the pumps is known at every period), the resulting nonlinear restricted problem has at most one possible solution, which is easy to compute by applying (through a hydraulic simulator such as EPANET [7]) a Newton method on the successive periods, while updating the tank levels/heads progressively. In this framework, the pump scheduling problem is seen as a bilevel program with the binary variables at the upper level and the nonlinear constraints at the lower level.

This property has also been considered in mathematical programming approaches

(but without formulating the bilevel model explicitly): Naoum-Sawaya et al. [8] drive the search in the upper-level space with logic Benders decomposition, Costa et al. [9] evaluate all the binary schedules explicitly. Several works implement an implicit (but possibly exhaustive) search by evaluating only the schedules feasible for a given mixed-integer linear programming (MILP) relaxation tackled with either cut generation (in the manner of Outer-Approximation) [10, 11] or branch-and-bound (in the manner of Branch-and-Check) [6, 12]. These last approaches benefit from very competitive MILP solvers to navigate the discrete search space and simulation to tackle the nonlinear constraints efficiently. Still, they highly depend on the quality of the MILP relaxation: schedules that are feasible for the MILP relaxation but not for the nonconvex MINLP problem are evaluated in vain, and these approaches suffer from the scarcity and sparsity of the nonconvex feasible solutions. Moreover, optimizing over the MILP relaxation alone is often already challenging for modern MILP solvers due to its scheduling nature with many symmetries and a dense objective function. Finally, because of a lack of benchmark test sets, it is hard to compare empirically all these methods.

In light of all these works, the pump scheduling problem appears as a very challenging nonconvex MINLP problem, mainly because of its dynamic (sequence-dependent) and combinatorial nature. However, this property provides a strong opportunity both to decompose the problem and to process the nonconvex constraints. This attribute can be found in other water optimization problems, such as in the static pipe layout/design problem and in different energy infrastructural systems (e.g., for crude oil or natural gas), which share the same potential-driven flow network structure. Studies on the pump scheduling problem may thus benefit other practical problems. On another note, computing a strictly feasible solution for this MINLP is already a challenge by itself, which is often neglected in previous approaches, probably because it is less of an issue in practice. However, computing feasible solutions is important when the goal is to develop reliable MINLP algorithms, not just to address a practical problem.

Finally, learning algorithms emerged in this domain, not only to forecast the demand but also to operate the water networks by drawing on the history of the operations. A data-based model could be complementary to a mathematical model, and ways to combine them are yet to be deeply investigated for nonconvex MINLP optimization in general, and for the pump scheduling problem in particular.

1.4 Contributions and organization of the thesis

This thesis contributes to the pump scheduling problem in various aspects in response to the above observations.

The primary goal is to take advantage of the near-decomposable structure of the problem to devise or enhance optimization algorithms. We consider two algorithms, exact and heuristic, focusing on the strict feasibility of the solutions regarding the MINLP formulation. In the exact algorithm, we address the limitations of the branch-and-check approaches by improving the MILP relaxation with optimization-based bound tightening and cut generation. By "optimization-based", we mean to solve auxiliary optimization problems to derive bounds and cuts. These auxiliary problems are built to take advantage of the problem decomposition. Regarding the heuristic, we devise an original variable-splitting algorithm akin to the classical alternative direction method (ADM). In this context, we also present two deep-learning (DL) architectures tailored to the pump scheduling problem, capable of learning a subset of the optimal solution. As a natural way to combine the decomposition algorithm with a learning strategy, we propose initializing ADM with the DL approximate solutions. From a different perspective, this strategy can also be seen as the use of ADM to recover feasibility from the DL approximate solutions.

On the sidelines, we make several theoretical and practical contributions to the study of the pump scheduling problem. From the theoretical side, we notably exhibit the bilevel structure of the problem and the relation with the general concepts of nonlinear flow networks and monotropic programming [13]. From the practical side, we developed a benchmark test set to compare optimization algorithms, as well as a methodology to build datasets from benchmark networks to train learning algorithms when no historical operational data is available. All our codes are available online.

Finally, we discuss bridges between our algorithms and optimization problems beyond pump scheduling. Hence, some cutting-planes that we present are not only valid for water network distribution but also for other types of capacitated flow networks. Moreover, our variable-splitting algorithm applies to various discrete control problems of systems which includes storage.

The present document is organized as follows:

We present the pump scheduling problem in Part I. The problem is defined and formulated as a MINLP in Chapter 2. We also exhibit the bilevel formulation and

relate it to the concepts of nonlinear flow networks. We then review some mathematical optimization approaches in the literature. Chapter 3 describes the benchmark test set and the dataset generation.

Part II is dedicated to the exact algorithm. We recall the branch-and-check algorithm of [6] in Chapter 4 and discuss its limitation, regarding the strength of the MILP relaxation in particular. We present our Optimization-Based Bound Tightening (OBBT) techniques in Chapter 5 and our cut generation in Chapter 6. Computational experiments are presented in Chapter 7.

Part III is dedicated to the heuristic algorithm. In Chapter 8, we detail the motivation behind our algorithm and review the recent literature on the hybridization of machine learning and mathematical optimization. We present the principle of ADMs and our variable splitting approach in Chapter 9. We present two different DL architectures in Chapter 10 and Chapter 11. Computational experiments are presented in Chapter 12.

Finally, a conclusion and some perspectives for future works are discussed in Chapter 13.

Some background on deep learning is provided in Appendix A.

Part II (exact algorithm with OBBT and cut generation) has been partly published in the proceedings of the conference ICAE 2022 (International Conference on Applied Energy) [14]. Part III (heuristic algorithm with ADM and DL) has been partly submitted to the AAAI'2023 conference.

Part I

The Pump Scheduling Problem

Chapter 2

Mathematical Formulation

This chapter introduces the pump scheduling problem and its standard mathematical formulation. A good description of the physical meanings of this mathematical model can be found in [6, 15]. The appellation *pump scheduling* covers many variants, and we study, in this thesis, one of the most common variant, which includes: discretized horizon with stationary operations, only fixed speed-drive pumps and gate valves, aging constraints, no leakage, no maximum withdrawal, etc. In Section 2.3, we also explicit a bilevel formulation of the problem including the nonlinear flow equilibrium problem at the inner level.

2.1 Elements of a water network

A Drinking Water Distribution Network, DWDN, is represented as a simple directed graph $\mathcal{G} = (\mathcal{J}, \mathcal{A})$ with nodes $\mathcal{J}$ and arcs $\mathcal{A} \subseteq \mathcal{J} \times \mathcal{J}$ (see e.g. Figures 3.1 and 3.2). Let $E \in \{0, 1, -1\}^{\mathcal{A} \times \mathcal{J}}$ denote the incidence matrix of $\mathcal{G}$, defined for all arc $a \in \mathcal{A}$ and node $j \in \mathcal{J}$ by:

$$E_{aj} = \begin{cases} 1 & \text{if arc } a \text{ enters node } j \text{ (i.e. } a \in \mathcal{J} \times \{j\}), \\ -1 & \text{if arc } a \text{ leaves node } j \text{ (i.e. } a \in \{j\} \times \mathcal{J}), \\ 0 & \text{otherwise.} \end{cases}$$

Nodes $\mathcal{J}$ are categorized into service nodes $\mathcal{S}$ and reservoirs $\mathcal{C}$, and reservoirs are divided into tanks $\dot{\mathcal{C}}$ and sources $\mathcal{C} \setminus \dot{\mathcal{C}}$. Arcs $\mathcal{A}$ are divided into pipes $\mathcal{A}_L$, pumps

$\mathcal{A}_K$ and valves $\mathcal{A}_V$. Pumps and valves are said to be controllable, $\mathcal{A}_K \bigcup \mathcal{A}_V = \dot{\mathcal{A}}$. The scheduling horizon is discretized in T periods, $t \in \mathcal{T} = \{0, ..., T-1\}$ of fixed length Δ (for example, we may consider one day with $T = 24$ and $\Delta = 1$ hour). We denote $\overline{\mathcal{T}} = \mathcal{T} \cup \{T\}$. We assume that the network is in a stationary state during each period $t \in \mathcal{T}$. In particular, the controllable arcs can be commanded only at time instants $t \in \mathcal{T}$, and we neglect the turbulence due to the change of state between two consecutive periods.

The transport of the water in the network is then characterized by the flow $q_{ta} \in \mathbb{R}$ (measured, e.g., in liter per second, L/s) going through each arc $a \in \mathcal{A}$ during a period $t \in \mathcal{T}$, with the convention that $q_{ta} \geq 0$ if water flows in the direction of the arc, and $q_{ta} \leq 0$ if water flows in the opposite direction. We assume that $q_{ta} \geq 0$ if arc a is unidirectional. The inflow (L/s) at a node $j \in \mathcal{J}$ is then defined as $q^{\mathcal{J}}_{tj} = \sum_{a \in \mathcal{A}} E_{aj} q_{ta}$.

The movement of the water is induced by its potential or pressure at the nodes of the network. The potential is also known as the hydraulic or piezometric head (in meters), which measures the sum of the geographical elevation and the water pressure head. As for the flow, we assume that the hydraulic head $h_{tj} \in \mathbb{R}_+$ is constant during each period $t \in \mathcal{T}$ for each node $j \in \mathcal{J}$. The head loss (in meters) through an arc $a \in \mathcal{A}$, i.e., the difference of head between its origin and destination, is then defined as $v_{ta} = -\sum_{j \in \mathcal{J}} E_{aj} h_{tj}$.

2.1.1 Nodes

Service nodes. A node $j \in \mathcal{S}$ represents either an arc junction or a service node from which the water is delivered to groups of consumers. Flow conservation and demand satisfaction at these kind of nodes are enforced as follows:

$$q^{\mathcal{J}}_{tj} = D_{tj}, \qquad \forall t \in \mathcal{T}, j \in \mathcal{S}$$

where $D_{tj} \in \mathbb{R}$ denotes the forecasted demand (in L/s) on period $t \in \mathcal{T}$ if $j \in \mathcal{S}$ is a service node or $D_{tj} = 0$ if it is a simple junction. Besides, the water has to be served within a certain pressure range (given in m):

$$h_{tj} \in [\underline{H}_j, \overline{H}_j] \subset \mathbb{R}_+, \qquad \forall t \in \mathcal{T}, j \in \mathcal{S}$$

Sources. They are represented as nodes $j \in \mathcal{C} \setminus \dot{\mathcal{C}}$ from which the water is supplied to consumers. The source nodes are assumed to have a known hydraulic head (given in meters) at any time, which we model as:

$$h_{tj} \in [\underline{H}_{tj}, \overline{H}_{tj}], \text{ with } \underline{H}_{tj} = \overline{H}_{tj} \in \mathbb{R}_+, \qquad \forall t \in \mathcal{T}, j \in \mathcal{C} \setminus \dot{\mathcal{C}}$$

Tanks. They are utilities to store and discharge the water in the network. We assume tanks are cylinders of known surface lying at a known elevation. Consequently, the hydraulic head at a tank node $j \in \dot{\mathcal{C}}$ is linearly related to the height of the water in the cylinder, thus to the volume and the inflow. Therefore, flow conservation at these nodes can be written as follows:

$$\sigma_j q^{\mathcal{J}}_{tj} = h_{(t+1)j} - h_{tj} \qquad \forall t \in \mathcal{T}, j \in \dot{\mathcal{C}},$$

where $\sigma_j = \frac{\Delta}{S_j} \in \mathbb{R}_+$ (with S_j the surface of the tank) is the linear factor between volume and head variation during a time period.

The tanks have limited capacities (in volume/height), which can be written as lower and upper bounds (in m) on the head values:

$$h_{tj} \in [\underline{H}_j, \overline{H}_j] \subset \mathbb{R}_+ \qquad \forall t \in \mathcal{T}, j \in \dot{\mathcal{C}}.$$

Pump scheduling is typically repeated every day, and the initial volume of the tanks at $t = 0$ is assumed to be known from the day before. In addition, the objective to minimize the energy costs tends to empty the tanks at the end of the schedule. With a high forecasted demand to satisfy at the beginning of the consecutive day, there would be a risk that pumps are not able to fulfill the requested consumption; there ought to be enough stored water in the tanks to intervene. It is thus reasonable to enforce that the levels of the tanks at the end of the horizon are not below their initial levels:

$$h_{Tj} \geq h_{0j} = H_{0j} \qquad \forall j \in \dot{\mathcal{C}}.$$

In a tank $j \in \dot{\mathcal{C}}$, a positive inflow $q_{jt} \geq 0$ is considered as water storage and a negative inflow as discharge.

2.1.2 Arcs

Pipes. They are responsible for transporting the water inside the network. They are represented as arcs $a \in \mathcal{A}_L$ in the graph, directed according to the orientation

of the water. In particular, if a pipe $a = (i, j) \in \mathcal{A}_L$ is unidirectional, then water is assumed to pass from node i to j and the flow value is positive, otherwise, if the pipe is bidirectional, the water may also pass from j to i, and then the flow value would be negative. Hence, we have the following constraint for unidirectional pipes:

$$q_{ta} \geq 0 \quad \forall t \in \mathcal{T}.$$

The flow passing through a pipe creates friction, inducing a head loss between the extremities of the pipe. Typically the head loss relationship is described by Darcy-Weisbach equation:

$$v_{ta} = \frac{8 L_a \tau_a q_a |q_a|}{\pi^2 g D_a^5}$$

and the Hazen-Williams equation:

$$\frac{10.7 L_a q_a |q_a^{0.852}|}{\kappa_a^{1.852} D_a^{4.8704}}$$

In this thesis, following [6, 16, 17], the nonlinear relation of the head loss through a pipe are fitted as quadratic relationships:

$$v_{ta} = \psi_a(q_{ta}) = A_a q_{ta} |q_{ta}| + B_a q_{ta} \qquad \forall t \in \mathcal{T}, a\mathcal{A}_L.$$

The parameters $A_a > 0$ and $B_a \geq 0$ are obtained by extrapolating the experiments to have injective function representing the head loss relationship. This guarantees the strict convexity of the solution and the uniqueness of the head-flow relationship (see Section 2.3.1).

Gate valves. They are used to fully opening and closing a pipe. Note, that in this thesis, we do not consider other types of valves like ball valves or pressure-reducing valves. Since gate valves are implemented on pipes, we model an integrated pair of pipe and valve as one controlled arc $a = (i, j) \in \mathcal{A}_V$ inheriting from the characteristics of the pipe (in particular the resistance function ψ_a). A gate valve has two possible states: when closed, water passes through it, and when open, water is stopped, and the hydraulic heads upstream and downstream are decoupled. The states are modeled as binary variables: $x_{ta} = 1$ if the valve $a \in \mathcal{A}_V$ is closed at period $t \in \mathcal{T}$, and $x_{ta} = 0$ if the valve is closed. Therefore, flow and head loss through a pipe equipped with a

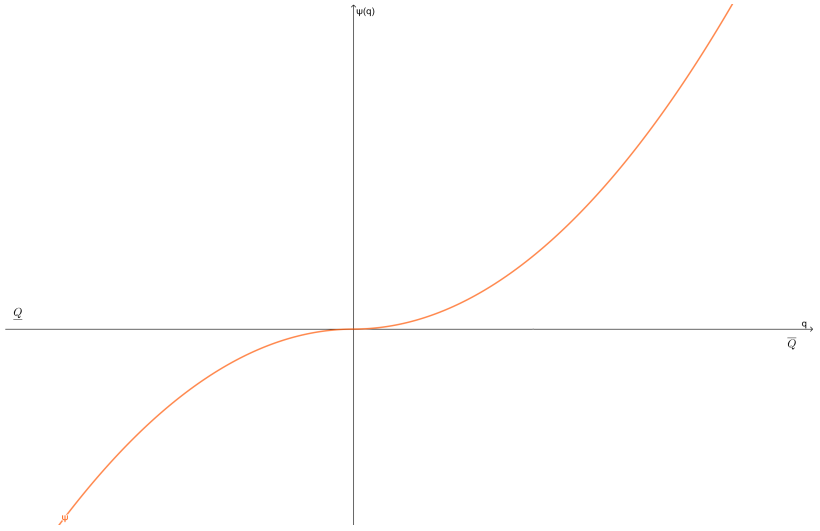

Figure 2.1: Typical head-flow relationship in pipes.

valve $a \in \mathcal{A}_V$ are modeled as for simple pipes, but with a boolean condition:

$$\begin{aligned} x_{ta} \in \{0,1\}, q_{ta} \in \mathbb{R}, v_{ta} \in \mathbb{R} \qquad & \forall t \in \mathcal{T}, a \in \mathcal{A}_V \\ x_{ta} = 0 \implies q_{ta} = 0, \qquad & \forall t \in \mathcal{T}, a \in \mathcal{A}_V \\ x_{ta} = 1 \implies v_{ta} = \psi_a(q_{ta}), \qquad & \forall t \in \mathcal{T}, a \in \mathcal{A}_V \end{aligned}$$

Pumps. They are powered to increase the hydraulic head at some network nodes. They are represented as controllable arcs $a \in \mathcal{A}_K$, directed according to the flow (i.e., $q_{ta} \geq 0$ for all $t \in \mathcal{T}$). In this thesis, we only consider fixed-speed pumps with only two possible states (on or off), which can be changed only at time steps $t \in \mathcal{T}$. As for gate valves, we thus model the status of the pumps with binary variables:

$$x_{ta} \in \{0,1\}, \qquad \forall t \in \mathcal{T}, a \in \mathcal{A}_K.$$

When a pump $a = (i,j) \in \mathcal{A}_K$ is off (i.e., $x_{ta} = 0$), it acts like an open valve (as if the arc were removed from the graph). When the pump is on (i.e., $x_{ta} = 1$), it permits the flow to pass from i to j, in the operational limits $[\underline{Q}_a, \overline{Q}_a] \subseteq \mathbb{R}_+$ of the pump. Together with the pump status, these limits can be modeled as follows:

$$\begin{aligned} x_{ta} = 0 \implies q_{ta} = 0, \qquad & \forall t \in \mathcal{T}, a \in \mathcal{A}_K \\ x_{ta} = 1 \implies q_{ta} \in [\underline{Q}_a, \overline{Q}_a], \qquad & \forall t \in \mathcal{T}, a \in \mathcal{A}_K. \end{aligned}$$

Following [15, 18], we model the head increase (i.e., the opposite of the head loss v_{ta}) as a quadratic function of the flow $q \geq 0$ passing through the pump:

$$-\psi_a(q) = A_a q^2 + B_a q_a + C_a.$$

in which we assume strictly concave function to describe head gain. The conditional relation between status, head and flow, can then be modeled as follows:

$$x_{ta} = 1 \implies v_{ta} = \psi_a(q_{ta}), \qquad \forall t \in \mathcal{T}, a \in \mathcal{A}_K.$$

Following [15] for fixed speed pumps again, we assume that the electric power consumption (kW) of a pump is linear in the flow q passing through it, with an additional cost when the pump is active ($x = 1$):

$$\gamma_a(x, q) = \gamma_a^0 x + \gamma_a^1 q$$

where $\gamma_a^0, \gamma_a^1 > 0$ are parameters computed empirically and provided by the pump manufacturer.

To mitigate the aging of the pumps, we may consider additional constraints on the number and frequency of operations. To this purpose, Ghaddar et al. [4] introduced the following constraints:

$$\begin{aligned}
&\sum_{t \in \mathcal{T}} u_{ta} \leq N, && \forall a \in \mathcal{A}_K \\
&u_{ta} \geq x_{ta} - x_{(t-1)a}, && \forall t \in \mathcal{T}, a \in \mathcal{A}_K \\
&u_{ta} \leq x_{t'a}, && \forall t, t' \in \mathcal{T} : 0 < t \leq t' \leq t + \tau_a^1, a \in \mathcal{A}_K \\
&w_{ta} \geq x_{(t-1)a} - x_{ta}, && \forall t \in \mathcal{T}, a \in \mathcal{A}_K \\
&w_{ta} \leq 1 - x_{t'a} && \forall t, t' \in \mathcal{T} : 0 < t \leq t' < t + \tau_a^0, a \in \mathcal{A}_K \\
&u_{ta}, w_{ta} \in \{0, 1\} && \forall t \in \mathcal{T}, a \in \mathcal{A}_K,
\end{aligned}$$

where the additional binary variables (u_{ta} and w_{ta}) models the switching operation (on and off) of a pump $a \in \mathcal{A}_K$ at time step $t \in \mathcal{T}$. The first constraint sets a maximum number of times $N \in \mathbb{Z}_+^*$ that a pump can be switched on. The other constraints fix the minimum number of consecutive periods that a pump has to remain on ($\tau_a^1 \in \mathbb{Z}_+^*$) and off ($\tau_a^0 \in \mathbb{Z}_+^*$). These constraints define a discrete set modeled as an integer polyhedron $\mathcal{X}_a \subseteq \{0, 1\}^{\mathcal{T}}$, which defines the domain of the schedule for a given pump $a \in \mathcal{A}_K$:

$$(x_{ta})_{t \in \mathcal{T}} \in \mathcal{X}_a.$$

2.1.3 Power cost

The prime objective for the water network operator is to satisfy the demand while minimizing the energy bill. When following a given electricity tariff profile $C \in \mathbb{R}^{\mathcal{T}}$ (kWh), this objective ties up with optimizing the energy consumption by limiting and delaying the pumping thoroughly on the time horizon.

We then consider the following objective function to be minimized [6]:

$$\sum_{t\in\mathcal{T}} \sum_{a\in\mathcal{A}_K} c^0_{ta} x_{ta} + c^1_{ta} q_{ta},$$

with $c^i_{ta} = C_t \Delta\gamma^i_a, \forall t \in \mathcal{T}, a \in \mathcal{A}_K$, $i = \{0, 1\}$.

2.2 Nonconvex MINLP formulation

We can elaborate a mathematical program for the pump scheduling problem after reformulating the logical relations involving the binary variables as analytical functions. There are different ways to handle logical relations of the following type:

$$x = 1 \implies f(y) \leq 0, x \in \{0, 1\}, y \in \mathbb{R}^n,$$

where f is a non-constant function. The above relation indicates that the inequality is deactivated when the binary variable x is zero.

The most famous modeling trick in MILP is the big-M reformulation or *indicator constraints* [19]:

$$f(y) \leq M(1 - x)$$

which requires knowing a valid upper bound M on f, i.e., $f(y) \leq M$ for any solution (x, y) of the problem such that $x = 0$. Indeed, if f is linear, the big-M reformulation remains linear, however, obtaining a valid bound M is not always trivial. Furthermore, if M must be chosen as a large value (compared to the other data in the model) to be valid, then it may induce a poor LP relaxation as well as possible numerical issues.

In the context of nonlinear programming, one may use the complementary reformulation instead:

$$f(y)x \leq 0$$

which creates a nonlinear nonconvex constraint involving a bilinear term xy, for example, if f is linear.

Finally, note that indicator constraints are now available in the API of several off-the-shelf MILP solvers when f is linear. Their implementation, for instance, in the SCIP solver [20] leads to the reformulation:

$$f(y) \leq s,\ s \geq 0,\ SOS1(x, s)$$

This introduces a slack variable s, and the Special Ordered Set of type 1 (SOS1) condition enforces at most one of x and s to be nonzero.

In the literature on pump scheduling, the big-M reformulation is often observed, in particular, in MILP approaches based on piecewise linear relaxations or approximations of the nonlinear constraints. In this context, big-M values of reasonable size can be easily retrieved from the analysis of the problem. Computing tight big-M values is precisely the topic of our work presented in Chapter 5.

We first formulate the pump scheduling as a MINLP using big-M reformulations of the indicator constraints. We then introduce bounds $[\underline{Q}, \overline{Q}]$ and $[\underline{V}, \overline{V}]$ on the flow q and head loss v variables, which are either specified exogenously or computed at a preprocessing step. Also, we homogenize the nodal constraints by introducing time-indexed bounds on the head variables h. If not specified, the lower bounds $\underline{H}$ is set to $-\infty$, and upper bounds $\overline{H}$ to $+\infty$.

To get a unified notation and a synthetic model, we introduce a constant $x_{ta} = 1$ for each non-controllable arc (i.e., simple pipe) $a \in \mathcal{A}_L = \mathcal{A} \setminus \dot{\mathcal{A}}$ and for each time $t \in \mathcal{T}$. We thus use the notation $(x_{ta})_{t\in\mathcal{T}} \in \mathcal{X}_a$, with $\mathcal{X}_a = \{(1)_{t\in\mathcal{T}}\}$ for a simple pipe $a \in \mathcal{A}_L$, and $\mathcal{X}_a = \{0, 1\}^{\mathcal{T}}$ for a pipe with valve $a \in \mathcal{A}_V$. The cost function is also extended to pipes by setting $c^0_{ta} = c^1_{ta} = 0$ for all $a \in \mathcal{A}_L \cup \mathcal{A}_V$.

The pump scheduling is then formulated as the following MINLP:

$$(\mathcal{P}) : \min_{x,q,h} \sum_{t\in\mathbb{T}}\sum_{a\in\mathcal{A}}(c^0_{ta}x_{ta} + c^1_{ta}q_{ta}) \tag{2.2}$$

$$\text{s.t: } q^{\mathcal{J}}_{tj} = D_{tj}, \qquad \forall j \in \mathcal{S}, \forall t \in \mathcal{T} \tag{2.3}$$

$$q^{\mathcal{J}}_{tj} = \sigma_j(h_{(t+1)j} - h_{tj}), \qquad \forall j \in \dot{\mathcal{C}}, \forall t \in \mathcal{T} \tag{2.4}$$

$$\underline{V}_{ta}(1 - x_{ta}) \leq v_{ta} - \psi_a(q_{ta}) \leq \overline{V}_{ta}(1 - x_{ta}) \qquad \forall a \in \mathcal{A}, \forall t \in T \tag{2.5}$$

$$\underline{Q}_{ta}x_{ta} \leq q_{ta} \leq \overline{Q}_{ta}x_{ta}, \qquad \forall a \in \mathcal{A}, \forall t \in \mathcal{T} \tag{2.6}$$

$$\underline{H}_{tj} \leq h_{tj} \leq \overline{H}_{tj}, \qquad \forall j \in \mathcal{J}, \forall t \in \mathcal{T} \tag{2.7}$$

$$x_a \in \mathcal{X}_a \subseteq \{0,1\}^{\mathcal{T}} \qquad \forall a \in \mathcal{A}. \tag{2.8}$$

The above problem is quadratically constrained and nonconvex.

2.3 A bilevel formulation

If the monolithic MINLP above is the most encountered formulation for the pump scheduling problem, a large share of the literature dedicated to this problem considers implicitly a bilevel formulation in the solution process. This is the case in many simulation-based metaheuristics and in decomposition or exact approaches [6, 21]. All these methods rely on the fact that once all binary variables are fixed, the problem reduces to compute a flow-head equilibrium at every time step sequentially. Therefore, the problem can be seen as a bilevel program with binary variables (the scheduling decisions) at the upper level and nonlinear constraints (the resistance relations) at the lower level, which are the KKT conditions of an inner strictly convex subproblem.

2.3.1 The equilibrium problem

Given any vectors $H \in \mathbb{R}^{\mathcal{C}}$, $D \in \mathbb{R}^{\mathcal{S}}$, we define the *equilibrium problem* on graph $\mathcal{G}$ as computing an element in the following set:

$$\mathcal{E}(H, D) = \{(q, h) \in \mathbb{R}^{\mathcal{A}} \times \mathbb{R}^{\mathcal{S}} : \tag{2.9}$$

$$v_a = \psi_a(q_a), \qquad \forall a \in \mathcal{A}, \tag{2.10}$$

$$q^{\mathcal{J}}_j = D_j, \qquad \forall j \in \mathcal{S}\} \tag{2.11}$$

where $q_j^{\mathcal{J}} := \sum_{a\in\mathcal{A}} E_{aj} q_a$ the inflow at node $j \in \mathcal{J}$ and $v_a := -\sum_{j\in\mathcal{S}} E_{aj} h_j - \sum_{j\in\mathcal{C}} E_{aj} H_j$ the head loss on arc $a \in \mathcal{A}$.

The elements of $\mathcal{E}(H, D)$ are the primal/dual optimal pairs of the following NLP:

$$(\mathcal{E}_P(H,D)) : \min_{q\in\mathbb{R}^{\mathcal{A}}} \{ f(q) := \sum_{a\in\mathcal{A}} \Psi_a(q_a) + \sum_{j\in\mathcal{C}} H_j q_j^{\mathcal{J}} : q_j^{\mathcal{J}} = D_j, \ \forall j \in \mathcal{S} \}.$$

where Ψ_a is the antiderivative of ψ_a passing through 0 for each arc $a \in \mathcal{A}$.

Indeed, $(\mathcal{E}_P(H, D))$ is a linearly-constrained convex NLP, thus strong duality holds and the primal/dual pairs are the stationary point of the Lagrangian function

$$L_{(H,D)}(q,h) = f(q) + \sum_{j\in\mathcal{S}} h_j(q_j^{\mathcal{J}} - D_j) = \sum_{a\in\mathcal{A}} (\Psi_a(q_a) + \sum_{j\in\mathcal{C}} E_{aj} H_j q_a + \sum_{j\in\mathcal{S}} E_{aj} h_j q_a) - \sum_{j\in\mathcal{S}} D_j h_j$$

Thus,

$$L_{(H,D)}(q,h) = \sum_{a\in\mathcal{A}} (\Psi_a(q_a) - v_a q_a) - \sum_{j\in\mathcal{S}} D_j h_j,$$

and its stationary points $(q, h) \in \mathbb{R}^{\mathcal{A}\times\mathcal{S}}$ are defined by $\frac{\partial L}{\partial q_a}(q,h) = \psi_a(q_a) - v_a = 0$ and $\frac{\partial L}{\partial h_j}(q,h) = q_j^{\mathcal{J}} - D_j = 0$. The set $\mathcal{E}(H, D)$ is then the set of KKT solutions of the NLP $(\mathcal{E}_P(H, D))$.

The objective function f is actually a measure of the total energy dissipation in the network induced by the resistance on the arcs, thus the equilibrium corresponds to a flow with minimal energy dissipation.

Under our assumption, the functions ψ_a are bijective on $\mathbb{R}$ for every $a \in \mathcal{A}$, and their antiderivatives Ψ_a are strictly convex. As a consequence, the objective function $f(q)$ is strictly convex and the optimal/dual solution is thus unique. Furthermore, because it is separable, we easily obtain the dual formulation. Indeed, for any dual vector $h \in \mathbb{R}^{\mathcal{S}}$, $L_{(H,D)}(., h)$ reaches its minimum at $q = (\psi_a^{-1}(v_a))_{a\in\mathcal{A}}$, thus the Lagrangian dual problem can be written as follows (taking the opposite optimum value):

$$(\mathcal{E}_D(H,D)) : \max_{h\in\mathbb{R}^{\mathcal{S}}} \min_{q\in\mathbb{R}^{\mathcal{A}}} L(q,h) = \min_{h\in\mathbb{R}^{\mathcal{S}}} g(h) := \sum_{a\in\mathcal{A}} \Psi_a^*(v_a) + \sum_{j\in\mathcal{S}} D_j h_j.$$

where $\Psi_a^* : v \in \mathbb{R} \mapsto \max_{q\in\mathbb{R}}(vq - \Psi_a(q)) = -\Psi_a(\psi_a^{-1}(v)) + v\psi_a^{-1}(v)$ is the convex conjugate of Ψ_a, $\forall a \in \mathcal{A}$.

The results above have been studied in the more general context of nonlinear flow

networks and monotropic programming by Rockafellar [13]. In [13], models ($\mathcal{E}_P$) and ($\mathcal{E}_D$) are referred to, respectively, optimal distribution and optimal differential problems, where Ψ_a are the nonlinear flow cost functions on each arc a. Collins [22] also studied these models in the context of hydraulic networks, referring to them as the content and co-content models.

The equilibrium problem above is the base of hydraulic simulation tools, such as EPANET [7]. It is usually solved with the Newton-Raphson technique to compute (q, h) as a solution of the equation system $\nabla L(q, h) = 0$, that is, following the algorithm formalized by Todini and Pilati [23].

2.3.2 Dynamic network and Extended Analysis

The pump scheduling problem appears as solving the equilibrium problem in a dynamic network, where vectors H and D change at each time step, so as the graph changes with the activation of the controllable arcs. Precisely, given a schedule $x \in \{0,1\}^{\mathcal{T}\mathcal{A}}$ of the pumps and valves over the whole horizon, we can check if it corresponds to a feasible solution for the pump scheduling problem by, progressively, for each $t \in \mathcal{T}$, computing a solution (q_t, h_t) of an equilibrium problem in the active subgraph of $\mathcal{G}$ defined by x_t. Afterward, we can update the tank heads h_{t+1} according to (2.4). This is known as the Extended Analysis Algorithm [24] as detailed below in Algorithm 1. Note that the given schedule is unfeasible if the value of the hydraulic heads does not satisfy the tank capacities (2.7) at some t.

We first introduce the following notation for restricting the equilibrium problem to the subgraph of $\mathcal{G}$ defined by a vector $x \in \{0,1\}^{\mathcal{A}}$, with $H \in \mathbb{R}^{\mathcal{C}}$ and $D \in \mathbb{R}^{S}$:

$$\mathcal{E}(H, D, x) = \{(q, h) \in \mathbb{R}^{\mathcal{A}} \times \mathbb{R}^{\mathcal{S}} : \tag{2.12}$$

$$q_a = 0, \qquad \forall a \in \mathcal{A} : x_a = 0, \tag{2.13}$$

$$v_a = \psi_a(q_a), \qquad \forall a \in \mathcal{A} : x_a = 1, \tag{2.14}$$

$$q_j^{\mathcal{J}} = D_j, \qquad \forall j \in \mathcal{S}\} \tag{2.15}$$

Algorithm 1 Extended Analysis Algorithm

Input $H^0 \in \mathbb{R}^{\mathcal{C}}$, $x \in \{0,1\}^{\mathcal{T}\mathcal{A}}$

Output a feasible solution of $\mathcal{P}$ or the first period for a violation

for $t \in \mathcal{T}$ **do**:
    compute $(q_t, h_t) \in \mathcal{E}(H_t, D_t, x_t)$
    compute $H_{t+1} = H_t + \frac{q_t^{\mathcal{J}}}{\sigma}$
    if $H_{t+1} \notin [\underline{H}_{t+1}, \overline{H}_{t+1}]$ **then**:
        return t
    end if
end for
return (x, q, h)

2.3.3 Graph partition

In addition to temporal decomposition, we can spatially decompose the pump scheduling problem once the level of the tanks is known, following a graph partition along the reservoirs. The rationale is that the flow and head of arcs and nodes belonging to a subgraph are only determined by pump status and levels of the tanks in the subgraph at each time step.

Precisely, the equilibrium problem is separable given any partition of $\mathcal{G}$ along nodes in $\mathcal{C}$. Indeed, consider a partition of $\mathcal{G}$ into subgraphs $\mathcal{G}^b = (\mathcal{J}^b, \mathcal{A}^b) \subseteq \mathcal{G}$ for $b \in B$, such that $\mathcal{A} = \bigsqcup_{b \in B} \mathcal{A}^b$ (disjoint union) and $\mathcal{J}^b \cap \mathcal{J}^{b'} \subseteq \mathcal{C}$, $\forall b, b' \in B, b \neq b'$. Since any arc $a \in \mathcal{A}$ and node $j \in \mathcal{S}$ belongs to exactly one subgraph $\mathcal{G}^b$, then (q, h) is an equilibrium on $\mathcal{G}$ if and only if its restriction is an equilibrium on every subgraph $\mathcal{G}^b$, that is:

$$(q, h) \in \mathcal{E}(H, D) \iff (q^b, h^b) \in \mathcal{E}(H^b, D^b)\ \forall b \in B,$$

where $b \in B$ as an exponent denotes the restriction of the vectors to elements of $\mathcal{G}^b$, i.e., $q^b = (q_a)_{a \in \mathcal{A}^b}$, $h^b = (h_j)_{j \in \mathcal{S} \cap \mathcal{J}^b}$, $D^b = (D_j)_{j \in \mathcal{S} \cap \mathcal{J}^b}$ and $H^b = (H_j)_{j \in \mathcal{C} \cap \mathcal{J}^b}$.

2.3.4 Bilevel formulation of the pump scheduling

Finally, we can reformulate the pump scheduling problem as follows:

$$(\mathcal{P}) : \min_{x,q,h,H} \sum_{t\in\mathcal{T}}\sum_{a\in\mathcal{A}}(c_t^0 x_{ta} + c_t^1 q_{ta}) \tag{2.16}$$

$$\text{s.t: } (q_t, h_t) \in \mathcal{E}(H_t, D_t, x_t), \qquad \forall t \in \mathcal{T} \tag{2.17}$$

$$q_{tj}^{\mathcal{C}} = \sigma_j(H_{(t+1)j} - H_{tj}), \qquad \forall j \in \dot{\mathcal{C}}, \forall t \in \mathcal{T} \tag{2.18}$$

$$\underline{H}_{tj} \leq H_{tj} \leq \overline{H}_{tj}, \qquad \forall j \in \mathcal{C}, \forall t \in \overline{\mathcal{T}} \tag{2.19}$$

$$x_a \in \mathcal{X}_a \subseteq \{0,1\}^{\mathcal{T}} \qquad \forall a \in \mathcal{A}. \tag{2.20}$$

In this model, we distinguish heads at service nodes (denoted by h), which only appear in the lower level program (as dual variables), and heads at reservoirs (denoted by H), which appear in the upper level program.

2.4 State of the Art

The pump scheduling problem is the subject of a vast literature. A popular optimization approach is based on metaheuristics, mainly evolutionary algorithms, that were a zeitgeist for a long period since the mid 90's. For instance, in [25], the authors employ a genetic algorithm to solve an instance with a night/day tariff profile, and a single demand profile stands for average daily consumption. In [26], a hybrid approach combining a genetic algorithm with a local search strategy was developed. The ranking function in the genetic algorithm is an inverse version of the cost penalized by constraint violation. In [27], the authors have developed an ant colony algorithm in which the ranking function prioritizes the less infeasible solutions. With additional maintenance constraints, the search space is drastically reduced, and their model uses the duration a particular pump stays on as a decision variable. The method is experimented over networks with a single demand profile. Other references are described in [28, 29]. Beyond the fact that metaheuristics do not provide information about the quality of the cost of the computed solution, in many applications to the pump scheduling problem, some of the functional constraints, in particular on the capacity of the tanks, are relaxed and penalized in the fitness function. On the other hand, they usually evaluate precisely the nonlinear resistance constraints by integrating a hydraulic simulator.

This is often the opposite in mathematical programming approaches for the pump scheduling. Global optimization solvers are not able to compute feasible solutions for even small-scale networks with a slightly high number of periods [4, 12, 30, 31].

Hence, in [32], a piecewise linearization of the resistance relationships is employed to approximate the optimal solution of the problem. Different numbers of pieces are considered to mimic the initial non-linear curve, and then a polyhedral approximation is passed to a MILP solver. The impact of the number of pieces can be observed in two different directions. More accurate approximation induces less violation and a higher possibility of ending up with a configuration, but simultaneously, less and less efficient algorithm.

Gleixner et al. [33] tackle the global optimality of the static operation problem over a single time-step. In other words, they assume a constant pressure at each tank and neglect the dynamic variations in the levels. Interestingly, they have proposed presolving methods, to mitigate the issue of large-scale and non-linear constraints. They deploy constraint propagation to reduce the domain of the variables in the MINLP formulation, or introduce dependencies constraints to break symmetries in the pumping station (parallel identical pumps). Global optimality is then achieved

by deploying a spatial branch-and-bound through the SCIP solver. As pointed out by the authors, the approach becomes intractable when the full-scale operation problem is considered with only two or three time periods.

Ghaddar et al. [4] take advantage of the fact that flow conservation at tanks are the only time-coupling constraints in the MINLP model. Therefore, dualizing this constraint decomposes each time interval formulation. The authors apply Lagrangian relaxation, solving each single-step subproblem with a MINLP solver, and the dual problem with a cutting-plane algorithm. While the lower bound of the Lagrangian relaxation is still valid for the original formulation, the solution found through this decomposition may be infeasible due to the relaxation. To recover feasibility and to enforce the time-coupling constraints on the aging of the pumps, the authors combine improved Limited Discrepancy Search (ILDS) with a branch-and-bound algorithm.

Naoum-Sawaya et al. [8] suggest a simulation-based optimization approach. A master MILP is responsible for finding binary schedules, while the follower checks their feasibility against the nonlinear equilibrium subproblems. Unfeasibility is reported progressively to the master program as combinatorial Benders cuts. This led to a better performance over *Poormond* benchmark.

Costa et al. [9] also apply an optimization-simulation approach, but evaluates explicitly all possible binary schedules. This may work for small networks, but the scalability of the method might be an issue.

Bonvin et al. [31] consider a branched network with parallel pumps located at the root. They show that a MILP relaxation provides an exact formulation when all pumps are identical. In other cases, a relaxed solution can easily be turned into a feasible solution by propagating the head loss correction along the branches, and the optimality gap is bounded.

Shi et al. [10] consider optimization-simulation within an outer-approximation framework: the master program is partially tightened by piecewise linear outer cuts, while the follower subproblems and the number of pieces are automatically refined during the search.

Fooladivanda and Taylor [34] introduce a second order cone relaxation and show conditions (presence of pressure valves and variable speed pumps) for the relaxation to be exact. To find high quality solution, an ADMM-based algorithm is developed.

Bonvin and Demassey [35] propose an extended continuous reformulation of the problem by allowing changing pump configurations during the time steps. The decision variables become the durations of each configuration within each time step.

By ignoring the impact of the tank levels on the output of the configurations, they propose a 3-step approach: (i) generate the configurations and check the hydraulic equilibrium (ii) populate the columns of the LP approximation with the feasible configurations (iii) apply a Variable Neighborhood Search heuristic based on mathematical programming and inspired from [8] to recover a feasible solution.

Vieira et al. [11] consider a polyhedral relaxation, refined progressively by adding breakpoints within a given error level. The solution computed by branch-and-cut might not be feasible, so they apply two specific heuristics to recover feasibility. The corrective actions are implemented in case of irregular pressure at a node and when the final tank levels are lower than the initial ones.

Bonvin et al. [6] introduce a tailored MILP relaxation and deploy a branch-and-check approach: at each feasible (integer) node of the branch-and-bound, the feasibility of the associated binary schedule is checked against the equilibrium subproblems. If unfeasible, a combinatorial nogood cut is generated, otherwise the cost of the feasible solution is computed and recorded as the new incumbent. The framework is also developed for pump scheduling with variable-speed pumps or pressure-reducing valves. In this context, the subproblem is tackled with a nonlinear program solver. Finally, in order to speed up the search, the authors also introduce an original primal heuristic to compute near-feasible solutions by adjusting the duration of the activity of the pumps. This is formulated as a MINLP but solved iteratively as a sequence of restricted linear programs.

More recently, Tassef et al. [12] investigate a similar approach enhanced with a more aggresive bound tightening and the progressive refinement of the relaxation of the resistance functions, with OA cuts, and with strong duality cuts [36] issuing from the linearization of the strong duality condition. the optimal pump scheduling problem by applying bound tightening techniques. These techniques have led to a domain reduction over feasible flows and the generation of cuts for decision variables.

Finally, we may refer to related works applying MINLP to other optimization problems in the context of water networks, including pipe dimensioning in gravity-fed networks [30, 37] or pump dimensioning in pressurized networks [38], or in the context of other energy infrastructural systems, in particular, gas networks [39, 40, 41, 42, 43]. These energy systems can also be represented as a potential-driven networks, and the equilibrium problem appears as a subproblem. Hence, due to some resemblance in the mathematical formulations, some ideas can be interchangeably adopted. Still, the dynamic nature of the pump scheduling, where water tanks act as storage devices and induce temporal interdependencies, make this problem particularly intractable.

Chapter 3

Benchmark and Dataset Generation

3.1 Existing networks and instances

Several water networks are considered in the literature. In the context of design of gravity-fed networks, the moderate instances [12] are *shamir*, *blacksburg*, *hanoi*, *foss-poly-0*, and *foss-iron* and among the large instances we can enumerate *fosspoly1*, *pescara*, and *modena*. However, for the pump scheduling problem, only a few networks are covered in the literature. One of the most analyzed is a toy network called *simple network* (or *FSD*), which comprises three symmetrical pumps parallel to each other. A pipe after the pump station transports water to a storage tank, and a connecting pipe is attached to a demand node. While this network may be too small to represent a real-scale counterpart, its underlying mechanism bears a resemblance to various other practical applications. Another case is represented by *van Zyl* water network [44]. As shown in Fig. 3.1, it consists of a pump station with two parallel symmetrical pumps and a booster pump to feed two tanks with different capacities. The booster pump and the tanks are in an acyclic loop and the same gravity line. The two demand nodes are located in this acyclic loop. Originally, the *van Zyl* network includes a pressure reduction valve. Following [45], we propose to change it to a gate valve.

One of the largest networks considered in the literature is *Poormond*, initially consisting of six cascading tanks supplying different pressure zones, with a primary water source coming from boreholes within the lowest zone [26]. To the best of our knowledge, the literature indicates this network as the most difficult to handle. We

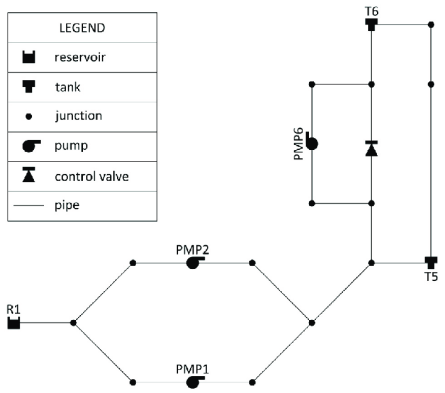

Figure 3.1: *van Zyl* water network distribution.

consider the version used in [6, 12] with five storage tanks depicted in Figure 3.2.

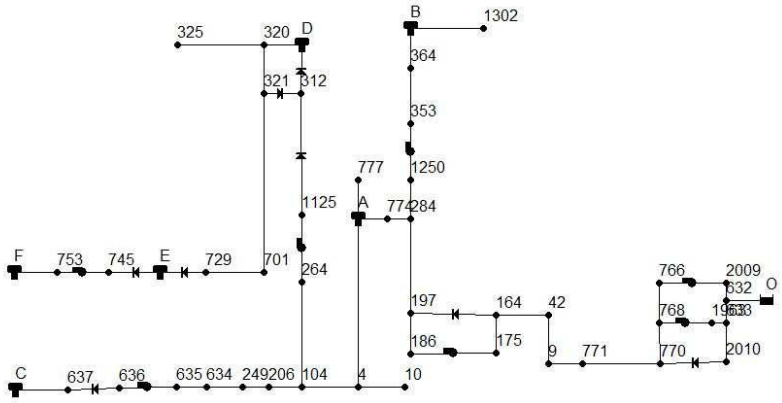

Figure 3.2: *Poormond* water network distribution

We show in Table 3.1 details about the networks considered in this work in terms of discretization of the time horizon, number of nodes and arcs.

A known issue in the optimization of the pump scheduling problem is the lack of a sufficient number of instances (i.e., actual time series of demand and tariff) to be used for proving the effectiveness of a proposed resolution method. In the original description of each of these networks, a single demand profile representing the average day consumption was considered [27, 44]. The demand at each node within the set $\mathcal{S}$ is established based on a fixed scaling factor. This implies that a demand profile is expressed as a vector rather than utilizing a matrix where each element represents demand profiles at individual nodes. The idea behind the constant proportionate

Network	$\mathcal{T}$	$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\dot{\mathcal{A}}\rvert$	$\lvert\dot{\mathcal{C}}\rvert$	$\lvert\mathcal{S}\rvert$
FSD	{12, 24, 48}	6	5	3	1	1
van Zyl	{12, 24, 48}	16	18	4	2	2
Poormond	{12, 24, 48}	52	55	11	5	11

Table 3.1
Details of the water networks used as a benchmark.

demand among different nodes likely has a basis in real-world observations. For tariff profiles, [4, 6] have added instances to the benchmark by considering dynamic tariff profiles instead of the simple day/night tariffs used before.

In Section 3.2, we first propose to consolidate a benchmark testset for the pump scheduling problem based on these networks. In Section 3.3, in order to investigate a supervised learning approach, we propose a method to generate a varied collection of instances (i.e. daily demand and tariff profiles) for any given network, by regression from the historical data of the real network studied in [38]. This network is operated in a touristic zone in Brittany, France, and the real consumption data show a high seasonality.

The generated instances (the benchmark set and the data set), corresponding solutions, and the generation code are made publicly available at: `https://github.com/amirhtavakoli94/bench_pmpscheduling`

3.2 Benchmark for optimization

We have generated new instances for the networks of Table 3.1 by using empirical consumption and tariff data. The electricity tariff is taken from Belgian spot market data, considering a reference period from 2007 to 2013. The demand profiles are drawn from real consumption considered in [38]. The raw consumption is sampled every 10 minutes for three years of data.

When mapped to a given network, we first sample and scale up the raw consumption data to the known average demand value for this network. Given that tanks and pumps are correctly dimensioned in these networks, an instance is often feasible if, on each period, the demand can be satisfied whatever the initial level of the tanks. To secure the feasibility of the demand profiles we generate, we solve a single-period optimization problem where the levels of the tanks are relaxed, and the objective function is the sink flow at demand nodes. In a second step, we solve the MILP

relaxation of the pump scheduling problem with $T = 12$. We reject the instance if unfeasibility is detected within a short time limit.

We summarize how we have generated demand profiles with a 2-hour time step for various networks from raw empirical data:

1. Since the raw data are sampled with a time step of 10 minutes, for each profile, we sum twelve consecutive demand values to obtain demand profiles with a 2-hour time step.
2. Each demand profile is then normalized by considering the root-mean-square of the total demand over one year.
3. We compute the maximum and the minimum demand values by solving the single-period problems:

   $$\overline{D}_{tj} = \max\{q^{\mathcal{J}}_{tj} | (q_t, h_t) \in \mathcal{E}(H_t, D_t, x_t), H_t \in [\underline{H}, \overline{H}], x_t \in \{0,1\}^{\mathcal{A}}\} \quad \forall t \in \mathcal{T}, j \in \mathcal{S}.$$

   Similarly, for the minimum value $\underline{D}_{tj}$.
4. We scale the demand values to be inside the computed bounds, in particular, we require $D_{tj} \in [1.05\underline{D}_{tj}, 0.95\overline{D}_{tj}]$, $\forall t = 1, ..., T-1, j \in \mathcal{S}$
5. We solve the MILP relaxation of the generated instance with $T = 12$ for ten seconds and keep the instance if a feasible solution is found. Otherwise, we add Gaussian noise to the demand values belonging to the first and third quartiles.

To visualize the variability of the raw consumption data, we applied a seasonal-trend decomposition [46] and derived the profiles in Figures 3.3 and 3.4. The second

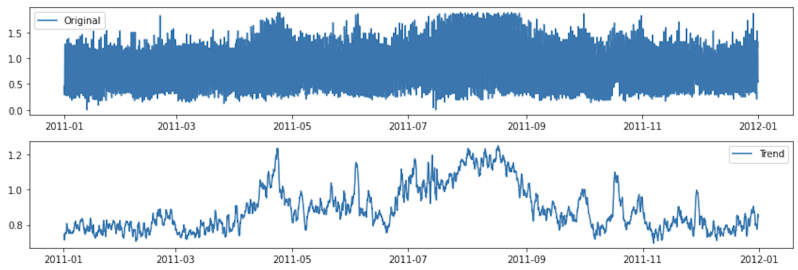

Figure 3.3: Raw consumption data and the corresponding trend over one year.

profile in Figure 3.3 shows well how the demand increases during the touristic period, in spring and even more during summer.

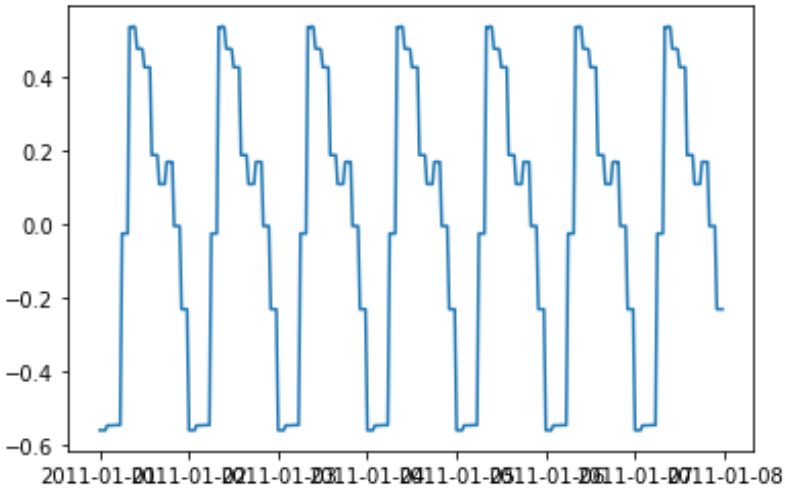

Figure 3.4: Daily seasonality over a week.

By looking at the daily seasonality over one week in Figure 3.4, we recognize the typical daily profile with a peak demand at midday.

Finally, we care to mention that we keep the time synchronization of the demand and tariff profiles. This allows us to match the seasonality and trend effects of the two profiles.

We also create instances for $T = 24$ and $T = 48$. To do that, we hold the value of the demand at a given time over two or four consecutive periods. Hence, we create piecewise constant demand profiles.

From generated data, we have selected six different instances for each network representing various trends in consumption and added to the benchmark available in `https://github.com/amirhtavakoli94/bench_pmpscheduling`

3.3 Historical dataset for supervised learning

To develop a learning model for the pump scheduling problem within a supervised learning framework, we need a training and test dataset, including inputs (i.e. feasible instances) and correct outputs (optimal solutions). Considering the *van Zyl* network with $T = 12$ as our testbed, we thus need to generate feasible instances and solve them to optimality.

We have generated instances (demand and tariff profiles) corresponding to six years of daily data by following the steps presented in the previous section. In the original *van Zyl* instance, the tanks are initially 95% full ($H_0 = 0.95\overline{H}$). The range of demand values satisfying the levels of the tanks at the very beginning and the end of the horizon are thus fairly restrictive. With respect to the benchmark in the literature, we have changed the initial level of the tank T6 in *van Zyl* network to 70% full at the initial stage of the scheduling problem.

To obtain optimal solutions, we solve each instance with our branch-and-check algorithm enhanced with preprocessing (described in Part II). If no solution is found within 5 minutes, we consider the instance infeasible and discard it from the dataset.

With this procedure, we generated a collection of 2113 unique daily observations (D, C, H), each given as $T = 12$ period profiles: the input features are the demand and tariff profiles $(D, C) \in \mathbb{R}^T \times \mathbb{R}^T$, and the target is the tank level profiles $H \in \mathbb{R}^{\dot{C}T}$ corresponding to an optimal solution of $(\mathcal{P})$ for input (D, C). Then, we split the dataset into 75 percent training, 15 percent validation, and a 10 percent test.

For finer temporal resolution $T = 24$ or $T = 48$, finding even feasible solutions is not always possible in a reasonable time. This limits the application of supervised learning. As mentioned in Part III, we integrate a scaling mechanism in our deep learning framework to tackle high temporal resolution instances from low temporal resolution data.

Finally, as a byproduct of this dataset, we generated an extended benchmark set by selecting 50 new daily instances with high variability. The benchmark sets (denoted VZ12, VZ24, and VZ48) are also available online.

Part II

Preprocessing for Global Optimization

Chapter 4

The Branch-and-Check Algorithm

Several algorithms are suggested in the literature to tackle general nonconvex MINLPs [47]. Spatial branch and bound [48] based on the Reformulation-Linearization Technique (RLT), and its variants (e.g., branch and reduce [49]), are usually the backbone of global optimization solvers. They solve an automatic MILP relaxation (the RLT relaxation) and recursively apply branching not only to integer variables, but also by partitioning the domain of continuous variables [50] involved in nonconvex constraints. Numerical experiments of direct applications to pump scheduling are, however, not encouraging [30, 31, 33].

These general-purpose solvers do not yet implement the advanced machinery (preprocessing, branching strategies, primal heuristics, etc.), which is available in the most sophisticated MILP solvers. They are de facto currently not able to detect and exploit some major characteristics of the pump scheduling problem. Firstly, the nonconvex constraints, arising from quadratic one-dimension functions, are easy to relax as convex/polyhedral constraints without introducing auxiliary variables, as RLT would do. Secondly, as described in Section 2.3, the pump scheduling problem has this very strong property: the model has exactly zero or one feasible solution for any complete instantiation of the binary variables (a binary schedule), and the solution is easy to compute by simulation (implementing the extended analysis algorithm EAA). As a consequence, a full enumeration of the binary schedules, either explicit or, more reasonably, implicit, leads to a global optimum (or the proof of unfeasibility). Several studies (e.g., [8, 9, 12, 42]) exploit this property by addressing the bilevel structure of the problem.

Among these works, Bonvin et al. [6] employ a branch-and-check approach to obtain an exact solution to problem ($\mathcal{P}$) through a MILP relaxation, solved with a

branch-and-bound, coupled with simulation to check the nonlinear constraints only at the integer nodes. We took their method as the starting point to derive a more efficient global optimization approach for the pump scheduling problem by strengthening the MILP relaxation in a new preprocessing step.

In this chapter, we first present the algorithm in [6] (for the case of fixed-speed pumps) and the associated MILP relaxation. We then devise two stronger MILP relaxations that we use in our preprocessing.

4.1 Branch-and-Check for Pump Scheduling

The algorithm in [6] starts with a MILP relaxation of reasonable size ($\mathcal{R}$) of the nonconvex MINLP ($\mathcal{P}$). The MILP relaxation ($\mathcal{R}$) is then solved with a slightly modified LP-branch-and-bound.

Each integer node of the search tree hence built, corresponds to a binary schedule $X \in \{0,1\}^{\mathcal{T}\times\mathcal{A}}$ that is feasible for the relaxation ($\mathcal{R}$), i.e., there exists an approximate flow/head equilibrium $(\overline{q}, \overline{h})$ such that $(X, \overline{q}, \overline{h})$ is a feasible solution of ($\mathcal{R}$). To check the feasibility of the complete assignment X regarding the actual nonlinear constraints and to compute the actual cost of the solution, the extended analysis algorithm (EAA) is run to solve the restricted nonconvex NLP ($\mathcal{P}(X)$). EAA computes progressively at each $t \in \mathcal{T}$, the flow-head equilibrium $(q_t, h_t) \in \mathcal{E}(D_t, H_t, X_t)$, then the resulting levels $H_{(t+1)j} = H_{tj} + \frac{q^{\mathcal{C}}_{tj}}{\sigma_j}$ in the tanks $j \in \dot{\mathcal{C}}$.

EAA stops prematurely if ever a tank capacity is exceeded at some iteration $\bar{t} \in \mathcal{T}$, i.e., $H_{\bar{t}+1} \notin [\underline{H}_{\bar{t}+1}, \overline{H}_{\bar{t}+1}]$. Since the schedule X is then unfeasible, the MILP solver is asked to prune the current search node. To this purpose, we add to the MILP relaxation a combinatorial nogood cut that invalidates the partial schedule until period $\bar{t}$:

$$\sum_{t=0}^{\bar{t}} \sum_{a:X_{ta}=1} (1 - x_{ta}) + \sum_{a:X_{ta}=0} x_{ta} \geq 1. \tag{4.1}$$

Otherwise, EAA has computed a feasible solution (X, q, h) for the problem ($\mathcal{P}$), and its cost $c(X, q, h)$ may be strictly greater than the cost $c(X, \bar{q}, \bar{h})$ of the relaxed MILP solution. This relaxed cost is thus not a valid upper bound, and the relaxed solution must also be ignored. Suppose the actual cost $c(X, q, h)$ does not improve upon the incumbent. In that case, the integer node is pruned, using the same combinatorial nogood cut as above, but covering the whole time horizon, i.e., taking $\bar{t} = T - 1$ in (4.1). Otherwise, the MILP solver is asked to set the actual solution associated

with the current integer node as the new incumbent and its cost as the best upper bound so far. The MILP solver then continues the search until closing the optimality gap between the best lower and upper bounds or if no open search node remains.

This algorithm is exact as the branch-and-bound explores a search space which initially contains all feasible schedules for $(\mathcal{R})$, thus all feasible schedules for $(\mathcal{P})$, and maintains, as the incumbent, a list of feasible schedules for $(\mathcal{P})$ in decreasing order of cost. The search space is progressively restricted by pruning only schedules that are either unfeasible for $(\mathcal{P})$ or feasible for $(\mathcal{P})$ but not strictly better than the incumbent, thus not optimal. Finally, the lower bound computed by the MILP solver at each node (continuous, integer feasible, or integer unfeasible) is valid for $(\mathcal{P})$.

Algorithm 2 presents the framework of this algorithm.

Algorithm 2 Branch-and-Check

Input $(\mathcal{R})$ a MILP relaxation
Output the optimal solution

solve $(\mathcal{R})$ with a branch-and-cut such that:
for each integer node $X \in \{0,1\}^{\mathcal{TA}}$ **do**:
    run algorithm EAA 1 with input X
    if x unfeasible **then**:
        get first violated period t
        add the linearized nogood cut $\sum_{t'=0}^{t} ||x_t - X_t||_1 \geq 1$
    else get solution (q, h)
        compute cost $c = c^0 x + c^1 q$
        if $c < UB$ **then**:
            update $UB = c$, $x^* = x$
        end if
        add a cut $z \geq c - c||x - X||_1$
    end if
end for

Communication with the MILP solver during the search is possible by using the callback functionality.

4.2 Discussion and enhancements

The above algorithm follows the bilevel structure of the pump scheduling problem, where the binary decisions appear at the upper level and the nonlinear constraints (i.e., the equilibrium) at the lower level. Unfeasibility and suboptimality at the lower level are communicated to the upper level through combinatorial nogood cuts, as in the branch-and-check algorithm described by [51] for general optimization problems (beyond the format of mathematical programming, including, e.g., the format logic programming).

Being able to identify the early part of a schedule that is unfeasible by itself to generate a smaller, then stronger nogood cut (4.1) (when $\bar{t} < T$) is a basic but effective enhancement of this method. This exploits the property that the tank heads are fixed at time $t = 0$ and then fixed at time $t = \bar{t} + 1$, in every solution extending a partial schedule from 0 to $\bar{t}$. Note that it is probably not trivial to improve this cut by identifying some minimum (as a set) unfeasible partial schedule without this property.

The nonconvexity of the lower level also prevents the generation of more efficient cuts, like Benders cuts (based on duality) in Benders decomposition or outer-approximation (OA) based cuts (based on supporting planes) in LP/NLP-branch-and-bound. Precisely, we could actually generate OA cuts when a point falls on the "convex side" of a nonlinear constraint by adding a supporting plane of the convex component of the constraint that separates the point. This strategy is employed by Tassef [52] both for the pipe sizing and pump scheduling problems. In our implementation, as in the original one [6], we do not generate these constraints dynamically, but only statically, when building the MILP relaxation ($\mathcal{R}$). This allows for better control of the size of the MILP relaxation, as well as of the numeric errors induced by the generation of constraints, which would be too close (considering also the required rounding). Moreover, violations of the nonlinear constraints occur more on the concave side (where the relaxation is weaker) than on the convex side, then it could be counterproductive to try to detect improbable violated OA cuts.

A framework similar to this branch-and-check algorithm is discussed in [39], also in the context of hydraulic network optimization, but for the pipe sizing problem. One major difference in the context of pump scheduling is that the cost of the relaxed solution does not correspond to the cost of the NLP solution. This requires managing the bounds, especially the incumbent, carefully beside the MILP solver.

This algorithm is not restricted to the condition that the relaxation ($\mathcal{R}$) is linear.

In particular, we could apply the exact same algorithm using a convex quadratic relaxation instead. However, convex quadratically-constrained MINLP solvers are not yet, at that time, as sophisticated as pure MILP solvers.

Finally, it is not possible to get a polyhedral relaxation of the nonlinear constraints (2.5) without knowing the domain of the involved variables. While the flow through a pump is bounded according to the operation of the pump, there are no bounds enforced on the flow through a pipe (except the lower bound 0 for unidirectional pipes). Bonvin et al. [6] implement a judicious bound tightening technique, considering the single time-step subproblem, as a preprocessing to derive the missing bounds, then to formulate the relaxation ($\mathcal{R}$). The bound preprocessing is a keystone of the algorithm as it is a determinant of the strength of the relaxation ($\mathcal{R}$), then of the efficiency of the branch-and-check. Their relaxation is presented in the next section, while their bound tightening technique is discussed in the following chapter, as well as our own contribution to improving these bounds.

4.3 MILP Relaxations of the Head-Flow Relation

There exist several MILP relaxations of ($\mathcal{P}$), which depend on the polyhedral relaxation chosen for the nonlinear relations $v_{ta} = \psi_a(q_{ta})$. We first present the MILP relaxation introduced in [6] to run their algorithm, then two other stronger but larger MILP relaxations we use in our preprocessing described in the next chapters.

4.3.1 Study of the resistance function

In our problem, we consider resistance functions of the following type:

$$\psi : q \in \mathbb{R} \mapsto Aq|q| + Bq + C$$

with $A, B, C \in \mathbb{R},\ A > 0$, and we look for either a continuous or an integer polyhedral relaxation of the set:

$$\Psi = \{(q, v) \in \mathbb{R}^2, h = \psi(q), q \in [\underline{Q}, \overline{Q}]\}$$

where $[\underline{Q}, \overline{Q}]$ are imposed or induced bounds on the flow value q.

Proposition 4.3.1. *Let $\psi(q) = Aq|q| + Bq + C$ with $A, B, C \in \mathbb{R},\ A > 0$, defined*

on $\mathbb{R}$, then ψ is continuously differentiable on $\mathbb{R}$, convex on $\mathbb{R}_+^*$ and concave on $\mathbb{R}_-^*$.

Proof. ψ is differentiable and its derivative $\psi'(q) = 2A|q| + B$ is continuous on $\mathbb{R}$. Its second derivative is defined on $\mathbb{R}^*$ by: $\psi''(q) = \begin{cases} 2A & \forall q > 0 \\ -2A & \forall q < 0 \end{cases}$ so it is positive (and ψ is convex) on R_+^* and it is negative (and ψ is concave) on R_+^-. □

4.3.2 Convex outer-approximation

The relaxation considered in [6] relies on computing an envelope on the resistance function, as depicted in Figure 4.1 and defined in the following proposition:

Proposition 4.3.2. *Let $\psi(q) = Aq|q| + Bq + C$ with $A, B, C \in \mathbb{R}$, $A > 0$, defined on $I = [\underline{Q}, \overline{Q}] \subseteq \mathbb{R}$, with $\underline{Q} < \overline{Q}$ then*

$$\psi \leq \begin{cases} f_{q^*}, \forall q^* \leq \min(\overline{Q}, \overline{Q}(1-\sqrt{2})) & \text{if } \underline{Q} < \overline{Q}(1-\sqrt{2}) \\ g_{[\underline{Q},\overline{Q}]}, & \text{otherwise} \end{cases}$$

$$\psi \geq \begin{cases} f_{q^*}, \forall q^* \geq \max(\underline{Q}, \underline{Q}(1-\sqrt{2})) & \text{if } \overline{Q} > \underline{Q}(1-\sqrt{2}) \\ g_{[\underline{Q},\overline{Q}]}, & \text{otherwise} \end{cases}$$

where $f_{q^*}(q) = \psi'(q^*)(q - q^*) + \psi(q^*)$ denotes the tangent of ψ for any $q^* \in \mathbb{R}$, and, g denotes the straight line intersecting ψ at $\underline{Q}$ and $\overline{Q}$.

Remark that if $\underline{Q} \geq 0$, the resistance function ψ is convex on its domain, and f_{q^*} are the supporting planes on the convex (under) side of ψ, while $g_{[\underline{Q},\overline{Q}]}$ is the chord on the concave (upper) side ψ. Then $\underline{\psi} = \max(f_{q^*})$ is an underestimator of ψ and $\overline{\psi} = g_{[\underline{Q},\overline{Q}]}$ is an overestimator of ψ. The case is antisymmetric when $\overline{Q} \leq 0$ (i.e. ψ is concave with $\underline{\psi} = g_{[\underline{Q},\overline{Q}]}$ and $\overline{\psi} = \max(f_{q^*})$). In these two cases, we can manage the size of the relaxed model regarding the relaxation gap.

Proposition 4.3.3. *If $\underline{Q} \geq 0$, then the minimum number of hyperplanes f_{q^*} required in the definition of the underestimator $\underline{\psi}$ to have a bounded difference between the projection of any feasible relaxed solution and the nonlinear quadratic function ψ less than a tolerance ϵ is $\lceil \frac{\overline{Q}-\underline{Q}}{2\sqrt{\frac{\epsilon}{A}}} \rceil$.*

Proof. mean value theorem. □

The proposition holds also if $\overline{Q} \leq 0$. When $\underline{Q} < 0 < \overline{Q}$, the difference between the projected relaxed solution and the nonlinear curve might exceed the defined tolerance ϵ on the interval $[\overline{Q}_l(1-\sqrt{2}), \underline{Q}_l(1-\sqrt{2})]$.

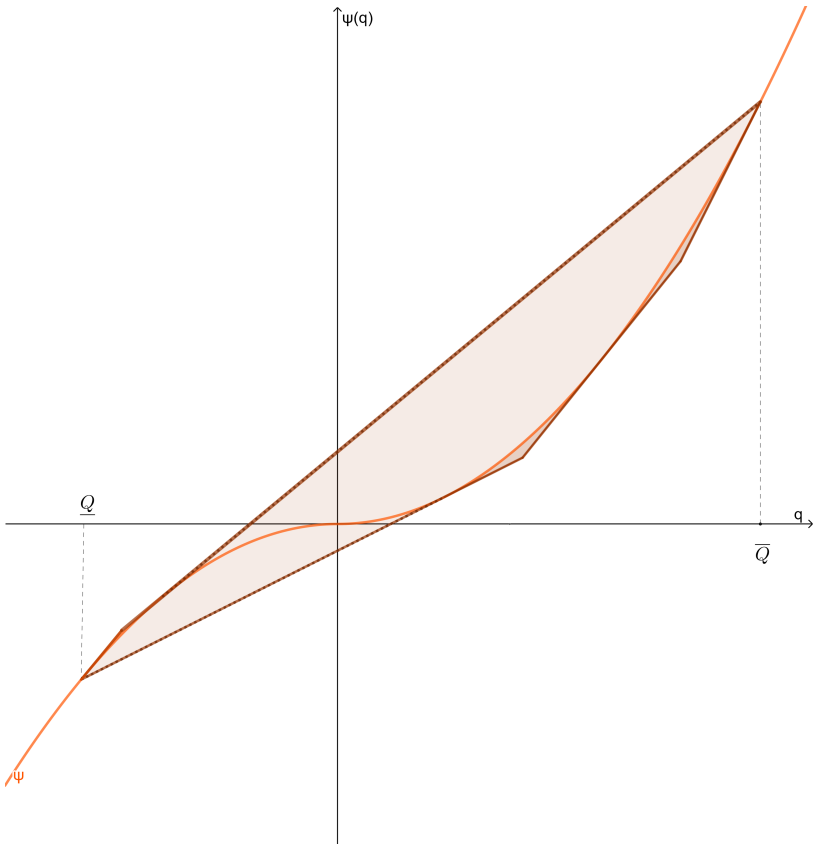

Figure 4.1: Outer approximation for nonlinear flow-head relationship.

The MILP relaxation ($\mathcal{R}^{OA}$) is then obtained by replacing Constraints (2.5) in ($\mathcal{P}$) with:

$$\underline{\psi_a}(q_{ta}) + \underline{V}_{ta}(1 - x_{ta}) \leq v_{ta} \leq \overline{\psi_a}(q_{ta}) + \overline{V}_{ta}(1 - x_{ta}) \quad \forall a \in \mathcal{A}, \forall t \in T.$$

The envelope of the resistance function being convex, this relaxation requires no additional binary variables.

4.3.3 Piecewise linear relaxation

The relaxation above requires no additional binary variables, but it can be loose on the "nonconvex side" of the resistance constraints. To tighten the relaxation on this side, we consider, as an alternative, a PWL envelope relaxation. Because this envelope is not convex, the relaxation requires auxiliary binary variables. By increasing the number of auxiliary binary variables, the PWL relaxation can mimic more accurately the exact nonlinear Ψ at the cost of complicating the relaxation.

Now, the maximum error ϵ can be controlled on both sides of function ψ, by defining the sufficient number of pieces. Furthermore, we can easily implement this relaxation for a given error by using modeling tools available in most MILP solvers, based on the definition of Special Ordered Set of Type 2 (SOS2) (see, e.g., [53]). While it is transparent for the modeler, this functionality introduces both linear constraints and binary variables for each piece of the approximation.

Consider the curve ψ restricted to $[\max(0, \underline{Q}), \overline{Q}]$, where it is convex. Then, a piecewise linear overestimator $\overline{\psi}(q)$ is defined by N segments linking consecutive points $(\bar{q}_p, \psi(q_p))_{p=1,...,N+1}$ distributed on the curve. The constraint $v \leq \psi(q)$ can then be relaxed as the nonlinear constraint $v \leq \overline{\psi}(q)$ or, equivalently, by $(q, v) \in PWO(\psi)$, where:

$$PWO(\psi) = \{(q, v) \in \mathbb{R}^2 : q = \sum_{p=1}^{N} \underline{\lambda}_p \bar{q}_p + \overline{\lambda}_p \bar{q}_{p+1} \tag{4.2a}$$

$$v \leq \sum_{p=1}^{N} \underline{\lambda}_p \psi(\bar{q}_p) + \overline{\lambda}_p \psi(\bar{q}_{p+1}) \tag{4.2b}$$

$$\sum_{p=1}^{n} z_p = 1 \tag{4.2c}$$

$$z_p = \underline{\lambda}_p + \overline{\lambda}_p \qquad \forall p \in \{1, ..., N\} \tag{4.2d}$$

$$\underline{\lambda}_p \geq 0, \overline{\lambda}_p \geq 0 \qquad \forall p \in \{1, ..., N\} \tag{4.2e}$$

$$z_p \in \{0, 1\} \qquad \forall p \in \{1, ..., N\} \ \}. \tag{4.2f}$$

When considering the concave part of curve ψ on the interval $[\underline{Q}, \min(0, \overline{Q})]$, we similarly derive a PWL underestimator $\overline{\psi}(q)$.

Proposition 4.3.4. *A convex function ψ can be relaxed by a set of overestimators such as $SOS2(\psi, N) \leq 0$ and $SOS2(\psi, N) \geq \epsilon$ while there is no need to introduce auxiliary binary variables z_i. In this situation, the nonlinear curve can be relaxed*

by the envelope constructed by such under and overestimators, and the number of minimum required pieces is controlled by ϵ *is* $\lceil \frac{\overline{Q}-\underline{Q}}{2\sqrt{\frac{\epsilon}{A}}} \rceil$

For bidirectional pipes, this minimum number of pieces might not be sufficient to ensure that no feasible solution is not eliminated during the construction of the envelope. Our proposal is initially to divide the head-flow function into two regions: one for $q \leq 0$ and $q \geq 0$. Therefore, for a positive flow, we have a convex curve, while for a negative flow, the function is concave.

4.3.4 Discretization and disjunctive formulation

The domain of the flow variable can be represented as consecutive intervals:

$$q \in [\underline{Q}, Q^{(1)}] \cup [Q^{(1)}, Q^{(2)}] \cup ... \vee [Q^{(N-1)}, \overline{Q}].$$

We can then construct an envelope of the resistance function ψ on each interval individually and relate the active intervals with additional binary variables $z^{(n)}$, $n = 1, ..., N$. The envelope is then modeled as follows:

$$q = \sum_{n=1}^{N} q^{(i)} \tag{4.3a}$$

$$v = \sum_{n=1}^{N} v^{(i)} \tag{4.3b}$$

$$Q^{n-1}.z^{(n)} \leq q^{(n)} \leq Q^{n}.z^{(n)} \ \forall n = 1..N \tag{4.3c}$$

$$\underline{\psi}^{(n)\epsilon}.q^{(n)} \leq v^{(n)} \leq \overline{\psi}^{(n)\epsilon}.q^{(n)} \ \forall n = 1, ..., N, \tag{4.3d}$$

$$\sum_{n=1}^{N} z^{(n)} = x \tag{4.3e}$$

$$z^{(n)} \in \{0, 1\} \ \forall n = 1..N \tag{4.3f}$$

where the over- and under-estimators in (4.3e) are generated for the given interval $[Q^{n-1}, Q^{n}]$ as in Proposition 4.3.2. Condition (4.3f) enforces that at most one of the possible regions is activated. In case of inactivity of the arc (i.e., $x = 0$), obviously, none of the regions can be active.

Using this envelope improves the MILP relaxation. However, this comes at the price of defining additional binary and continuous variables and numerous constraints.

As for the previous PWL relaxation, we will use this kind of relaxation only in some specific bound tightening techniques.

4.4 Strengthening the Relaxation

The branch-and-check algorithm discussed above relies on a branch-and-bound applied to the MILP relaxation ($\mathcal{R}$), which solves its LP relaxation iteratively at each node of the search tree. The efficiency of the algorithm then heavily depends on the strength and the size of the LP relaxation: too loose and the number of iterations increases; too large and the computational time of each iteration increases.

Off-the-shelf MILP branch-and-bound solvers already implement highly elaborate automatic techniques to improve LP relaxation. The main techniques are run in a preprocessing phase, before and during the resolution of the root node, and concern the domain reduction of variables and the generation of valid inequalities. These techniques exploit the knowledge of the integrality constraints to identify feasible regions of the LP relaxation that do not contain any feasible MILP solution. These regions are then excluded from the LP relaxation, either by tightening the variable bounds or by adding linear constraints.

In our MINLP branch-and-bound algorithm, intensifying the preprocessing phase with our own domain reduction and valid inequalities generation techniques is of paramount importance.

First, and above all, we build the MILP relaxation itself directly from the known bounds of the variables involved in the nonlinear constraints of our MINLP model, namely the flow and head variables defining the resistance relation $v_{ta} = \psi_a(q_{ta})$ for each period $t \in \mathcal{T}$ and each arc $a \in \mathcal{A}$. Tightening the bounds of these variables specifically is then a priority. Indeed, by reducing the domain of the flow variables (i.e., the length $\overline{Q}_{ta} - \underline{Q}_{ta}$), we reduce the number of supporting planes required to relax the convex side of the constraints for a chosen tolerance value, and we also reduce the relaxed area on the concave side.

Furthermore, we can enforce some propagation effect: tightened bounds for one variable may help to infer bound tightening for another variable. In the context of pump scheduling, the feasible flow-head equilibrium on a period $t \in \mathcal{T}$ depends on the levels of the tanks at the beginning of the period H_t and, in turn, it impacts the levels at the end of the period H_{t+1}. We thus propose to exploit this interdependency between variables q, h, H to force the propagation effect. Also, conditional bounds

appear as big-M values in the constraints (2.5) linearizing the logical relations between the status of an arc (active/inactive) and the flow/head through the arc. Considering the interdependency between variables q, h, x during the bound-tightening process allows for improving those big-M values and, thus, the LP relaxation. Our bound-tightening algorithm is presented in Chapter 5.

On another note, as our algorithm directly acts on the MILP branch-and-bound (particularly when adding the nogood cuts), then the automatic preprocessing (in particular, the valid inequalities generation) is restrained by the MILP solver to prevent conflicts or numerical issues. More importantly, the underlying MILP solver is not aware of the original nonlinear constraints. It then might overlook some deductions (which are based only on the integrality constraints). In branch-and-check, shrinking the LP relaxation of the nonlinear constraints implies fewer integer nodes to check. Finally, because our algorithm is dedicated to the pump scheduling problem, we can even infer reductions from our own knowledge of the practical problem for a given network topology or a given demand/tariff pattern. This knowledge is particularly useful for detecting impossible combinations of active elements. We are then able either to fix some binary variables or generate valid linear equalities on these binary variables to exclude these invalid combinations. Our valid inequalities generation process is presented in Chapter 6.

Chapter 5

Bound Tightening

In mathematical programming, bound tightening refers to techniques for reducing the (interval) domains of variables in a model without altering the optimal value: the reduced domain product must contain at least one, if not all, the original optimal solutions (optimality-oriented techniques), or even all original feasible solutions (feasibility-oriented techniques).

Bound-tightening is a major component of preprocessing for MILP branch-and-bound. By improving the MILP relaxation, it reduces the size of the search tree: fixing binary variables directly reduces the number of choices and branches, and it mitigates the effect of the relaxation of the corresponding integrality constraints while adjusting the continuous domains increases the LP lower bounds and the possibility to fathom search nodes. Furthermore, modifying the variable bounds in a MILP does not affect the size and the structure of the LP relaxation (by opposition to cut generation), nor does it affect the processing time of each search node.

Bound-tightening is even doubly important in our context as we build the MILP relaxation specifically from the bounds of the variables involved in the nonlinear constraints: both for approximating functions ψ_a (with piecewise linear functions) and for linearizing the disjunctive relations (with big-M constraints). Figure 5.1 illustrates the impact of the flow bounds on the strength of the OA relaxation. For the same OA tolerance ϵ, and with the improved bounds, the feasible space of the OA relaxation on the concave side of the curve is drastically shrunk, and the number of planes defining the convex side is also reduced. This implies fewer constraints in the MILP relaxation. The impact on the PWL relaxation is even more visible as it also limits the number of auxiliary binary variables.

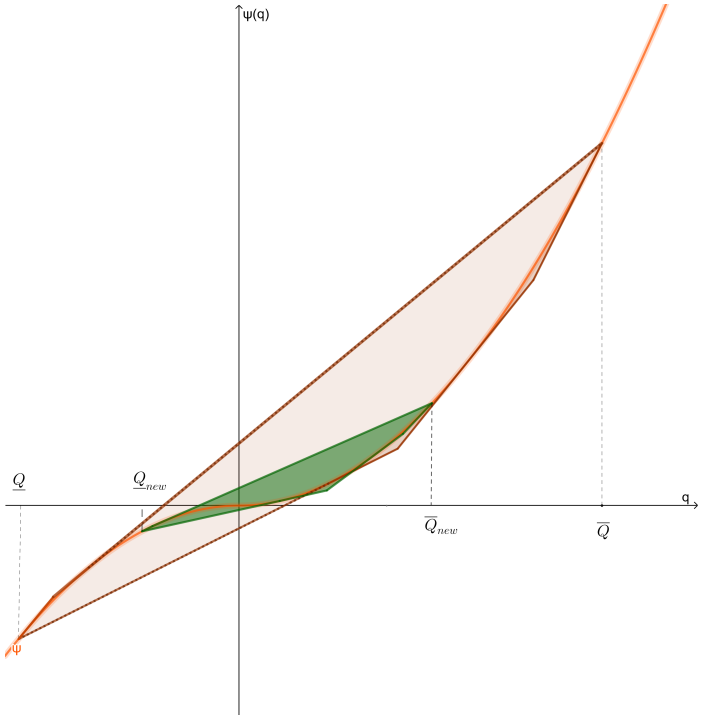

Figure 5.1: A nonlinear constraint (orange curve) and the OA relaxations computed with the initial bounds of the flow variables (orange space), and with improved bounds (green space).

In the original algorithm [6], the authors propose a preprocessing phase to tighten the bounds on the flow/head variables before formulating the MILP (OA) relaxation. They rely on Optimization-Based Bound Tightening (OBBT) techniques [54, 55], i.e., they derive bound improvements by solving a certain mathematical programming relaxation of the problem. We call this kind of relaxation, the OBBT relaxation or OBBT problem, and we denote $(\mathcal{O})$ the optimization problem and $\mathcal{O}$ its feasible set by opposition to the MILP and LP relaxations used in the branch-and-check algorithm, denoted, respectively, $(\mathcal{R})$ and $(\overline{\mathcal{R}})$ with feasible sets $\mathcal{R}$ and $\overline{\mathcal{R}}$.

In this chapter, we first recall the principle of OBBT and its application in [6] using a steady-state relaxation of the pump scheduling problem. We then investigate ways to improve the process. In particular, we index the relaxation to each period independently, then focus on the bounds of the time-coupling variables, i.e., the level of the tanks. Finally, we compute bounds conditionally to the status of related network elements. These conditional bounds may be enforced as big-M constraints in the MILP relaxation of branch-and-check. Finally, we describe how the procedure is iterated to propagate any bound improvement by updating the subsequent OBBT relaxations.

We assume that the demand scenario in $(\mathcal{P})$ is standard and adapted to the network dimension. In other words, we assume that, for all $t \in \mathcal{T}$, $D_t \in [\underline{D}, \overline{D}] \subseteq \mathbb{R}^S$, an absolute range of possible demand configurations, such that any feasible solution of $(\mathcal{P})$ naturally satisfies also the bounds on the flow values defined by the operational domain of the pumps or by the direction of the pipes, as well as the minimum head values at the service nodes. The design of the network makes that these bounds are much less constraining than the capacity of the tanks. Hence, we consider the following initial domains for the variables in $(\mathcal{P})$:

$$q_{ta} \in [\underline{Q_a}, \overline{Q_a}] \subseteq \mathbb{R}, v_{ta} \in [\underline{V_a}, \overline{V_a}] \subseteq \mathbb{R} \ \forall a \in \mathcal{A}, t \in \mathcal{T}.$$

We can thus linearize the conditional constraints (2.13) and (2.14), as big-M constraints

$$\underline{Q}_a x_a \leq q_a \leq \overline{Q}_a x_a, \qquad \forall a \in \mathcal{A}, \tag{5.1}$$

$$\underline{\psi_a}(q_a)x_a + \underline{V}_a(1 - x_a) \leq v_a \leq \overline{\psi_a}(q_a)x_a + \overline{V}_a(1 - x_a), \qquad \forall a \in \mathcal{A}, \tag{5.2}$$

to formulate the relaxation $\tilde{\mathcal{E}} \supseteq \mathcal{E}$, then get the relaxation $\mathcal{R} \supseteq \mathcal{P}$ by replacing (2.17) with:

$$(q_t, h_t) \in \tilde{\mathcal{E}}(H_t, D_t, x_t), \qquad \forall t \in \mathcal{T}. \tag{5.3}$$

Also, we denote $\{\underline{X}, \overline{X}\} \subseteq \{0, 1\}^{\mathcal{T}\mathcal{A}}$ the initial bounds on the x variables. Note that $\underline{X}_{ta} = \overline{X}_{ta} = 1$ when arc $a \in \mathcal{A}$ cannot be set inactive, in particular, when a is a simple pipe, and $\underline{X}_{ta} = \overline{X}_{ta} = 0$ when arc $a \in \mathcal{A}$ cannot be set active (e.g., for maintenance) at period $t \in \mathcal{T}$. These initial conditions are represented in Constraints (2.8) in $(\mathcal{P})$.

5.1 Optimization-based Bound Tightening

5.1.1 Principle

There are mostly two approaches to computing valid lower and upper bounds of a variable in the mathematical program $(\mathcal{P})$: by local inference (as for domain consistency and constraint propagation in constraint programming) or by minimizing/-maximizing the variable over a set $\mathcal{O}$ which includes either all the optimal solutions (optimality-oriented) or all the feasible solutions of $(\mathcal{P})$ (feasibility-oriented). The

second approach, which can be seen as global inference, is called Optimization-based Bound Tightening (OBBT). The efficacy and efficiency of OBBT are highly dependent on the choice of the relaxation $\mathcal{O}$: too tight and solving one OBBT problem ($\mathcal{O}$) may be as hard as solving ($\mathcal{P}$), too loose and it provides no new information to the MILP relaxation ($\mathcal{R}$) that the optimization process cannot infer alone. Therefore, we should find a trade-off between these two to keep the computational cost of preprocessing under control. At the same time, the ultimate relaxation gleaned from these bound-tightening techniques improves the runtime of the branch-and-check process.

5.1.2 Steady-state relaxation with floating demand

In [6], the authors propose to preprocess the branch-and-check algorithm and derive the MILP relaxation ($\mathcal{R}$) by applying feasibility-oriented OBBT to the variables involved in the nonlinear constraints (2.13) and (2.14), namely the flow and head loss variables q and v. In any feasible solution of ($\mathcal{P}$), the values of q and v at a given time period are entirely determined by the equilibrium constraints $\mathcal{E}$ associated with this period. In other words, by relaxing the time-coupling constraints in ($\mathcal{P}$), we may consider a relaxation in which the scope is restricted to a single t:

$$\mathcal{E}' = \{(x, q, h) : (q, h) \in \mathcal{E}(H, D, x), x \in X \in \{0,1\}^{\mathcal{A}}, H \in [\underline{H}, \overline{H}], D \in [\underline{D}, \overline{D}]\},$$

where $[\underline{H}, \overline{H}] \subseteq \mathbb{R}^{\mathcal{C}}$ denotes, as before, the range of tank capacities, and $[\underline{D}, \overline{D}] \subseteq \mathbb{R}^{\mathcal{S}}$ includes any possible (static) demand scenario for this network, in particular the demand scenario of the instance under scrutiny:

$$D_{tj} \in [\underline{D}_j, \overline{D}_j] \ \forall j \in \mathcal{S}, \ \forall t \in \mathcal{T}.$$

In practice, these values are obtained from the historical data of the given water network.

For these ranges, we thus have the following relation:

$$(x, q, h) \in \mathcal{P} \implies (x_t, q_t, h_t) \in \mathcal{E}', \ \forall t \in \mathcal{T},$$

then the bounds obtained by optimizing over this set are valid for all periods and even for all instances of the pump scheduling problem associated with a given network. For example, minimizing a flow variable q_a over this set provides a lower bound for all

the flow variables q_{at}, $t \in \mathcal{T}$ in $(\mathcal{P})$:

$$(x, q, h) \in \mathcal{P} \implies q_{ta} \geq \underline{Q}'_a = \min\{q_a : (x, q, h) \in \mathcal{E}'\} \; \forall t \in \mathcal{T}, \forall a \in \mathcal{A}.$$

The number of decision variables in $\mathcal{E}'$ is smaller than in $\mathcal{P}$ as we only consider a single period. Still, optimizing over this exact nonconvex MINLP might be expensive (for a comprehensive view on the single-time pump scheduling problem with fixed demand, the reader can refer to [33]). However, this makes it possible to consider instead very tight relaxations. Obviously, the two previous formulae remain true when replacing $\mathcal{E}'$ by any relaxation $\mathcal{O} \supseteq \mathcal{E}'$. In our experiments, we considered the piecewise linear MILP relaxation with a fine discretization step $\mathcal{O} = \mathrm{PWL}(\mathcal{E}')$ built from the initial domains of the variables.

Finally, the bounds on the flow and head loss variables only appear as big-M values in the relaxation $(\mathcal{R})$ for the conditional constraints (2.13) and (2.14), and they must be computed accordingly to the possible status on the associated arc. Indeed, the arc flow bounds $\underline{Q}_{at}, \overline{Q}_{at}$ are active in constraints (2.13) only if arc a is active at time t ($x_{at} = 1$), otherwise the flow q_{at} is null. Similarly, the head-loss bounds $\underline{V}_{at}, \overline{V}_{at}$ are active in constraint (2.14) only if arc a is inactive at time t ($x_{at} = 0$), otherwise the head loss v_{at} is between $\psi(\underline{Q}_{at})$ and $\psi(\overline{Q}_{at})$.

We thus enforce the condition $x_a = 1$ (resp. $x_a = 0$) by adding it to $\mathcal{O}$ when computing the flow bounds (resp. head loss bounds). It happens that this condition makes the OBBT problem infeasible. For example, if, when computing the flow bounds, we observe that $\mathcal{O}^{x_a=1} = \mathcal{O} \cap \{(x, q, h) : x_a = 1\} = \emptyset$, this implies that arc a can never be active, i.e., that $x_{at} = 0$ is a valid constraint of $(\mathcal{P})$ for any $t \in \mathcal{T}$. This principle, known as probing [56], allows reducing both the big-M values and the number of binary variables in $\mathcal{P}$ (thus in $\mathcal{R}$).

The preprocessing is detailed in Algorithm 3. Here, τ denotes the tolerance for solving the OBBT subproblem (e.g., $|L_a - \min\{q_a : (x, q, h) \in \mathcal{O}^{x_a=1}\}| < \tau$), and we consider this safety margin to update the bounds (e.g., $\underline{Q}_a := \max(\underline{Q}_a, L_a - \tau)$). Ideally, a larger tolerance value should be taken when solving the branch-and-check relaxation $(\mathcal{R})$ built from these bounds. Note that improving flow bounds $\underline{Q}_a$, $\overline{Q}_a$ at a given iteration could be propagated to tighten the relaxation $\mathcal{O}$ (by computing new approximations $\underline{\psi}$, $\overline{\psi}$) in the next iterations. However, when considering a tight relaxation from the outset, we should not expect much improvement in the bound computation. On the contrary, keeping the iterations independent allows for the parallelization of this algorithm.

Algorithm 3 static OBBT for the arc variables

Input $\mathcal{O} \supseteq \mathcal{E}'$ a single-period relaxation of $\mathcal{P}$, optimization tolerance $\tau > 0$

Output $[\underline{Q}, \overline{Q}] \subseteq \mathbb{R}^{\mathcal{A}}$, $[\underline{V}, \overline{V}] \subseteq \mathbb{R}^{\mathcal{A}}$, $\{\underline{X}, \overline{X}\} \subseteq \{0,1\}^{\mathcal{A}}$

for $a \in \mathcal{A}, \overline{X}_a = 1$ **do**:
    compute $L_a := \min\{q_a : (x,q,h) \in \mathcal{O}^{x_a=1}\}$, $U_a := \max\{q_a : (x,q,h) \in \mathcal{O}^{x_a=1}\}$ with tolerance τ
    if infeasible **then**:
        $\overline{X}_a = 0$
    else update $\underline{Q}_a := \max(\underline{Q}_a, L_a - \tau)$, $\overline{Q}_a := \min(\overline{Q}_a, U_a + \tau)$
    end if
end for
for $a \in \mathcal{A}, \underline{X}_a = 0$ **do**:
    compute $L_a := \min\{v_a : (x,q,h) \in \mathcal{O}^{x_a=0}\}$, $U_a := \max\{v_a : (x,q,h) \in \mathcal{O}^{x_a=0}\}$ with tolerance τ
    if infeasible **then**:
        $\underline{X}_a = 1$
    else update $\underline{V}_a := \max(\underline{V}_a, L_a - \tau)$, $\overline{V}_a := \min(\overline{V}_a, U_a + \tau)$
    end if
end for

5.1.3 Steady-state relaxation with fixed demand profiles

The previous preprocessing is fast, as it is both time-independent, instance-independent and can be fully parallelized: it consists of solving at most four small MILPs for every arc, independently, and it has to be done once for all for a given network and does not need to be run for every instance. Still, probing and strong bound reduction are unlikely to occur without considering the exact demand scenario for a given instance ($\mathcal{P}$). Given $D \in \mathbb{R}^{\mathcal{T}\mathcal{S}}$, we can actually apply Algorithm 3 to each time $t \in \mathcal{T}$ after enforcing $D = D_t$ in the single-time model $\mathcal{E}'$, then using the OBBT relaxation:

$$\mathcal{O}_t \supseteq \{(x,q,h) : (q,h) \in \mathcal{E}(H, D_t, x), x \in X \subseteq \{0,1\}^{\mathcal{A}}, H \in [\underline{H}, \overline{H}]\}$$

The nature and size of the OBBT problems do not change, and the iterations remain independent. Still, their number is multiplied by T. One trivial improvement is aggregating the time instants with the same demand values.

Algorithm 4 dynamic OBBT for the arc variables

Input $(\mathcal{O}_t)_{t\in\mathcal{T}}$ single-period relaxations of $(\mathcal{P})$, optimization tolerance $\tau > 0$
Output $[\underline{Q}, \overline{Q}] \subseteq \mathbb{R}^{\mathcal{T}\mathcal{A}}$, $[\underline{V}, \overline{V}] \subseteq \mathbb{R}^{\mathcal{T}\mathcal{A}}$, $\{\underline{X}, \overline{X}\} \subseteq \{0,1\}^{\mathcal{T}\mathcal{A}}$

1: **for** $t \in \mathcal{T}$ **do**:
2: call Algorithm 3 with input $\mathcal{O} := \mathcal{O}_t$ and τ
3: update $[\underline{Q}_t, \overline{Q}_t]$, $[\underline{V}_t, \overline{V}_t]$, $\{\underline{X}_t, \overline{X}_t\}$
4: **end for**

5.1.4 Multi-period relaxation for the state variables

Besides flow and head loss variables q and v, which appear in the nonlinear constraints in $(\mathcal{P})$, the tank level variables H which appear in the time-coupling constraints (2.18) are also serious candidates for bound-tightening. Indeed, the main feasibility issue in $(\mathcal{P})$ comes from these time-coupling constraints and tank capacities, thus, tightening these bounds could directly help to fathom search nodes (including integer nodes) in the branch-and-bound.

Furthermore, the solution of the equilibrium problems $\mathcal{E}(H_t, D_t, x_t)$ is fully determined by the tank levels H_t, which reveal all scheduling activities so far $x_0, \dots, x_{t-1}$. Tightening the bounds $[\underline{H}_t, \overline{H}_t] \in \mathbb{R}^{\mathcal{C}}$ could thus also have a great impact on the steady-state relaxations considered in the previous OBBT subproblems:

$$\mathcal{O}_t \supseteq \{(x, q, h) : (q, h) \in \mathcal{E}(H, D_t, x), x \in X \subseteq \{0,1\}^{\mathcal{A}}, H \in [\underline{H}_t, \overline{H}_t]\},$$

and could then contribute to the whole preprocessing algorithm. Hence, we propose to apply OBBT also to the variables H and to iterate Algorithm 4 so that any bound improvement is propagated by updating the relaxation $\mathcal{O}$ in the subsequent OBBT problems.

Bounding the tank inflow. First, remark that the tank level variables H_{tj} are uniquely determined by the (fixed) initial tank levels H_{0j} and by the tank inflow values $q_{0j}, \dots, q_{(t-1)j}$ according to (2.18). Since these inflow variables are determined by the equilibrium problem for each period (as the sum of incident arc flows), we can also apply OBBT on the single-period relaxations, as previously, to handle their bounds. Because inflows result as the sum of positive and negative flows, this direct application of OBBT to the tank inflow variables is not redundant with OBBT applied to the arc flow variables. In practice, the bounds $[\underline{Q^{\mathcal{C}}}_{tj}, \overline{Q^{\mathcal{C}}}_{tj}]$ are significantly tighter than summing the limits on the incident flow variables $[\sum_{a\in\mathcal{A}} \min(E_{aj}\underline{Q}_{ta}, E_{aj}\overline{Q}_{ta}), \sum_{a\in\mathcal{A}} \max(E_{aj}\underline{Q}_{ta}, E_{aj}\overline{Q}_{ta})]$, typically when both ingoing

arcs ($E_{aj} = -1$) and outgoing arcs ($E_{aj} = +1$) are incident to tank j. Algorithm 5 illustrates the bounding procedure for the tank inflow.

Algorithm 5 dynamic OBBT for the arc and tank variables

Input $(\mathcal{O}_t)_{t\in\mathcal{T}}$ single-period relaxations of $\mathcal{P}$, optimization tolerance $\tau > 0$
Output $[\underline{Q}, \overline{Q}] \subseteq \mathbb{R}^{\mathcal{T}\mathcal{A}}$, $[\underline{V}, \overline{V}] \subseteq \mathbb{R}^{\mathcal{T}\mathcal{A}}$, $[\underline{Q^{\mathcal{C}}}, \overline{Q}^{\mathcal{C}}] \subseteq \mathbb{R}^{\mathcal{T}\mathcal{C}}$ $\{\underline{X}, \overline{X}\} \subseteq \{0,1\}^{\mathcal{T}\mathcal{A}}$

1: **for** $t \in \mathcal{T}$ **do**:
2: call Algorithm 3 with input $\mathcal{O} := \mathcal{O}_t$ and τ
3: update $[\underline{Q}_t, \overline{Q}_t]$, $[\underline{V}_t, \overline{V}_t]$, $\{\underline{X}_t, \overline{X}_t\}$
4: **for** $j \in \dot{\mathcal{C}}$ **do**:
5: compute $L := \min\{q^{\mathcal{C}}_{tj} : (x, q, h) \in \mathcal{O}_t\}$, $U := \max\{q^{\mathcal{C}}_{tj} : (x, q, h) \in \mathcal{O}_t\}$ with tolerance τ
6: update $\underline{Q}^{\mathcal{C}}_{tj} := \max(\underline{Q}^{\mathcal{C}}_{tj}, L - \tau)$, $\overline{Q}^{\mathcal{C}}_{tj} := \min(\overline{Q}^{\mathcal{C}}_{tj}, U + \tau)$
7: **end for**
8: **end for**

Bounding the tank head. Still, the steady-state relaxation is not relevant for applying OBBT to variables H_t since they depend on the decisions made at $t-1$ and before. In order to capture this dependency, we propose to consider multi-period relaxations $(\mathcal{M}_{[t_s,t_e]})$ of $(\mathcal{P})$, restricted to some time interval $[t_s, t_e] \subseteq \mathcal{T}$, and including all the time-coupling constraints on this interval:

$$\mathcal{M}_{[t_s,t_e]} \supseteq \{(q_t, h_t) \in \tilde{\mathcal{E}}(H_t, D_t, x_t), \qquad \forall t \in [t_s, t_e] \tag{5.4}$$
$$q^{\mathcal{C}}_{tj} = \sigma_j(H_{(t+1)j} - H_{tj}), \qquad \forall j \in \dot{\mathcal{C}}, \forall t \in [t_s, t_e] \tag{5.5}$$
$$\underline{H}_{tj} \leq H_{tj} \leq \overline{H}_{tj}, \qquad \forall j \in \mathcal{C}, \forall t \in [t_s, t_e + 1] \tag{5.6}$$
$$x \in \{0,1\}^{[t_s,t_e]\times\mathcal{A}}\}. \tag{5.7}$$

We also denote the continuous relaxation of the mentioned set as $\bar{\mathcal{M}}$ by replacing the constraint $x \in \{0,1\}$ with $x \in [0,1]$.

To compute bounds for $H_{t'j}$ for a given tank $j \in \dot{\mathcal{C}}$ and time $t' \in \overline{T}$, we need to capture the trajectory of the tank level from the start of the schedule to time t', then consider the OBBT relaxation $\mathcal{M}_{[0,t'-1]}$. Remember that the relaxation $\tilde{\mathcal{E}}$ includes constraints (5.1) and (5.2), thus $\mathcal{M}_{[0,t'-1]}$ relies on the bounds $[\underline{Q}_t, \overline{Q}_t]$ and $[\underline{V}_t, \overline{V}_t]$ for all $t \in [0, t'-1]$. Therefore, obtaining tighter bounds for H_t may propagate (by updating $\mathcal{O}_t$) tighter bounds over q_t, v_t, x_t as well, which in turn may propagate (by updating $\mathcal{M}_{[0,t]}$) on H_{t+1}. Algorithm 6 catches this propagation effect. However, after several iterations, the improvement following each iteration may become a blunt and

ineffective tool. If the average improvement is smaller than a certain amount, we can stop the bound tightening. We will discuss this in numerical experiment chapter 7.

Algorithm 6 OBBT propagation between arc and tank variables

Input $(\mathcal{O}_t)_{t\in\mathcal{T}}$, $(\mathcal{M}_{[0,t]})_{t\in\mathcal{T}}$ single and multiple period relaxations of $\mathcal{P}$, optimization tolerance $\tau > 0$

Output $[\underline{Q}, \overline{Q}] \subseteq \mathbb{R}^{\mathcal{TA}}$, $[\underline{V}, \overline{V}] \subseteq \mathbb{R}^{\mathcal{TA}}$, $[\underline{Q}^{\mathcal{C}}, \overline{Q}^{\mathcal{C}}] \subseteq \mathbb{R}^{\mathcal{T}\dot{\mathcal{C}}}$, $[\underline{H}, \overline{H}] \subseteq \mathbb{R}^{\overline{\mathcal{T}}R}$, $\{\underline{X}, \overline{X}\} \subseteq \{0,1\}^{\mathcal{TA}}$

call Algorithm 5 with input $(\mathcal{O}_t)_{t\in\mathcal{T}}$ and τ
while sufficient bound improvement **do**:
    for $t \in \mathcal{T}$, $j \in \dot{\mathcal{C}}$ **do**:
        update the bounds in $\mathcal{M}_{[0,t]}$
        compute $L = \min\{h_{(t+1)j} : (x, q, h) \in \mathcal{M}_{[0,t]}\}$, $U = \max\{h_{(t+1)j} : (x, q, h) \in \mathcal{M}_{[0,t]}\}$, with tolerance τ
        update $\underline{H}_{(t+1)j} := \max(\underline{H}_{(t+1)j}, L - \tau)$, $\overline{H}_{(t+1)j} := \min(\overline{H}_{(t+1)j}, U + \tau)$
    end for
    update the bounds in $(\mathcal{O}_t)_{t\in\mathcal{T}}$
    call Algorithm 5 with input $(\mathcal{O}_t)_{t\in\mathcal{T}}$ and τ
end while

The multi-period OBBT problems $(\mathcal{M})$ might take a significant time to be solved when they cover a long time interval. In addition, the inference capacity decreases between temporally distant events. Hence, instead of variable-length intervals $[0, t]$, we also considered in our experiments on the largest instances, multi-period relaxations of fixed length $\mathcal{M}_{[t-\delta,t]}$. Furthermore, while we enforce tight approximations of the nonlinear constraints in the single-period relaxation used for computing the bounds on the static variables – including the bounds on the inflow variables which are enforced in $\mathcal{M}$ – we can employ a much looser relaxation of these nonlinear relations when computing the bounds on the tank levels. Similarly, the integrality constraints (5.7) in $\mathcal{M}_{[t-h,t]}$ do not affect much the resulting bounds, but their number increases linearly with h, and the computational time for solving the OBBT relaxation increases exponentially. Therefore, we propose relaxing these integrality constraints in $\mathcal{M}$ when considering large time intervals and large networks.

Bounding the difference of tank heads. By taking the reasoning further, the flow passing through an arc depends on the difference of heads upstream and downstream. In response to this fact, we apply bound tightening to the difference of

heads in pairs of tanks $j, j' \in \mathcal{C}$ connected to each other by a sequence of arcs (with no other reservoirs in-between). To this purpose, we also employ the multi-period relaxation, for $t \in \mathcal{T} \setminus \{0\}$: $\underline{H}_{tjj'} = \min\{H_{tj} - H_{tj'} : (x, q, h) \in \mathcal{M}_{[t-\delta,t-1]}\}$, $\overline{H}_{tjj'} = \max\{H_{tj} - H_{tj'} : (x, q, h) \in \mathcal{M}_{[t-\delta,t-1]}\}$.

These bounds are then enforced in the branch-and-bound relaxation ($\mathcal{R}$) with the additional constraints:

$$\underline{H}_{tjj'} \leq H_{tj} - H_{tj'} \leq \overline{H}_{tjj'}.$$

5.1.5 Probing on related pairs network elements

We can also apply probing to related network elements. For instance, we can consider the situation when some pump or valve $a \in \mathcal{A}$ regulates the filling/draining of a tank $j \in \dot{\mathcal{C}}$. We can compute the bounds on the tank inflow variables q_j depending on the status $x_a \in \{0, 1\}$, then enforce the following lower bounding constraint (and similarly for the upper bound):

$$q^{\mathcal{C}}_{tj} \geq \underline{Q}^{a1}_{tj} x_{(t-1)a} + \underline{Q}^{a0}_{tj}(1 - x_{(t-1)a}),$$

where $\underline{Q}^{a1}_{tj}, \underline{Q}^{a0}_{tj}$ are the lower bounds on inflow in the tank j computed conditionally to the status of the pump a at the period $t \in \mathcal{T}$. Note that for one of these two bounds (say $\underline{Q}^{ab}_{tj}$ with $b \in \{0, 1\}$), we can use the bound $\underline{Q}^{\mathcal{C}}_{tj}$ computed in Algorithm 6 if $x_{(t-1)a} = b$ in the associated OBBT solution[1]. In this case, only the second OBBT problem has to be solved: $\underline{Q}^{a(1-b)}_{tj} = \min\{q_{tj} : (x_t, q_t, h_t) \in \mathcal{O}_t, x_{ta} = 1 - b\}$.

Obviously, the related elements to inspect should be chosen thoroughly so as not to multiply unfruitful probes. The resulting constraint should be added to the branch-and-bound relaxation ($\mathcal{R}$) only if probing leads to a significant bound improvement, e.g., if the gap $\underline{Q}^{a(1-b)}_{jt} - \underline{Q}^{\mathcal{C}}_{jt}$ exceeds a given value.

5.1.6 Extended probing and disjunctive programming

The probing approach is not restricted to pairs of variables. We can also consider combinations of several network elements and their possible configurations on one or several periods in order to examine their impact on the bounds of a related variable

[1]This happens if $x_{ta} \in \{0, 1\}$ is enforced in the selected OBBT relaxation

in $\mathcal{P}$. The domain of the variable can thus be restricted to the union of the resulting intervals. We can linearize this information by applying disjunctive programming techniques via big-M or convex-hull formulations [57, 58].

Application to state variables. Consider, for example, extending the previous constraint by relating the tank inflow $q^{\mathcal{C}}_{tj}$, or equivalently the tank level H_{tj}, with a pump $a \in \mathcal{A}$ standing upstream and its status on the two consecutive time-steps t and $t+1$. Assume that we are able to derive a high lower bound $\underline{H}^{10}_{tj} >> \underline{H}_{tj}$ for H_{tj} when enforcing both $x_{ta} = 1$ and $x_{(t+1)a} = 0$ in the OBBT relaxation, e.g., $\underline{H}^{10}_{tj} = \min\{h_{tj} : (x, q, h)_{[t,t+1]} \in \mathcal{M}_{[t,t+1]}, x_{ta} = 1, x_{(t+1)a} = 0\}$. Then, we can enforce this bound in $(\mathcal{R})$ through the following linear constraint:

$$H_{tj} \geq \underline{H}^{10}_{tj} \tilde{x}_{ta} + \underline{H}_{tj}(1 - \tilde{x}_{ta}) \tag{5.8}$$

after introducing an auxiliary binary variable $\tilde{x}_{ta} \in \{0, 1\}$ defined as $\tilde{x}_{ta} = 1 \iff x_{ta} = 1 \wedge x_{(t+1)a} = 0$. In turn, this latter condition must be linearized in the MILP relaxation $(\mathcal{R})$ as:

$$\tilde{x}_{ta} \leq x_{ta}, \tilde{x}_{ta} \leq 1 - x_{(t+1)a}, \tilde{x}_{ta} \geq x_{ta} - x_{(t+1)a}$$

Note that, in this particular case, the auxiliary variable $\tilde{x}_{ta}$ may already exist in the model of the non-aging constraints (2.8).

Application to control variables. Another interest in this approach, which is favorable in our context, comes from its application to derive conditional bounds on the arc flow variables. Indeed, if by probing, we are able to restrict the domain of a variable q_{ta} to the union of disjoint intervals $\bigcup_c [\underline{Q}^c, \overline{Q}^c]$, then we could greatly improve the OA relaxation of the relation $v_{ta} = \psi_a(q_{ta})$ by computing tighter under- and over-estimators $\underline{\psi}^c_a$ and $\overline{\psi}^c_a$ independently on each interval, as depicted in Figure 5.2. Again, by introducing one auxiliary binary variable x^c for each interval c, we can linearize

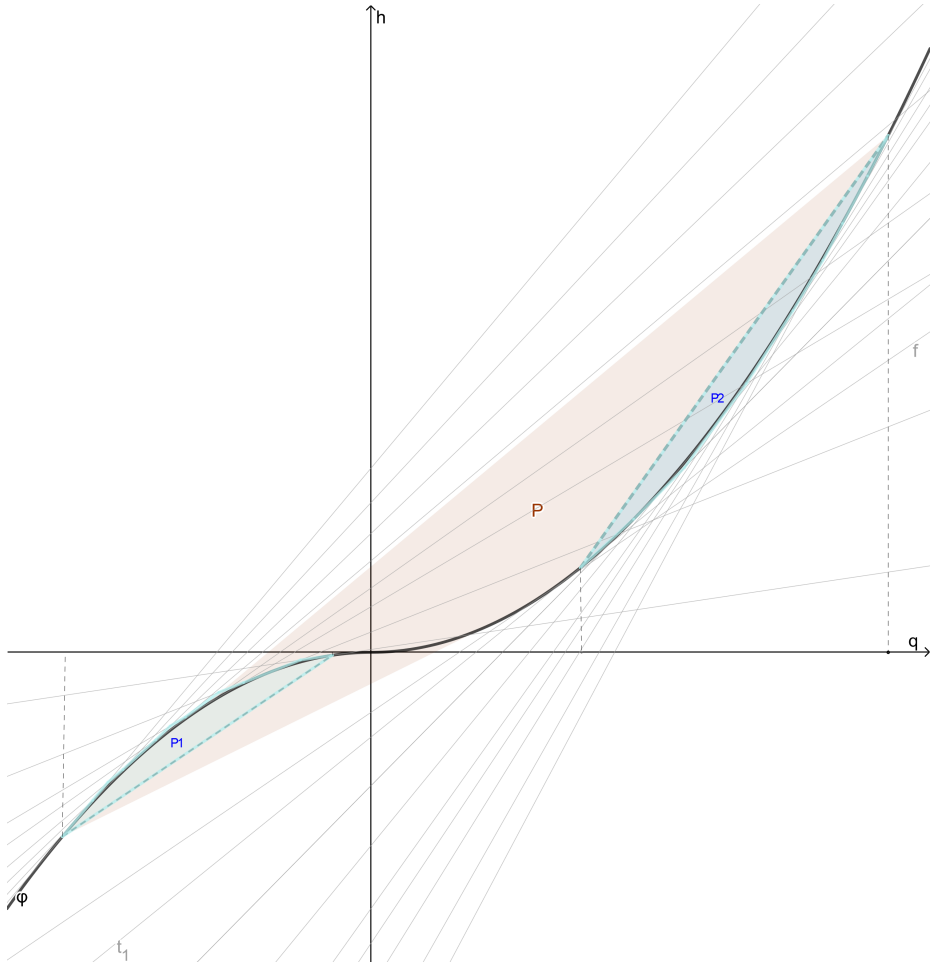

Figure 5.2: A nonlinear constraint (black curve), the OA relaxation computed from the initial bounds on q (orange), the new relaxation derived from disjoint domain intervals for q (blue).

the disjunction in $(\mathcal{R})$ as the following big-M formulation:

$$\sum_{c \in C} \underline{Q}^c x^c \leq q_{ta} \leq \sum_{c \in C} \overline{Q}^c x^c \tag{5.9}$$

$$\sum_{c \in C} (\underline{\psi}_a^c(q_{ta}) - \underline{V}_a) x^c + \underline{V}_a \leq v_{ta} \leq \sum_{c \in C} (\overline{\psi}_a^c(q_{ta}) - \overline{V}_a) x^c + \underline{V}_a \tag{5.10}$$

$$\sum_{c \in C} x^c = x_{ta}. \tag{5.11}$$

Another option is also to introduce copies q^c, v^c of the continuous variables on

each interval c as in the following convex-hull formulation:

$$\underline{Q}^c x^c \leq q^c \leq \overline{Q}^c x^c, \quad \underline{\psi}_a^c(q^c) \leq v^c \leq \overline{\psi}_a^c(q^c) \quad \forall c$$

$$\sum_{c \in C} x^c = x_{ta}, \sum_{c \in C} q^c = q_{ta}$$

$$\sum_{c \in C} v^c + \underline{V}_a(1 - x_{ta}) \leq v_{ta} \leq \sum_{c \in C} v^c + \overline{V}_a(1 - x_{ta}).$$

The LP relaxation of the convex-hull formulation is known to be stronger than for the big-M formulation [59]. In addition, to strengthen the relaxation ($\mathcal{R}$), we can relate the auxiliary variables x^c with the condition enforced in the OBBT relaxation that led to the domain reduction $[\underline{Q}^c, \overline{Q}^c]$.

A special case (illustrated in Figure 5.2) is when applying this approach to a bidirectional pipe by computing the bounds on the arc flow according to the direction of the flow through the pipe. The conditional bounds of a variable q_{ta} are then computed after enforcing either $q_{ta} \leq 0 \wedge x_{ta} = 1$ or $q_{ta} \geq 0 \wedge x_{ta} = 1$ in the OBBT relaxation $\mathcal{O}_t$. This results in two intervals, one in $\mathbb{R}_-$, the other one in $\mathbb{R}_+$, associated with two auxiliary binary variables, say x^- and x^+. Then, the relation between these variables and their respective conditions $q_{ta} \leq 0$ and $q_{ta} \geq 0$ is actually already modeled in the disjunctive formulation above.

Another special case is when introducing auxiliary variables can be evaded. For instance, consider a pipe outgoing from a pumping station made of some parallel pumps and valves $a \in PS \subseteq \mathcal{A}$. The constraints (2.8) typically include interdependencies between these elements, which limit the number of possible configurations to examine: $|C| << 2^{|PS|}$. Furthermore, if an element $a \in PS$ is set on (or off) in only one of these configurations, say $c_a \in C$, then the status variable x_{at} can be used as the indicator of this configuration, rather than adding an auxiliary variable x^{c_a}. More importantly, in this specific case, the conditional bounds $[\underline{Q}^c, \overline{Q}^c]$ on the flow outgoing the pump station can be inferred from the conditional bounds on the flows in the pumps and valves of the pumping station. Hence, they do not ask for solving additional OBBT problems.

Chapter 6

Valid Inequalities Generation

In this chapter, we show how to adapt several techniques of valid inequalities generation, such as minimum cardinality, cutset-based, and flow cover inequalities in the context of pump scheduling. These valid inequalities are computed during the preprocessing phase, after the OBBT algorithm, as they are usually based on some computed bounds. All constraints hence generated are added to the MILP relaxation ($\mathcal{R}$) before running the branch-and-check algorithm. As in [6], and contrarily to [39, 60], we do not generate cuts others than nogoods during the tree search. In particular, we do not generate locally valid inequalities, as it is not allowed by the MILP solver we employed (Gurobi). Finally, because we generate nogood cuts within the branch-and-check, the MILP solver hinders its built-in cut-generation process. We expect that generating our own valid inequalities can mitigate this loss.

6.1 Minimum cardinality cuts

We take advantage of the time horizon restriction by considering a subset of time intervals, and we focus on pushing the dual/lower bound on the objective. Instead of directly bounding the objective function, we propose considering the least number of pumps required to be activated. The first motivation is that the number of pumps activated on the whole time horizon provides a good approximation of the solution cost. Then, we expect that pushing up this bound will mechanically increase the dual bound. Indeed, the inequalities can be lifted along the storage of the tanks. We exploit the integrality of these values to strengthen the bound with the Mixed-Integer Rounding (MIR) technique [61].

Let us compute, for a restricted time horizon $\mathcal{T}' \subseteq \mathcal{T}$ and for a subset of pumps $\dot{\mathcal{A}}' \subseteq \dot{\mathcal{A}}$ the OBBT bound: $X' = \min\{\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'} x_{ta} : (x,q,h) \in \mathcal{M}_{\mathcal{T}'}\}$, where $\mathcal{M}_{\mathcal{T}'}$ denotes the multi-period relaxation introduced in Section 5.1.4. Then, the following inequality is valid in $(\mathcal{R})$:

$$\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'} x_{ta} \geq X'.$$

Considering a restricted set of pumps and a restricted time horizon has two purposes: (i) to make the OBBT subproblem $\mathcal{M}_{\mathcal{T}'}$ more tractable by reducing its size while enforcing a tight relaxation of the nonlinear constraints, and (ii) to obtain better deductions (minima, we need that $X' \geq 1$) by focusing on interdependent pumps and periods with high demand.

In particular, we propose to apply this technique when a group of pumps $\dot{\mathcal{A}}' \subseteq \dot{\mathcal{A}}$ serves one given branch of the network, possibly together with a tank $j \in \dot{\mathcal{C}}$ anywhere on the branch. In this case, we can strengthen the valid inequalities by lifting it regarding the tank level variable.

Let consider a time interval $\mathcal{T}' = [t_s, t_f - 1]$ and compute the lower bound after fixing the tank level arbitrarily to H^s and H^f at times t_s and t_f, respectively, as follows

$$X' = \min\Big\{\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'} x_{ta} \ : H_{t_s j} = H^s, H_{t_f j} = H^f, (x,q,h) \in \mathcal{M}_{\mathcal{T}'}\Big\}.$$

If the OBBT relaxation $\mathcal{M}_{\mathcal{T}'}$ is continuous, we also get the dual optimal values μ^s, μ^f of the tank level constraints.

Proposition 6.1.1. *The following inequality*

$$\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'} x_{ta} \geq \frac{-1}{b - \lfloor b \rfloor}(\mu^s \tilde{H}_{t_s j} + \mu^f \tilde{H}_{t_f j}) + \lceil b \rceil \tag{6.1}$$

is valid, with

$$\tilde{H}_{tj} := \begin{cases} H_{tj} - \underline{H}_{tj} & \text{if } \mu \geq 0 \\ -H_{tj} + \underline{H}_{tj}, & \text{otherwise} \end{cases}$$

and

$$b = X' + |\mu^s(H^s - \underline{H}_{t_s j})| + |\mu^f(H^f - \underline{H}_{t_f j})|.$$

Proof. By duality, the following inequality is valid:

$$\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'} x_{ta} \geq X' + \mu^s(H_{t_{sj}} - H^s) + \mu^f(H_{t_{fj}} - H^f).$$

Let denote $\tilde{\mu} = \mu^s\tilde{H}_{t_{sj}} + \mu^f\tilde{H}_{t_{fj}}$. Note that $\tilde{\mu}$ is non-negative and that $\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'} x_{ta}$ takes only integer values. Then, we can consider the disjunctive conditions $\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'} x_{ta} \leq \lceil b\rceil - 1$ and $\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'} x_{ta} \geq \lceil b\rceil$, then apply the arguments in [62], we obtain the desired inequality:

$$\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'} x_{ta} \geq \frac{-1}{b - \lfloor b\rfloor}\tilde{\mu} + \lceil b\rceil. \tag{6.2}$$

□

6.2 Cutset-based inequalities

We consider a connected subgraph of G made of one tank $r' \in \dot{\mathcal{C}}$, all the pumps and valves $\dot{\mathcal{A}}' \subseteq \dot{\mathcal{A}}$ deserving tank r', as well as all the pipes $\mathcal{A}' \subseteq \mathcal{A}\setminus\dot{\mathcal{A}}'$ and service nodes $\mathcal{S}' \subseteq \mathcal{S}$ in-between. An example is depicted in Figure 6.1: r' is the tank labelled A, $\dot{\mathcal{A}}'$ is made of one valve (labelled a) and 4 pumps (labelled b, c, d, e), then $\mathcal{A}'$ and $\mathcal{S}'$ are the pipes and service nodes delimited or crossed by the dashed line.

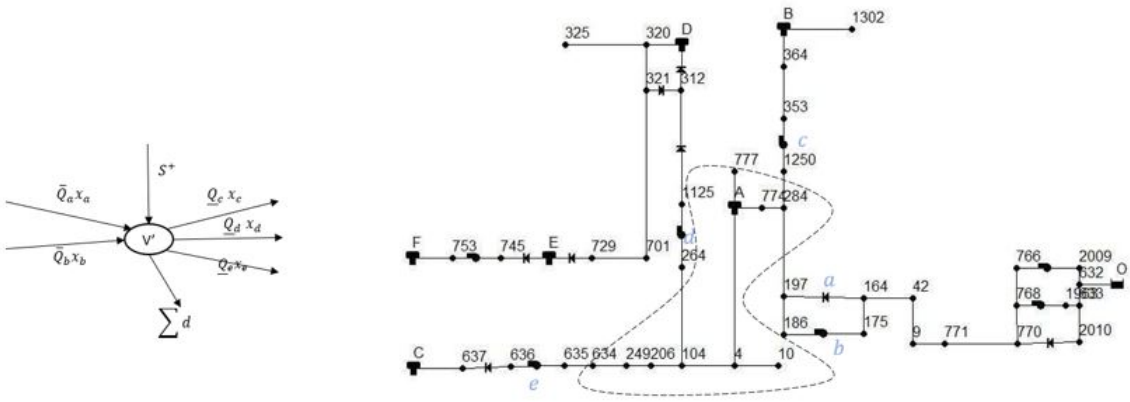

Figure 6.1: Example of a cutset made of one tank A served by 1 valve a and 4 pumps b, c, d, e.

By summing all the flow conservation constraints on the nodes $\mathcal{J}' := \{r'\} \cap \mathcal{S}'$ in the cutset, we obtain:

$$\sum_{j \in \mathcal{S}'} \sum_{a \in \mathcal{A}} E_{aj} q_{ta} + \sum_{a \in \mathcal{A}} E_{ar'} q_{tr'} = \sum_{j \in \mathcal{S}'} D_{tj} + \sigma_{r'}(h_{(t+1)r'} - h_{tr'}).$$

Each arc in $\mathcal{A}'$ connects two nodes in the cutset, while each arc in $a \in \dot{\mathcal{A}}'$ either enters the cutset ($\sum_{j \in \mathcal{J}'} E_{aj} = 1$) or leaves the cutset ($\sum_{j \in \mathcal{J}'} E_{aj} = -1$). By denoting $E'_a = \sum_{j \in \mathcal{J}'} E_{aj}$, we get:

$$\sum_{a \in \dot{\mathcal{A}}'} E'_a q_{ta} + \sigma_{r'}(h_{tr'} - h_{(t+1)r'}) = \sum_{j \in \mathcal{S}'} D_{tj}. \tag{6.3}$$

6.2.1 Coefficient reduction

Remember that pumps and valves are unidirectional arcs directed by flow, then $0 \leq \underline{Q}_{ta} x_{ta} \leq q_{ta} \leq \overline{Q}_{ta} x_{ta}$ for all $a \in \dot{\mathcal{A}}'$, and the equation (6.3) above can be relaxed as follows:

$$\sum_{a \in \dot{\mathcal{A}}'_+} \overline{Q}_{ta} x_{ta} + \sigma_{r'}(h_{tr'} - h_{(t+1)r'}) \geq \sum_{j \in \mathcal{S}'} D_{tj} + \sum_{a \in \dot{\mathcal{A}}'_-} q_{ta},$$

where we distinguish the entering arcs $\dot{\mathcal{A}}'_+ = \{a \in \dot{\mathcal{A}}' : E'_a = 1\}$ and leaving arcs $\dot{\mathcal{A}}'_- = \{a \in \dot{\mathcal{A}}' : E'_a = -1\}$. By summing this inequality for consecutive periods $t \in \mathcal{T}' = \{t_s, t_f - 1\}$, we get:

$$\sum_{t \in \mathcal{T}'} \sum_{a \in \dot{\mathcal{A}}'_+} \overline{Q}_{ta} x_{ta} + \sigma_{r'}(h_{t_s r'} - h_{t_f r'}) \geq \sum_{j \in \mathcal{S}'} \sum_{t \in \mathcal{T}'} (D_{tj} + \sum_{a \in \dot{\mathcal{A}}'_-} q_{ta}).$$

Now consider a time interval such that $h_{t_s r'} \geq \underline{H}_{t_s r'} \geq \underline{H}_{t_f r'}$ and any non-negative lower bound $d \geq 0$ of the right-hand side, then we can apply the coefficient reduction technique introduced in [63] to further tighten the inequality as follows:

$$\sum_{t \in \mathcal{T}'} \sum_{a \in \dot{\mathcal{A}}'_+} \min(\overline{Q}_{ta}, d) x_{ta} + \sigma_{r'}(h_{t_s r'} - \underline{H}_{t_f r'}) \geq d. \tag{6.4}$$

6.2.2 Mixed Integer Rounding

We now propose to apply the Mixed Integer Rounding (MIR) technique [64, 65, 66] to equality (6.3).

As above, we can relax (6.3) as:

$$\sum_{a\in\dot{\mathcal{A}}'_+} \overline{Q}_{ta}x_{ta} - \sum_{a\in\dot{\mathcal{A}}'_-} \underline{Q}_{ta}x_{ta} + \sigma_{r'}(h_{tr'} - \underline{H}_{(t+1)r'}) \geq \sum_{j\in\mathcal{S}'} D_{tj}.$$

We can reformulate the terms $-\underline{Q}_{ta}x_{ta}$ in the second added as $\underline{Q}_{ta}(\bar{x}_{ta}-1)$, by considering the complementary binary variables $\bar{x}_{ta} = 1 - x_{ta}$, so that all binary variables have now a non-negative coefficient. We then obtain an inequality with the following structure:

$$\sum_k c_k z_k + s \geq d,$$

where all binary variables z_k (either x_{ta} or $\bar{x}_{ta}$) have non-negative coefficients $c_k \geq 0$, s includes a positive continuous variable (here $h_{tr'}$) and $d \geq 0$ is a non-negative constant right hand side. The coefficient reduction related to MIR consists of replacing each coefficient c_k and constant d by their fractional part, $c'_k = c_k - \lfloor c_k \rfloor$ and $d' = d - \lfloor d \rfloor$, then to partition the variables as follows:

$$\sum_{k:c'_k<d'} c'_k z_k + d'\Big(\sum_{k:c'_k\geq d'} z_k + \sum_{k:c'_k<d'} \lfloor c_k \rfloor z_k\Big) + s \geq d'. \tag{6.5}$$

To be effective, the inequality can be normalized (by dividing c_k, s, d with a positive constant) such that there exists at least one binary variable z_k with $c'_k \geq d'$.

6.2.3 Flow cover inequalities

We now propose to generate cover subsets [64, 66, 67, 68], when considering a cutset as above. In this case, the tank inflow and the storage must be higher than the demand and the outflow from the cutset.

As before, we relax and reformulate (6.3) by introducing complementary binary

variables, then we sum over a time interval $\mathcal{T}' = \{t_s, \ldots, t_f - 1\}$:

$$\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'_+} \overline{Q}_{ta}(1-\bar{x}_{ta}) - \sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'_-} \underline{Q}_{ta}x_{ta} + \sigma_{r'}(h_{t_s r'} - \underline{H}_{(t_f)r'}) \geq \sum_{t\in\mathcal{T}'}\sum_{j\in\mathcal{S}'} D_{tj},$$

or, equivalently, by rearranging the terms, we obtain a knapsack constraint with non-negative coefficient on the binary variables:

$$\sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'_+} \overline{Q}_{ta}\bar{x}_{ta} + \sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'_-} \underline{Q}_{ta}x_{ta} - \sigma_{r'}(h_{t_s r'} - \underline{H}_{t_f r'}) \leq -\sum_{t\in\mathcal{T}'}\sum_{j\in\mathcal{S}'} D_{tj} + \sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'_+} \overline{Q}_{ta}.$$

We choose a cover that exceeds the right hand side, i.e. $C_+ \in \mathcal{T}' \times \dot{\mathcal{A}}'_+$ and $C_- \in \mathcal{T}' \times \dot{\mathcal{A}}'_-$ such that:

$$\lambda = \sum_{(t,a)\in C_+} \overline{Q}_{ta} + \sum_{(t,a)\in C_-} \underline{Q}_{ta} - (-\sum_{t\in\mathcal{T}'}\sum_{j\in\mathcal{S}'} D_{tj} + \sum_{t\in\mathcal{T}'}\sum_{a\in\dot{\mathcal{A}}'_+} \overline{Q}_{ta}) \geq 0$$

after simplifications, we have $\lambda = \sum_{t\in\mathcal{T}'}\sum_{j\in\mathcal{S}'} D_{tj} + \sum_{t\in\mathcal{T}'}\sum_{a\in C_-} \underline{Q}_{ta} - \sum_{t\in\mathcal{T}'}\sum_{a\in C'_+} \overline{Q}_{ta}$, where C'_+ is the complement of C_+ in $\mathcal{T}' \times \dot{\mathcal{A}}'_+$, and we can order the values in subsets C_- and C'_+ in decreasing order.

Hence:

$$\sum_{(t,a)\in C'_+} \overline{Q}_{ta}(1-\bar{x}_{ta}) - \sum_{(t,a)\in C_-} \underline{Q}_{ta}(1-x_{ta}) \leq -\lambda + \sigma_{r'}(h_{t_s r'} - \underline{H}_{t_f r'})$$

then applying the coefficient reduction:

$$\sum_{(t,a)\in C'_+} min(\overline{Q}_{ta}, \lambda)x_{ta} - \sum_{(t,a)\in C_-} min(\underline{Q}_{ta}, \lambda)(1-x_{ta}) \leq -\lambda + \sigma_{r'}(h_{t_s r'} - \underline{H}_{t_f r'})^+$$

By lifting variables from sets C'_+ and C_-:

$$\begin{aligned}
&\sum_{(t,a)\in C'_+} min(\overline{Q}_{ta}, \lambda)x_{ta} - \sum_{(t,a)\in C_-} min(\underline{Q}_{ta}, \lambda)(1-x_{ta}) \\
&- \sum_{(t,a)\in C'_+} \Phi(\overline{Q}_{ta})(1-x_{ta}) - \sum_{(t,a)\in C_-} \Phi(\underline{Q}_{ta})x_{ta} \\
&+ \sigma_{r'}(h_{t_s r'} - \underline{H}_{t_f r'})^+ \geq \lambda
\end{aligned} \tag{6.6}$$

with Φ defined as:

$$\Phi(a) = \begin{cases} i\lambda & \text{if } A_i \leq a \leq A_{i+1} - \lambda \\ i\lambda + (a - A_i), & if A_i - \lambda \leq a \leq A_i \\ p\lambda + (a - A_p) & A_p - \lambda \leq a \end{cases}$$

and $\lambda = d + \sum_{ta \in C_-} \underline{Q}_{ta} - \sum_{ta \in C'_+} \overline{Q}_{ta} \geq 0$, values in subsets C'_+ and C_- are sorted in decreasing order and each member of them are bigger than λ.

$$\{i \in C'_+ \cup C_- : a_i > \lambda\} = \{1, 2, ..., p\}, A_i = \sum_{k=1}^{i} a_k$$

Obviously, the strength of the cutset inequalities is highly dependent on the bounds of the flow variables. If the bounds are loose then the obtained inequalities can even be looser than the original equality. Another important aspect is how to find the cover and pack set. This will impact the strength of the flow cover inequality.

6.2.4 Difficulties of minimum cardinality cut generation

The most time-consuming valid inequalities to generate are to determine the minimum number of pumps are required to be generated until a certain time step to fulfill the demands and satisfy constraints such as tanks levels. It can push up the objective cost function since the number of active pumps directly contributes to the cost. However, this may have major drawbacks: the generation of such valid inequalities may demand a significant computational resources though most of the valid inequalities have been obtained for decomposed problems (e.g., for time step 0 to t' so effectively there is no impact of variables belong to larger time steps.)

6.3 Surrogate model

Generating tight inequalities based on the same MILP formulation implemented in the branch-and-check algorithm may take unaffordable computational efforts. Moreover, generating valid inequalities derived from the same relaxation as the one deployed for the branch-and-check algorithm may not be so informative. In response to these

issues, a new relaxation is introduced as complementary to the previously defined MILP relaxations constructed via outer approximation of the head-flow relationship.

We introduce a lighter relaxation of the previous formulation. For each linking branch between two tanks, we can define new types of relaxation by projecting out the majority of flow variables using the capacities of the arcs. This is analogous to the method defined in cutset-based inequalities and establishing supernodes. In fact, the surrogate model is a simplified representation of the original one materialized by collapsing some nodes and introducing subgraphs. This yields a graph with fewer arcs and nodes and a new relaxation of the problem. This less detailed model might yield a weak lower estimation for the original problem, in particular for cases where the difference between the upper and lower bounds of the flow variable (flow capacities of the arcs) is relatively large. For these cases, to control the loose relaxation of the conservation of the flows by using capacities of the arcs (max and min possible passing flows), We propose to associate the flow bounds to the difference in the levels of adjacent tanks (i.e., two tanks connected through arcs with no other reservoir in between.) The rationale is to restrict more the capacities of the arcs. The difference in the levels of the tanks are discretized into some intervals and, for each interval, we have associated flow bounds. As a result, for a branch linking two tanks, if the difference in the levels of the tanks is discretized to N intervals, we have N corresponding lower and upper bound capacities, as if these N arcs are located parallel to each other and at each time at most one of them can be activated.

Let us suppose a simple topology that the tank $j' \in \mathcal{J}$ is fed by a pump or a valve going out from the tank $j \in \mathcal{J}$, $a = (j, j')$. The bounds of the flow through a pump being in a connecting path from tank j to tank j' is $q_{jj'} \in [\underline{Q}_{tjj'}, \overline{Q}_{tjj'}]$ which can be discretized by associating the differences of the tank j and j' levels to flow bounds. Hence, we discretize the bound interval corresponding to the difference of the levels in two adjacent tanks $H_{tj} - H_{tj'} \in [\underline{H}_{tjj'}, \overline{H}_{tjj'}]$ then we apply probing on the conditional bounds of flows over steady-state relaxation. If we discretize the difference of the levels of the tanks in N intervals and we find the corresponding feasible bounds, then:

$$H_{tjj'} = H_{tj} - H_{tj'} \in [\underline{H}^{(1)}_{tjj'}, \overline{H}^{(1)}_{tjj'}] \vee ... \vee [\underline{H}^{(N)}_{tjj'}, \overline{H}^{(N)}_{tjj'}]$$

and

$$\text{if } H_{tj} - H_{tj'} \in [\underline{H}^{(n)}_{tjj'}, \overline{H}^{(n)}_{tjj'}] \text{ then } q_{tjj'} \in [\underline{Q}^{(n)}_{tii'}, \overline{Q}^{(n)}_{tjj'}],$$

More precisely for each arc $a = (j, j'), j, j' \in \dot{\mathcal{C}}$ connecting the tank j and j' we can obtain corresponding discrete bounds: $\overline{Q}^{(n)}_{ta} = \max\{q_{ta} | (q_t, h_t) \in \mathcal{E}(H_t, D_t, x_t), x_t \subseteq \{0,1\}^{\mathcal{A}}, H_{tj} - H_{tj'} \in [\underline{H}^{(n)}_{tjj'}, \overline{H}^{(n)}_{tjj'}], x_{ta} = 1\}$ and $\underline{Q}^{(n)}_{ta} = \min\{q_{ta} | (q_t, h_t) \in$

$\mathcal{E}(H_t, D_t, x_t), x_t \subseteq \{0,1\}^{\mathcal{A}}, H_{tj} - H_{tj'} \in [\underline{H}^{(n)}_{tjj'}, \overline{H}^{(n)}_{tjj'}], x_{ta} = 1\}$. Conspicuously, discretizing more the difference between levels of the tanks connected by a controlled arc in between creates more accurate relaxations at the price of more complicated combinatorial problem.

The trade-off between the accuracy of the relaxation and the integral complexity can be tuned by considering the bounds of the flow over the controllable arcs and the complexity of the original model (e.g., the number of time steps, pumps, and valves).

To illustrate better the way of constructing this surrogate model and establishing the simplified version of the graph G, we have shown the procedure over the *Poormond* network. As shown in Figure 6.2, the surrogate model defined below can

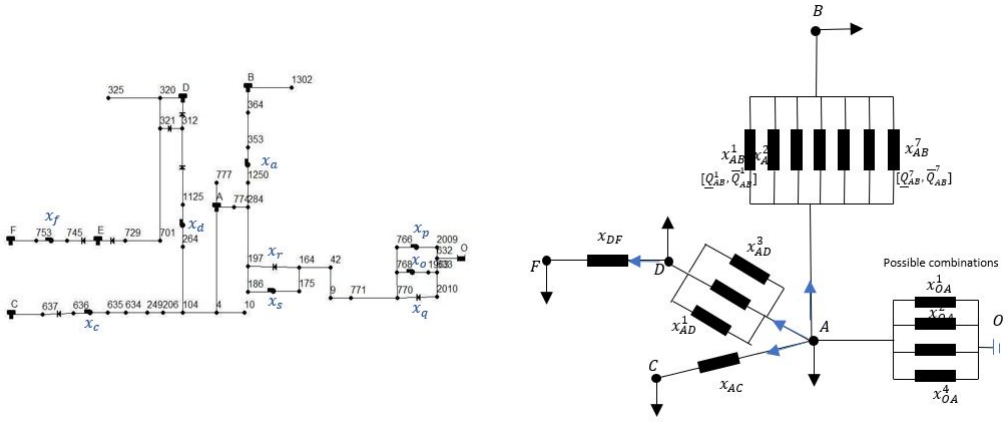

Figure 6.2: The graph simplifications and supernodes in *Poormond* network to establish new relaxation as a surrogate model.

be represented as a multigraph with various parallel edges connecting each pair of supernodes.

Surrogate model for *Poormond*. The *Poormond* network can morph into several supernodes comprising aggregated demands and storage tanks. The link between every two tanks with no reservoir between them can be represented as a controllable arc or a set of parallel arcs. The capacity (i.e., upper and lower flow passing through them if they are active) of each one of these parallel arcs is associated with the difference in the intervals of the tanks. If the controllable arc is on, then exactly one of these parallel control arcs can be activated. Apart from associating capacities to the difference of the levels of the tanks, from source reservoir to the central tank (i.e., tank A), there are several controllable arcs with relatively large bound variations; makes

it difficult to simplify the network and at the same time maintain the tightness of the relaxation. Considering different combinations of the activity of the controllable arcs facilitates to obtain a relatively small difference between the lower and upper bound for each combination. This leads to a tighter relaxation without discretizing further the levels of the tanks. Precisely, there are five possible combinations: *i.*(x_o, x_r), *ii.* (x_o, x_s), *iii.* (x_o, x_p, x_r), *iv.* (x_o, x_p, x_s), *v.* (x_q). While once the $x_q = 1$ (combination v is active) then all flow is passing through the arc is only satisfying a demand node (so there would be no inflow to tank A from the pump stations.) As a result, all arcs located at these pump stations and nodes can be collapsed and only be considered as parallel arcs which their flows upper and lower bound can be computed by enforcing the activity of each possible combination. The flow variable is only defined for in-outgoing of the tanks. In Chapter 7, We have introduced 7 discrete capacities between tank A and B and 3 between tank A and D. For other pathways we do not add any further auxiliary binary variables since the lower and higher capacities of the branches are very close to each other and the potentials of the tanks are not affected much by the flow variations in the arcs. Precisely, the relative difference of an upper and lower bounds of the linking arcs with respect to capacity of the destination or source tank is considered as an indicator for the number of required discretization. In this way the potential-based flow relaxation is modeled as a flow network with fictitious parallel arcs representing the difference of the storage in the tanks. All flow variables are excluded except the inflow and outflow of the storage tanks. As a result the number of continuous variables corresponding to arcs and nodes reduces significantly, thus instead of having flow variables for each element at each time step and the cutting planes determining the outer-approximation of the flow-head relationship, we have merely flows corresponding to storage tanks. We can write the new formulation for the simplified multi graph $\mathcal{G}'(\mathcal{J}, \mathcal{A})$ as below $\forall t \in \mathcal{T} \setminus \{0\}$:

$$\underline{H}_{tij} \leq H_{tij} \leq \overline{H}_{tij} \qquad \forall (i,j) \in \dot{\mathcal{A}}$$

$$\underline{H}_{tj} \leq H_{tj} \leq \overline{H}_{tj} \qquad \forall j \in \dot{\mathcal{C}}$$

$$\sum_{n \in N_a} x_{ta}^{n} \leq 1 \qquad \forall a \in \dot{\mathcal{A}}$$

$$x_{ta}^{n} \in \{0,1\} \qquad \forall a \in \dot{\mathcal{A}}, \forall n \in N_A$$

$$H_{tij} \leq \overline{H}_{tij}^{(n)} x_{tij}^{(n)} + \overline{H}_{tij}(1 - x_{tij}^{(n)}) \qquad \forall i,j \in \dot{\mathcal{C}}, a = (i,j) \in \mathcal{A}, \forall n \in N_a$$

$$H_{tij} \geq \underline{H}_{tij}^{(n)} x_{tij}^{(n)} + \underline{H}_{tij}(1 - x_{tij}^{(n)}) \qquad \forall i,j \in \dot{\mathcal{C}}, a = (i,j) \in \mathcal{A}, \forall n \in N_a$$

$$q_{ta} \leq \sum_{n \in N_a} x_{ta}^{(n)} \overline{Q}_{ta}^{(n)} \qquad \forall a \in \dot{\mathcal{A}}$$

$$q_{ta} \geq \sum_{n \in N_a} x_{ta}^{(n)} \underline{Q}_{ta}^{(n)} \qquad \forall a \in \dot{\mathcal{A}}$$

$$q_{tj}^{\mathcal{J}} = \sum_{a \in \mathcal{A}_+} q_{ta} - \sum_{a \in \mathcal{A}_-} q_{ta} - D_{tj} \qquad \forall j \in \dot{\mathcal{C}}, \forall a \in \mathcal{A}$$

$$q_{tj}^{\mathcal{J}} = \sigma_j (H_{tj+1} - H_{tj}) \qquad \forall j \in \dot{\mathcal{C}}$$

$$\underline{Q}_{tj}^{\mathcal{J}} \leq q_{tj}^{\mathcal{J}} \leq \overline{Q}_{tj}^{\mathcal{J}} \qquad \forall j \in \dot{\mathcal{C}}$$

Chapter 7

Numerical Results

In this section, we compare the effect of our preprocessing on the branch-and-check algorithm introduced in [6]. In the following, we first describe the computational resources and networks used as benchmarks. Then, we illustrate the impact of bound tightening and cut generation in the global optimization framework of the drinking water networks considered in this thesis.

7.1 Computational setup

We deploy various levels of preprocessing on different networks. Three networks are considered for evaluation of each formulation, namely *simple networks FSD*, *Poormond*, and *van Zyl* with time horizons $T = 12$, 24, and 48. The details about the networks are given in Table 3.1. The instances for *FSD* and *Poormond* are taken from [6]. The demand profiles of the instances belonging to *van Zyl* are from [26] while the tariff are different (they had considered only a single day/night tariff).

Below, we describe the different formulations of preprocessing to be combined with the branch-and-check algorithm.

C0. This formulation is used in the branch-and-check algorithm by Bonvin et al. in [6]. This may tighten the bounds more than what is calculated as the capacity of each arc based on the assumption of the lumped elements, usually obtained by considering the size and the material of the pipes to avoid wave propagation through the medium. We reconstruct the C0 reformulation by applying bound

tightening over steady-state relaxation with floating demand. The demand is assumed to vary from 80% of the minimum to 120% of the maximum of the given instance demand profile. The reason behind considering larger margins is that the initial bound should be able to handle everyday forecast demand profiles without removing feasible solutions.

C1. This formulation is derived after applying bound tightening over each variable (no compound variables or extended disjunctive programming), that is by using the techniques presented in Section 5. The type of relaxations in the preprocessing and the number of iterations are mentioned specifically for each network in the sections below.

C2. This formulation is derived after bound tightening, probing, and disjunctive programming introduced in Chapter 5.

C3. This formulation is obtained after adding problem-specific cut generation, presented in Chapter 6. The details are provided for each network in the sections below.

7.2 Parameters to control bound tightening and cut generation

The efficacy and effectiveness of the bound tightening and cut generation can be controlled through several parameters.

Steady-state relaxation with bound tightening. For the relaxation of the nonconvex head-flow constraint $\psi : q \in \mathbb{R} \rightarrow Aq|q| + Bq + C$, we have considered two main options: the outer approximation ($OA(\mathcal{E})$), which implements a polyhedral relaxation of the equality constraint for the head-flow relationship, and a piecewise linear relaxation ($PWLR(\mathcal{E})$) viable via auxiliary binary variables. The complexity of these relaxations (i.e., the number of cutting planes to establish outer approximation and auxiliary binary variables) is controlled by the maximum tolerance between the relaxation and the exact nonlinear function.

Multi-period bound tightening. We evaluate the impact of the bound tightening on domain reduction over the levels of the tanks and the difference of adjacent ones considering multi-period relaxations. The propagation and resulting domain reduction are also assessed over the domains of the other variables.

Number of iterations and stopping criteria Bound tightening is implemented in a sequential manner and the number of iterations can affect the domain reduction of the variables. However, in general, increasing the number of iterations significantly raises the computational cost of the preprocessing without tangible improvement of the domain reduction.

All algorithms are implemented in Python, and the numerical experiments are executed on an Intel Core(TM) 6148 1.10GHz i7-10810U and 32 GB memory. Gurobi 9.5 was used to solve the MILP relaxations and in the implementation of the branch-and-check algorithm. All the implementations can be found at `https://github.com/amirhtavakoli94/pmpscheduling_branch_and_check`

The effect of the bound tightening is assessed over variable domains. The improvement is reported as the percentage with respect to those derived by the formulation C0.

7.3 Effect of preprocessing on selected networks

7.3.1 Simple network

We begin by briefly commenting on the effect of preprocessing on the most difficult instances of the *Simple Network* (*FSD*), which is the case with $T = 48$. The impact of the formulations presented above on the branch-and-check algorithm is reported in Table 7.1. In this table and similar ones to be shown below, we indicate the name of the network and the time horizon, the type of formulation, and the reference of the instance solved. Moreover, we display the best upper and lower bounds computed as 'ub' and 'lb,' respectively. The column titled 'gap' reports the optimality gap (and the computational time when the gap is closed). To have a fair comparison, we consider a longer run time for C0 formulation. For *FSD* network, the first feasible solution is always obtained quickly (less than a few seconds), so we have not reported it in the table.

The results show that formulation C0 cannot close the optimality gap for four over five instances within the time limit of 4800s. We also consider formulation C1 and set a maximum computational time for the bound tightening equal to five minutes. Since this bound tightening is fairly quick, we consider $\mathcal{M}_{[0,48]}$, that a for multi-period relaxation over all the periods. In this case, we can solve three instances to optimality and reduce the optimality gap of the remaining. Finally, in formulation C3, we apply

cut generations consisting of minimum cardinality cuts with MILP formulation and the lifted version while the solution of the LP relaxation is utilized. The generation of cuts takes less than 950s. This formulation can find a global solution for all the considered instances.

network-T	formulation	instance	ub	lb	gap
FSD48	C0	1	150.9	150.9	0%(1960)
		2	155.7	154.8	0.4%
		3	168.6	167.1	0.9%
		4	176.0	175.6	0.2%
		5	145.6	144.6	0.7%
FSD48	C1	1	150.9	150.9	0%(1059)
		2	155.7	155.7	0%(2729)
		3	168.6	167.9	0.4%
		4	176.0	176.0	0% (799)
		5	145.6	144.9	0.5%
FSD48	C3	1	150.9	150.9	0%(182)
		2	155.7	155.7	0%(1720)
		3	168.6	168.6	0%(250)
		4	176.0	176.0	0% (220)
		5	145.6	145.6	0%(3237)

Table 7.1
Effect of the different formulations on the branch-and-check algorithm for *Simple Network* with $T = 48$.

7.3.2 Poormond

We compare the effect of the bound tightening over *Poormond* network. The first evaluation is on the effect of the type of relaxation, either piecewise relaxation or outer-approximation, with a tolerance of 0.01 on the single-period relaxation. The relative domain reductions of the variables and the time requirement to run each relaxation are crucial indicators of the efficacy and effectiveness of the method. We compare these two relaxations only considering the steady-state relaxation with no update of levels of the tanks.

The steady-state PWL relaxation takes 323 s to be completed for one iteration in the case $T = 12$ (PM12). The relative reductions of the bounds of the flow variables in

the controllable arcs and pipes are 53% and 27%. In comparison, each iteration of the bound tightening with outer-approximation is quicker and in the first iteration takes 58 s. However, the improvement in the bounds is smaller, that is, around 40% and 26%. After six consecutive iterations, which require 325 s, the bound improvement of the flow variables in arcs is only 51%. For $T = 24$ (PM24) after the first iteration with a run time of 473 s, the amount of improvement is 51% for the controllable arcs. While in outer-approximation, the first iteration takes 125 s and the domain reduction is 39%. After 5 iterations taking less than 600 s the improvement reaches 49%. The same trend is also seen for $T = 48$ (PM48).

In summary, the above results show that while outer-approximation requires a smaller computational effort than PWL relaxation to complete one iteration, the same amount of domain reduction is achieved with multiple iterations. Indeed, the computational results suggest that iteratively applying bound tightening over steady-state relaxation without updating the levels of the tank is not effective, and it does not improve the bounds for tight steady-state relaxation. We assess the improvements after bound tightening the levels of the tanks after each evaluation of the steady-state relaxation. For instances with $T = 12$ at periods closer to the initialization of the scheduling (e.g., $t = 1, 2$), we encounter substantial reductions over levels of the tanks since the initial levels of the tanks are known. The domain reduction over levels of the tanks is propagated to other variables and may tighten the steady-state relaxation.

Adding multi-period bound tightening enhances the bound propagation through the steady-state relaxation. The relative difference between bounds in the formulation C0 and the ones obtained from C1 is considered as a way to indicate the improvement. This is shown in Table 7.2. Note that if the average improvement of the flow bounds in two consecutive iterations is less than 0.5%, we do not run another iteration of bound tightening. With this strategy, the bound tightening requires three iterations and a computational time of 847 s, 1739 s, and 2963 s for $T = 12$, 24, and 48, respectively. Table 7.2 shows that a remarkable decrease in the flow domains is obtained, both over the controllable arcs ($Q_{\mathcal{A}}$) and the pipes ($Q_{\mathcal{A}_L}$). Good results are also shown for the head loss of the controllable arc, denoted as $V_{\mathcal{A}}$.

The relative improvement seems less remarkable for the levels of the tanks, denoted as H_J. However, any shrinkage over these bounds propagates and impacts the single-period bound tightening. Therefore, its effect cannot be deemed merely by the reduction of the capacities of the tanks. For multi-period relaxation $\mathcal{M}_{[t_s,t_e]}$, we have not considered all periods but only a small neighborhood around each $t \in \mathcal{T}$. In particular, we have considered three steps before and after for any t far from the beginning and the end of the time horizon. The domain reduction is not very different from the situation in which all periods are considered; however, the computational effort is substantially lower. For instance, for PM12, the required time to $\mathcal{M}_{[0,12]}$

network-T	$Q_{\dot{A}}$	$Q_{\mathcal{A}_L}$	$V_{\dot{A}}$	H_J
PM12	64.9%	36.2%	49.4%	17.9%
PM24	63.2%	33.4%	47.7%	12.2%
PM48	62.5%	33.0%	47.3%	12.1%

Table 7.2
Performance of the bound tightening in terms of domain improvement *Poormond* network.

and two iterations and the same configurations as reported parameters is more than 4600 s. Instead, when a smaller number of periods is considered, $\mathcal{M}_{[t-3,t+3]}$, the computational time for the same number of iterations is less than 600 s.

The formulation C2 involves the application of probing and disjunctive programming techniques. For each tank in this network, we have enforced different configurations of the controllable arcs attached to the tanks, as presented in equations (5.8). For each one that has more than one period, we have lifted the decision variables to model the cut. For the tanks located at the end of the graph (i.e., three tanks in this network), there is no reason to enforce inactivity of the feeding controllable arcs since the amount of the flow exiting the tanks, while the feeding pump is off, is equal to the demand and cannot be affected by the potential nodes. This conditional bound tightening overall does not take more than one minute (e.g., for $T = 24$, it takes around 35 s). The *Poormond* graph gives many opportunities to enforce extended probing and disjunctive programming, represented in (5.11). The flow bound of each of the arcs located after pumps stations can be further tightened by considering possible combinations of controllable arcs activities. Therefore, the outer-approximation of the head-flow relationship in these arcs can be associated to the activity of each combination and be modeled by disjunctive programming. Therefore, for each arc the number of outer-approximations is associated to the lifted binary variables representing each combination. The directions of the flow through bidirectional arcs are also dependent on the activity of the control arcs (except for one pipe connected to the central storage tank, for which the direction is also dependent on the levels of the tanks). This can be deduced directly from the flow conservation and the derived bounds of the active control. Instead of running one conditional bound tightening by enforcing the direction of the flow through the arc, we simply relate the conditional bounds to the activity of controllable arcs and the conservation of the flow. For PM24, finding these conditional bounds to apply extended probing and disjunctive programming only takes less than 30s.

The formulation C3 is obtained after enforcing cuts described in Chapter 6. For each tank and at each t, lifted minimum cardinality inequalities have been generated

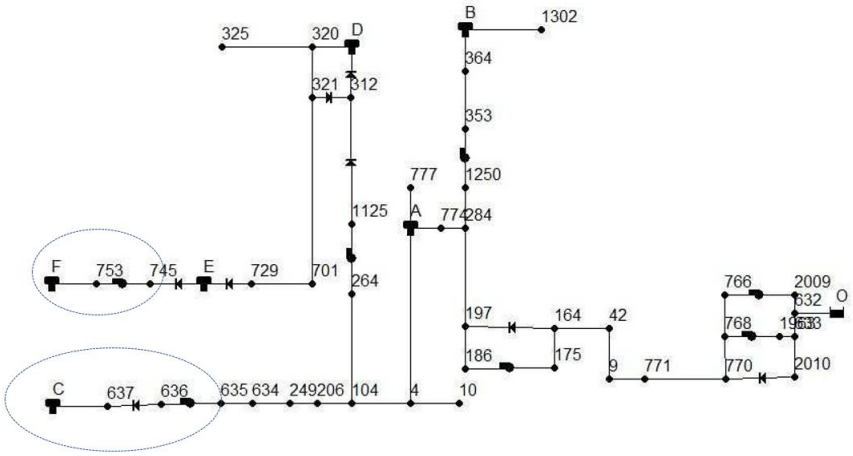

Figure 7.1: Bidirectional pipes in *Poormond* network: the direction of the flow and, consequently, the outer approximation is dependent on linking controllable arcs.

for levels of the tank solution of the LP relaxation and their associated Lagrangian multipliers. Moreover, the minimum cardinality cuts are generated for the number of active pumps located in the pump stations from the source to the central storage tank, that is $|\mathcal{A}'| = 4$ and all relatively bigger pumps $|\mathcal{A}'| = 5$ and for each subset of time steps $\mathcal{T}' = [0, t]$ with $t \in \{1, ..., T\}$. For t far from the beginning of the scheduling horizon, closing the gap of the auxiliary optimization problem is computationally expensive, and it makes it unreasonable to be applied at the preprocessing. Besides, extracting inequalities from the same relaxation which will be applied in branch-and-check algorithm may not be so informative. To find a tight bound in a reasonable amount of time, we have constructed the surrogate model presented in Section 6.3.

To generate cutset-based inequalities, we suggest projecting out the flow variables and using supernodes. The supernodes are established for each tank located in the network. For tanks directly connected to the demand node, a cutset-based inequality such as (6.4) is enforced. Similar inequalities are generated by aggregating consecutive periods (i.e., $|\mathcal{T}'|$ is set equal to 2, 3, or 4 in numerical experiments). For instance, we can see in Figure 7.1 that tank B is fed by a pump and is connected to a demand node separated from the rest of the network by the tank itself. We apply cutset-based inequalities for several aggregated periods. For storage tanks connected to in-outgoing controllable arcs, inequalities such as (6.4) can be enforced. For each tank fed by control arcs we have enforced such inequalities to exploit the integrality constraints. To enforce these inequalities, a supernode established by aggregating all the demand satisfied directly by the storage in the tank A. To have tighter representation of the flow variables of the in-outgoing arcs, the possible combinations of active controllable arcs at the pump stations are considered (similar to surrogate model example in 6)The

capacities of a subset of in-outgoing arcs are considered as a cover and remaining ones are entered via lifting. This subset is obtained based on the LP relaxation at the root node.

The first sign of improvement in the mathematical formulation after enforcing bound tightening and cut generation can be observed in the improvement of dual bound at the root node illustrated in Table 7.3. On average for all the instances, by applying bound tightening on variables (i.e., formulation C1), the bound is improved around 4%. However, the extended probing and disjunctive programming does not dramatically change the LP relaxation. The reason can stem from the fact that the tighter polyhedral relaxations are modeled by introducing auxiliary binary variables or by relating them to the existing ones (see Chapter 5), and consequently, in an LP relaxation, the tighter polyhedra may be compromised by fractional values. On the other hand, formulation C3 allows us to obtain promising results. In one of the instances, the improvement with respect to formulation C0 even reaches 19%. The reason for the efficacy of formulation C3 might be related to the fact that the cutting planes proposed in Chapter 6 directly act on the relaxation of the binary variables considered in the relaxed formulations.

In Tables 7.4, 7.5, and 7.6, we report the impact on the branch-and-check algorithm of formulations C1, C2, and C3. The notation is similar to that used in a previous table, and we also add the column titled '1st' to indicate the time to acquire the first feasible solution. The progress can be observed in various aspects: the optimality gap and the time to compute the first feasible solution. We also look at the number of instances with certified optimality.

For some instances, even finding a feasible solution is in limbo, and in practice, it might exacerbate the situation for an operator to plan for a day ahead. This situation can be observed in Table 7.4 for the case $T = 12$, where the formulation C0 has difficulties finding any feasible solution for the instances of PM12. On the other hand, employing the formulation C1 makes it possible to obtain a feasible solution for all instances and close the optimality gap for one of them. A great improvement can be observed in formulation C2 as all instances can be solved to optimality. The performance of the branch-and-check algorithm for the scheduling problems $T = 24$ with different formulation is illustrated in Table 7.5. The optimality gap is around 4.4% on average in formulation C0 and 2.8%, 2.3% and 2.1% on average in C1, C2 and C3, respectively. The average time to obtain the first feasible solution shrinks from around 700 s to 100 s on average. A similar improvement rate can be observed in Table 7.6 for the case $T = 48$. The results of the instances in PM24 and PM48 clearly illustrate the advantage of powerful preprocessing. The formulations C2 and C3 outperform C0 and C1 in both upper and lower bounds.

network-T	instance	C0	C1		C2		C3	
			Value	% Impr.	Value	% Impr.	Value	% Impr.
PM12	1	89.26	94.50	5.87	94.62	6.00	-	-
	2	90.79	96.40	6.18	96.58	6.37	-	-
	3	100.27	106.45	6.16	106.74	6.44	-	-
	4	110.88	116.96	5.49	117.07	5.59	-	-
	5	82.12	88.19	7.39	88.26	7.47	-	-
PM24	1	89.27	93.14	4.32	93.30	4.51	103.23	15.63
	2	90.80	94.91	4.52	95.13	4.77	105.44	16.11
	3	100.27	104.50	4.21	104.86	4.57	117.24	16.93
	4	110.90	115.33	4.00	115.49	4.15	127.60	15.06
	5	75.30	78.30	3.98	78.37	4.07	89.78	19.25
PM48	1	89.27	93.11	4.30	93.26	4.46	102.23	14.53
	2	90.80	94.87	4.49	95.10	4.74	103.74	14.25
	3	100.27	104.44	4.16	104.82	4.54	115.86	15.54
	4	110.90	115.31	4.24	115.47	4.41	126.64	14.30
	5	75.33	78.26	3.89	78.35	4.01	87.70	16.42

Table 7.3
Percentage of dual bounds improvements for *Poormond*.

The improvement in the LP relaxation has not always been as impressive in the branch and bound algorithm. In particular the performance of branch and check algorithm from C1 to C2 is significant whereas the improvement in LP relaxation is not so impressive. On the contrary, in some instances remarkable improvements in LP relaxation cannot proportionally be translated to improvement in the performance. This can be partially the result of the fact that the cuts generated in C3 might overlap with built-in cutting planes.

network-T	formulation	instance	ub	lb	gap	1st
PM12	C0	1	NA	110.7	NA%	NA
		2	NA	113.5	NA%	NA
		3	NA	123.9	NA%	NA
		4	NA	137.2	NA%	NA
		5	NA	112.5	NA%	NA
PM12	C1	1	114.6	112.7	1.7%	727
		2	117.7	115.7	1.8%	86
		3	133.9	126.9	5.3%	2493
		4	141.6	141.6	0%(1517)	243
		5	118.0	114.8	2.7%	2149
PM12	C2	1	114.1	114.1	0%(757s)	40.6
		2	117.5	117.5	0%(2088s)	67.6
		3	130.3	130.3	0%(3982s)	211.8
		4	141.6	141.6	0%(916s)	180.6
		5	117.1	117.1	0%(938s)	45.3

Table 7.4
Effect of the different formulations on the branch-and-check algorithm for *Poormond* with $T = 12$.

network-T	formulation	instance	ub	lb	gap	1st
PM24	C0	1	113.2	107.8	4.8%	572
		2	114.4	110.6	3.2%	125
		3	126.0	121.5	3.6%	319
		4	140.0	134.6	4.5%	2001
		5	96.1	92.8	5.8%	450
PM24	C1	1	111.9	108.0	3.5%	117
		2	114.2	110.4	3.3%	463
		3	125.3	122.4	2.3%	27
		4	138.0	134.5	2.6%	104
		5	96.1	93.8	2.3%	150
PM24	C2	1	111.0	108.3	2.4%	48.6
		2	113.8	111.3	2.2%	159
		3	125.3	122.4	2.3%	74
		4	138.5	134.9	2.6%	44
		5	96.1	94.1	2%	204
PM24	C3	1	111.8	108.9	2.5%	94
		2	114.0	111.6	2.0%	164
		3	125.7	123.2	2.0%	110
		4	138.8	136.1	2.0%	104
		5	96.1	94.4	1.8%	49

Table 7.5
Effect of the different formulations on the branch-and-check algorithm for *Poormond* with $T = 24$.

network-T	formulation	instance	ub	lb	gap	1st
PM48	C0	1	109.4	106.9	2.3%	1041
		2	112.9	109.1	3.4%	259
		3	125.9	120.5	4.3%	1017
		4	135.4	132.9	1.9%	585
		5	94.5	90.6	4.0%	166
PM48	C1	1	110.3	107.2	2.8%	214
		2	113.3	109.4	3.3%	235
		3	124.3	121.4	2.4%	252
		4	136.9	133.5	2.5%	353
		5	93.9	91.2	2.9%	62
PM48	C2	1	109.4	107.4	1.8%	118
		2	112.2	109.6	2.3%	158
		3	124.1	121.3	2.2%	163
		4	135.9	133.6	1.7%	124
		5	93	91.3	1.9%	114
PM48	C3	1	109.5	107.4	1.9%	869
		2	111.9	109.7	2.0%	242
		3	123.6	121.4	1.8%	265
		4	136.2	133.7	1.9%	93
		5	93.6	91.6	2.1%	347

Table 7.6
Effect of the different formulations on the branch-and-check algorithm for *Poormond* with $T = 48$.

7.3.3 van Zyl

We show that the use of bound tightening significantly impacts the domain of the variables in *van Zyl* (VZ) network. We stop the bound tightening algorithm when the improvement of flows through the arcs in the current iteration with respect to the previous one is less than 0.5% percent. Table 7.7 shows the number of iterations and the computational time in seconds to reach this stopping criterion.

When the single-period relaxation in the bound tightening is considered, the reduction in the domain of the flow bounds over the controllable arcs is about 26%, 29%, and 32% in VZ12, VZ24, and VZ48, respectively. Table 7.8 reports the impressive results on the domain reduction derived from multi-period bound tightening. The bound tightening over levels of the tanks are deployed after other bound tightenings (i.e., after bound tightening over steady state relaxation) at each iteration. The improvement of the capacities of the tanks can be an indicator of the subsequent improvements. If there is no reduction in the bounds of the levels of the tanks, no further improvement may be achievable for other variables.

In formulation C2, we consider extended probing and disjunctive programming. In this network, there are four controllable arcs at each time step. We have introduced auxiliary binary variables by the lifting approach to model multilinear terms representing each possible combination of activity of controllable arcs. On top of the bound tightening procedures in C1, we have applied conditional bound tightening for each combination. We have applied steady-state PWL relaxation for probing to obtain these conditional bounds. For $T = 12$, 24, and 48, the computational time required to run probing is 73 s, 170 s, and 410 s, respectively.

In formulation C3, the minimum cardinality cuts are implemented to further tighten the polyhedral relaxations. We devise different relaxations to derive the cuts, different from the one deployed in the branch-and-check algorithm. The network can be split into two subgraphs (see graph partition in Chapter 2). The first subgraph comprises pumps and valves. The polyhedral outer-approximation can be further tightened only by relating the activity of these controllable arcs to polyhedral outer-approximations. However, the other part is isolated by two tanks. Therefore, the polyhedral relaxation may not be tightened by relating the activity of the controllable arcs. We define new relaxations according to the difference in the levels of the tanks. For the arcs between these two tanks, we have computed new bounds by discretizing the difference in the levels of the tanks. Then, for each discretized interval, new flow bounds have been associated with the tank intervals with implied binary variables. The outer-approximation is built up based on these new bounds with disjunctive programming as shown in Section 4.3.4. Then, we use this relaxation to find

network-T	# iterations	required time
VZ12	3	205 s
VZ24	4	1045 s
VZ48	5	1300 s

Table 7.7
Performance of the bound tightening in terms of computational time and iterations for *van Zyl*.

network-T	$Q_{\mathcal{A}}$	$Q_{\mathcal{A}_L}$	$V_{\mathcal{A}}$	H_J
VZ12	50.9%	32.8%	64.0%	35.6%
VZ24	48.0%	30.6%	62.1%	39.0%
VZ48	43.8%	27.0%	62.3%	35.7%

Table 7.8
Performance of the bound tightening in terms of domain improvement *van Zyl*.

the minimum number of pumps needed to be activated from $t = 0$ to $t' \in \mathcal{T}$. For VZ24, we have used five intervals to discretize the difference of levels of the tanks, and for VZ48, we have two intervals for all periods $t \in \mathcal{T} \setminus \{0\}$.

In Tables 7.9 and 7.10, we show the effect of the different formulations on the branch-and-check algorithm. The notation is similar to the experiments performed on the other networks. For the case with $T = 12$ in Table 7.9, formulation C0 has difficulties solving instances to optimality. Within one hour of runtime, the branch-and-check algorithm leaves two instances without proof of optimality. It takes around 2600s and 3000s to solve two instances to optimality. The bound tightening can significantly improve the branch-and-check algorithm. The proof of optimality for all instances with C1 formulation is given in less than a few minutes. For formulation C2, all instances are solved in less than 70s. The results suggest the importance of the preprocessing stage in overall computational cost.

The preprocessing also substantially improves the global optimality of VZ24, as shown in Table 7.10. While the branch-and-check algorithm cannot obtain any feasible solution within the given maximum computational time with formulation C0, the optimality gaps shrink to approximately 2% in formulation C3. The other interesting result is that using more sophisticated relaxation by applying more cutting planes for outer-approximation of the head-flow relationship and introducing binary variables, have not improved the performance. It is difficult to claim that the formulation C2 outperforms C1. Probably, a trade-off is required to control deployments of the cuts.

network-T	Formulation	instance	ub	lb	gap	1st
VZ12	C0	1	265.3	263.5	0.7%	NA
		2	274.9	271.3	1.3%	NA
		3	302.4	302.4	0%(3076s)	NA
		4	340.6	340.6	0%(2672s)	NA
VZ12	C1	1	265.3	265.3	0%(31s)	NA
		2	274.9	274.9	0%(54s)	NA
		3	302.4	302.4	0%(301s)	NA
		4	340.6	340.6	0%(34s)	NA
VZ12	C2	1	265.3	265.3	0%(40s)	NA
		2	274.9	274.9	0%(72s)	NA
		3	302.4	302.4	0%(15s)	NA
		4	340.6	340.6	0%(41s)	NA

Table 7.9
Effect of the different formulations on the branch-and-check algorithm for *van Zyl* with $T = 12$.

network-T	Formulation	instance	ub	lb	gap	1st
VZ24	C0	1	NA	237.6	NA	NA
		2	NA	249.8	NA	NA
		3	NA	251.5	NA	NA
		4	NA	289.2	NA	NA
VZ24	C1	1	267.5	259.6	3%	245
		2	278.3	274.3	1.5%	72
		3	279.1	276.1	1.1%	31
		4	324.7	316.2	2.7%	75
VZ24	C2	1	266.9	258.6	3.3%	25
		2	280.1	271.8	3.3%	62
		3	283.0	274.7	2.9%	30
		4	325.6	315.1	3.3%	18
VZ24	C3	1	265.4	259.7	2.1%	85s
		2	278.3	273.1	1.9%	306
		3	282.4	276.2	2.2%	23s
		4	324.4	316.8	2.3%	91

Table 7.10
Effect of the different formulations on the branch-and-check algorithm for *van Zyl* with $T = 24$.

If we were to implement the formulation like the one from which we derived cardinality cuts (introducing implied binary variables to indicate the difference of the levesl of the tanks), It not only would fail to improve the performance but also would exacerbate the optimality gap. This may be related to the introduction of implied binary variables, which increases the complexity of the formulation. For instance, by defining five new binary variables for discretizing the differences of levels of the tanks and new outer-approximation based on these discrete differences, both lower and upper bounds obtained after one hour of branch-and-check algorithm are worse than the formulation C2. Therefore, this relaxation might not be appropriate and too heavy for the branch-and-check algorithm. The surrogate model for VZ24 is built by dividing the possible difference of the two tanks at each time step into ten discrete values. The time required to enforce bound tightening is less than 120 s, and finding the bounds for cardinality cuts for all time steps $t \in \mathcal{T}$ requires around 400 s. For VZ48, the formulation C0 cannot find any feasible solution. However, even employing bound tightening and cutting planes does not allow finding any upper bound. This indicates the limitation of the proposed bound tightening and preprocessing.

Part III

Combining Machine Learning and Mathematical Decomposition

Chapter 8

Motivation and Literature Review

8.1 Decomposition for pump scheduling

For a long period, the pump scheduling has been solved only approximately, using linear or piecewise linear approximations of the resistance constraints. Another body of research focused on feasibility at the expense of optimality. Metaheuristics based on evolutionary algorithms have been a predominant method to find feasible solutions for pump scheduling problems, as shown in the review paper [28]. Among others, we indicate genetics algorithms [69] possibly paired with a local search strategy [26], simulated annealing [70], or ant colony algorithms [27]. These are often simulation-based optimization methods, which exploit the bilevel structure of the pump scheduling problem and the subsequent time decomposition of the inner-level problem (see Section 2.3.2). They search (incompletely) candidate solutions (binary schedules $x \in \{0,1\}^{T|\mathcal{A}|}$) and evaluate each candidate by applying a hydraulic simulator (such as EPANET [71] or the extended analysis algorithm) to solve the inner-level problem. Except for the fitness value computed, no additional information guides the search.

The matheuristic of Ghaddar et al. [4] exploits the bilevel structure and time decomposition in two-way communication. They dualize the time-coupling constraints (2.4) in the MINLP ($\mathcal{P}$) and solve the resulting Lagrangian relaxation with a cutting-plane algorithm so that the dual information is then used to guide the search. However, the proposed decomposition leads to solving, iteratively, independent single-period subproblems (studied in [54]) far being as simple as the equilibrium problem. Indeed, the solution of a subproblem is a fixed configuration of both pumps

and tank levels (i.e., $x_t \in \{0,1\}^{\mathcal{A}}$ and $H_t \in \mathbb{R}^{\dot{\mathcal{C}}}$) such that the associate equilibrium solution $(q_t, h_t) \in \mathcal{E}(H_t, D_t, x_t)$ minimizes some penalized objective function. Furthermore, after solving the Lagrangian relaxation, a second stage is required (a Limited-Discrepancy Search is used in [4]) to restore a feasible solution from the solution of the Lagrangian relaxation. The time-coupling constraints (2.1) for limiting the aging of the pumps are ignored in the first stage and enforced in the second stage.

In [72], Ulanicki et al. propose a two-stage algorithm relying on the following assumption, based on numerical experiments and industrial experience (e.g., [73]): the reservoir trajectories from a continuous solution are sufficiently close to optimal reservoir trajectories of the mixed-integer solution. They propose to solve the continuous relaxation of $(\mathcal{P})$ in the first stage, then to derive a discrete schedule tracking the optimal tank trajectories in the second stage, using time decomposition and local branch and bound. The paper describes an efficient alternative to solve the continuous relaxation using dynamic optimization and Sequential Quadratic Programming, but it does not describe, nor experiment, the second stage. The authors argue that experienced network operators can manually translate the relaxed continuous solution and resulting tank-level trajectories into a practical pump schedule. Still, it is not trivial to get a feasible solution of the theoretical model $(\mathcal{P})$ based on this.

The present work follows this line of research. We exploit the separability of the pump scheduling problem into single-period subproblems to derive strictly feasible solutions of $(\mathcal{P})$ of good quality (but possibly sub-optimal). Contrary to metaheuristics and similarly to Lagrangian relaxation, we aim to use the solutions of the subproblems, not just their costs, to guide the search. But, contrary to the Lagrangian relaxation approach in [54], we also exploit the simplicity of the equilibrium subproblems once the tank level trajectories are known.

Our proposition, detailed in Chapter 9, is to solve the same Lagrangian relaxation (dualizing the time-coupling constraints), but in an approximate way, using a variable-splitting approach by iterating over two restrictions: (1) fix the tank level trajectories and get the best schedule and associate flows; (2) fix the flows and recompute the tank level trajectories. As the tank levels are fixed, the subproblem (1) is separable both in time and space, as seen in Section 2.3. It then becomes possible to enumerate the configurations of pumps independently on each graph component, and subproblems (1) (as well as the linear subproblems (2)) can be solved quickly. Our application of variable splitting thus appears as an option for solving the second stage in Ulanicki's approach, i.e., to derive a feasible solution for $(\mathcal{P})$ starting from given tank level trajectories. However, to initialize our algorithm, instead of solving the continuous relaxation in the first stage, we propose to predict the tank trajectories by learning

from the history of the water network operations.

8.2 Opportunity for machine learning

In practice, the pump scheduling problem is solved on a daily basis. This means that for a given network, an operator can gather a considerable amount of data sets, such as a (sub-)optimal schedule corresponding to given demand and tariff profiles. Such a repetitive and data-rich environment leads us to devise a data-driven approach for the pump scheduling problem.

Reinforcement learning absorbs some attention. In [5, 74], authors propose a sequential optimal decision making. In the presence of uncertainties, the agent modifies the decisions (activity of the pumps or their speed) after receiving new measurements or updates on forecast demand. This provides a reasonable framework for tackling uncertainties.

On the other hand, in the pump scheduling problem, the difference between instances only stands in the input data (demand and tariff), while the topology of the network remains the same. Therefore, the problem can be represented as supervised learning, which aims to map (approximately) from the input data to optimal solutions.

Recent advances in Machine Learning (ML) algorithms and their applications to optimization problems reinforce this idea of developing a learning-based algorithm to derive the solutions, i.e., the idea of building a data model instead of (or in addition to) solving a mathematical model. This approach does not provide certificates of optimality, but it may, in some applications, result in feasible solutions. However, in some applications, such as pump scheduling problems, even finding a feasible solution is a difficult task. Indeed, feasible decisions are usually sparse and scarce in the space of solutions $\{0,1\}^{\mathcal{TA}}$, especially when the pumps are big, the tanks are small, or the time steps are long. In ML-based optimization, a post-processing stage is required to recover feasibility from an approximate learned solution. The decomposition method we proposed can be seen as a feasibility-recovery post-processing stage.

In this work, we propose two learning approaches to approximate the (sub-)optimal partial solution of the pump scheduling problem.

The first approach, to be presented in detail in Chapters 10, is to use ML to learn the state variable H (the level of the tanks) instead of the control decision x (the

schedule), as usually done in the literature. This is motivated first by the similar assumption as in [72], that the optimal tank level trajectories in approximate models are close to the optimal tank level trajectories in the exact model. On the contrary, the neighborhood of a learned vector x is likely to contain no feasible solution. Also, it allows for smoother (continuous vs. discrete) local moves when exploring the neighborhood. As we will show in the numerical section (Chapter 12), when this learning approach is combined with the proposed decomposition method, feasible solutions of good quality (i.e., with a small objective function value) can be obtained.

The second learning approach is still designed to predict the state variable as the final goal. However, two main differences can be found. One is in the implementation of an intermediary step, which involves learning the control variables. The other aspect is the introduction of the physical constraints in the learning architecture. Also, in this case, we will combine the output of the learning approach with the decomposition algorithm. As shown in Chapter 12, this combination allows obtaining feasible solutions in a shorter computational time but with a lower quality.

8.3 Literature review on hybrid methods

Before describing and experimenting with our hybrid machine learning / mathematical programming in the next chapters, we review some works in this category.

The use of machine learning for enhancing mathematical programming has attracted lots of attention in both communities. In particular, in combinatorial optimization, the development of learning algorithms to guide the optimization process can be categorized at various stages, as explained below.

Significant efforts have been devoted to speeding up branch and bound algorithms to find the best variables and nodes to branch on. Alvarez et al. [75, 76] employ supervised learning to approximate strong branching with a small computational cost to speed up the Oracle process time. The concept behind the work is similar to shallow explorations to investigate which part of the branch and bound tree search is better to focus on. Khalil et al. [77] develop a framework for data-driven, on-the-fly design of variable selection strategies by deploying a supervised approach. The aim is to build a surrogate function able to mimic strong branching with lower computational cost by solving a learning-to-rank problem common in machine learning. A similar approach can be seen in Hansknecht et al. [78]. They analyze the branching problem to solve the time-dependent travel salesman problem, in which a ranking surrogate function decides on variable branching. In the seminal work of Gasse et al. [79], MILP

problems are encoded as a graph neural network [80] model. It leverages the variable-constraint bipartite graph representation of integer linear programming. This may emancipate the need for hand-craft feature engineering.

Learning algorithms have also been developed for the problem of node selection in branch and bound algorithms as a way to trade-off between depth first and breadth first strategies. In this context, we can mention the research works He et al. [81], Song et al. [82] using imitation learning to mimic quickly the Oracle decisions in node selection.

Recently, learning algorithms have been employed to guide solvers to cut selection. Tang et al. [83] present a reinforcement learning formulation for an intelligent adaptive selection of Gomory's cutting planes. Turner et al. [84] develop a reinforcement learning framework realized via graph neural network (see also [79]) for learning parameters decisive in cut selection. Baltean-Lugojan et al. [85] deploy a neural network model to select subsets of cutting planes to approximate the semidefinite constraints used in the relaxation formulation of quadratic optimization problems. Cutting ranking and selection are also implemented in Huang et al. [86] by learning a scoring function based on instance-specific features. The novelty introduced in that paper is the implementation of a score not applied to single cuts but to *bags* of cuts.

The other major direction in combining learning algorithms with optimization has been specified to learn directly the solution of mathematical optimization problems. This is known as end-to-end learning [87]. Most of these methods in combinatorial optimization were devoted to problems where feasibility is less of an issue than optimality.

Several studies have focused on Traveling Salesman Problems (TSP) and Vehicle Routing Problems (VRP). Kool et al. [88] implement a pointer network with attention layers to learn strong heuristics for different combinatorial optimization problems. Emami and Ranka [89] use an actor-critic neural network to devise a reinforcement learning agent to solve TSPs. Problems with harder constraints, such as scheduling problems with strict resource, time, or regulatory constraints, particularly handling simultaneously nonlinearity and integrality, had seen less focus on end-to-end learning methods, but interest has been growing. Nair et al. [90] address MILP problems with general structure by using two parallel neural networks. One generates multiple partial assignments for integer variables in a MILP problem, which allows solving a smaller sub-MILP by an off-the-shelf MILP solver. The other neural network is trained to imitate a full strong branching policy. The idea of partial assignment is also employed in Ding et al. [91]. The MILP instances are represented as tripartite graphs where the three sets of nodes are the variables, the constraints, and the objective function. A subset of variables deemed as *stable* by the learning model are

fixed to their predicted values, and a branch and bound algorithm solve the remaining sub-MILP. Yilmaz and Buyuktahtakin [92] devise a bi-directional LSTM framework for decision-making in single-item capacitated lot-sizing problems. The learning algorithm is a regression problem that predicts binary variables with a value in the interval $[0, 1]$. Based on a predefined percentage, the binary variables are fixed to the closest integer value, and a MILP solver solves the consequent sub-MILP. For an arbitrary given percentage, that is, for a fixed subset of binary variables, the sub-MILP might be infeasible. It is observed that the number of infeasible instances increases by considering larger instances. Splitting the subset of binary variables into fixed and unfixed is also done in Masti and Bemporad [93]. In fact, the chance of getting a feasible configuration increases in this framework.

Applications in the domain of energy also exist. Anderson et al. [94] design a generative adversarial architecture for a deep neural network to learn directly binary decision variables for the MILP formulation of decision-making in gas networks. The consequent predicted solution is used as a warm start for the MILP solver. The physical constraints are softened in the mathematical model, i.e., their violation is only penalized in the objective function. The learned solution is thus de facto feasible in this model. This penalization is also employed in the AC Optimal Power Flow (OPF) context. The OPF problem is a nonlinear and nonconvex (but continuous) optimization problem that determines the generator setpoints for power and voltage, given a set of load demands. It is often solved repeatedly under various conditions in real-time or large-scale studies. Neel Guha et al. [95] propose an end-to-end learning implemented as a neural network mapping a given input, a load profile at demand nodes, to an optimal voltage magnitude and active power injections of the generators. The rest is again retrieved by a MILP solver. Zamzam and Baker [96] propose learning not all variables, but a subset of them (the voltage and active power). To increase the chance of feasibility, the output of the neural net is restricted to the given box constraints, and then, for a predicted value, the associated NLP is solved. The paper [97] proposes a new approach to predicting the OPF of an electrical power system. The deep learning model used in the paper learns to predict the OPF solution by minimizing the Lagrangian dual function while adding a penalty term to the objective function.

More relevant to our work, the authors in [98] propose a learning approach to guide an alternating direction method of multipliers (ADMM) search based on the network partition. They propose a novel way to learn the consensus parameters in the ADMM algorithm using machine learning. The machine learning model is trained on a dataset of previously solved OPF problems. The model learns to predict the consensus parameters that lead to fast convergence of the ADMM algorithm. The linear separability of the model, in this case, guarantees the convergence of the algorithm to a feasible solution. Furthermore, as the model is continuous, the

predictions cover both primal and dual optimal solutions, and the predicted dual solutions serve to initialize the penalties in ADMM.

Chapter 9

An Alternating Direction Method for Pump Scheduling

9.1 Principle of ADM and state-of-the-art

Initially conceived in the mid-20th century [99], the Alternating Direction Method (ADM) has undergone various modifications and has found applications across diverse fields [100, 101]. The core principle of ADM involves decomposing a high-dimensional or intricate optimization problem into a sequence of simpler sub-problems. ADM is a powerful approach for solving complex optimization problems by decomposing them into more manageable sub-problems. The method alternates between optimizing different subsets of variables, holding one subset constant while optimizing the other [100]. This decoupling feature enables ADM to tackle large-scale problems efficiently, making it well-suited for parallel computing and distributed systems [101].

The classical ADM considered in [102] is a solution method for optimization problems with a clear partition of the decision variables in two sets, as in the following mathematical program:

$$(P_S) : \min\{f(u, v) : u \in \mathcal{U}, v \in \mathcal{V}\},$$

where the objective function f is convex, and it is optimized over two disjoint feasible sets. In [103], the authors consider the extension to biconvex objective functions.

The main steps of ADM are presented in Algorithm 7.

Algorithm 7 Standard Alternating Direction Method

Input initial values $(u^0, v^0) \in \mathcal{U} \times \mathcal{V}$

Output a partial minimum solution (u^*, v^*) of (P_S)

1: **while** (u^i, v^i) is not a partial minimum **do**:
2: Compute: $u^{i+1} \in \arg\min_u \{f(u, v^i) : u \in \mathcal{U}\}$
3: Compute: $v^{i+1} \in \arg\min_v \{f(u^{i+1}, v) : v \in \mathcal{V}\}$
4: **end while**

In [102], the authors state that for two disjoint compact sets $\mathcal{U}$ and $\mathcal{V}$, and having a continuous objective function, the sequence $(u^i, v^i)_{i\geq 0}$ may have several convergent subsequences. Any limit point of the alternating approach is a point (u^*, v^*), called a partial minimum solution of (P_S), in a way that:

$$\begin{aligned} f(u^*, v^*) &\leq f(u, v^*) \quad \forall u \in \mathcal{U} \\ f(u^*, v^*) &\leq f(u^*, v) \quad \forall v \in \mathcal{V}. \end{aligned}$$

Once the algorithm meets such criteria, alternating between two (or several) subproblems is terminated. The concept of partial minimum is a rather weak property. The partial minimum may not even be a local minimum. If either the optimization of the first or second subproblem has a unique solution, then the ADM algorithm converges to a partial minimum. This is given as a theorem in [53].

Under certain further assumptions, stronger implications can be derived from the result of the algorithm. If the objective function is continuously differentiable, then the algorithm converges to a stationary point. If the objective function and the constraint set are convex, then the partial minimum is the global one [103].

Let us assume now that the problem is provided with constraints that make it nonseparable (or quasi-separable), that is

$$(P) : \min\{f(u, v) : g_i(u, v) = 0, h_i(u, v) \geq 0, u \in \mathcal{U}, v \in \mathcal{V}\}.$$

Then, the coupling constraints impede the implementation of the ADM algorithm in its original format. The ADM algorithm has been extended for nonseparable and quasi-separable cases by slight changes in the formulation. This has been realized through the well-known Alternating Direction Method of Multipliers (ADMM) initially proposed for convex problems.

The ADMM can be seen as an extension of the ADM approach and Lagrangian multipliers. While it decouples the two sets of constraint sets, it enforces a consensus (coupling constraints) between u and v, which allows ADMM to be used for distributed optimization. This is realized by incorporating both penalty terms and

Lagrange multipliers. The theoretical convergence of the ADMM for the strict convex subproblems and affine coupling constraint is illustrated in [104]. Boyd et al. [105] show the possibility of parallelization of the optimization problem and the scalability for large-scale problems and ADMM application in solving machine learning problems. The possibility to solve each subproblem independently at each iteration and the straightforward update policy for Lagrangian multipliers of the linking constraint absorbed lots of attention in large-scale convex problems with relatively sparse representation. Recently, research has been conducted on extending the convergence for nonconvex and nonsmooth subproblems. In [106], the theoretical convergence is proved for nonconvex and nonsmooth subproblems while two sets are linearly split. However, to the best of our knowledge, there exists no general proof for nonconvex optimization problems with nonlinearly coupled ADMM.

The other variant of ADM is based on the penalization of the coupling constraints. The Penalized Alternating Direction Method (PADM) [53, 107] is the ADM applied after penalizing the coupling constraints, that is,

$$(P_{PADM}) : \min\{f(u,v) + \sum_{i=1}^{m} \rho_i |g_i(u,v)| + \sum_{i=1}^{p} \mu_i [h_i(u,v)]^- : u \in \mathcal{U}, v \in \mathcal{V}\},$$

where the penalized objective function is denoted by $\phi(u, v; \rho, \mu)$. Algorithm 8 illustrates the mechanism of PADM.

Algorithm 8 Penalized Alternating Direction Method

Input initial values $(u^{0,0}, v^{0,0}) \in \mathcal{U} \times \mathcal{V}, (\rho^0, \mu^0) \in \mathbb{R}^m \times \mathbb{R}^p_+$
Output a partial minimum solution (u^*, v^*) of (P)

for $k = 0, 1, ...$ **do**:
  while $(u^{k,i}, v^{k,i})$ is not a partial minimum **do**:
    Compute: $u^{k,i+1} \in \arg\min_u \{\phi(u, v^{k,i}; \rho^k, \mu^k) : u \in \mathcal{U}\}$
    Compute: $v^{k,i+1} \in \arg\min_v \{\phi(u^{k,i+1}, v; \rho^k, \mu^k) : v \in \mathcal{V}\}$
    $i \leftarrow i + 1$
  end while
  update penalty terms μ^k, ρ^k
end for

PADM was initially introduced in [53] for the power-constraint gas problem. While the conventional method, such as using MILP relaxation, is reported to help solve gas transport problems, adding power constraints impedes finding satisfactory results. The authors show that the ADM is a promising approach to acquiring feasible solutions. The PADM is also employed as a primal heuristic for supply chain problems [108], bilevel optimization [107], and gas network [109]. Despite ADMM, in PADM,

the partial minimum of (P_{PADM}) concept may violate some coupling constraints; therefore, the convergence of the inner iteration does not necessarily lead to a feasible solution. On the other hand, the convergence proof in the PADM framework requires weaker assumptions. As long as the solution of one of the subproblems is unique, the inner loop converges to a partial minimum.

The success of the algorithm to find a good solution depends heavily on two features. First, the updating policy of the penalty terms. The other crucial point is the initial point to launch the algorithm.

9.2 Adaptation to the pump scheduling problem

We now present our application of the Alternating Direction Method to the pump scheduling problem, modeled as follows:

$$(\mathcal{P}') : \min_{x,q,H} \sum_{t\in\mathcal{T}} c_t(x_t, q_t) = \sum_{t\in\mathcal{T}}\sum_{a\in\mathcal{A}} (c_t^0 x_{ta} + c_t^1 q_{ta}) \tag{9.1}$$

$$\text{s.t: } q_t \in \mathcal{E}'(H_t, D_t, x_t), \qquad \forall t \in \mathcal{T} \tag{9.2}$$

$$q_{tj}^{\mathcal{C}} = \sigma_j(H_{(t+1)j} - H_{tj}), \qquad \forall j \in \dot{\mathcal{C}}, \forall t \in \mathcal{T} \tag{9.3}$$

$$\underline{H}_{tj} \leq H_{tj} \leq \overline{H}_{tj}, \qquad \forall j \in \mathcal{C}, \forall t \in \overline{\mathcal{T}} \tag{9.4}$$

$$x_t \in \mathcal{X}_t \subseteq \{0,1\}^{\mathcal{A}} \qquad \forall t \in \mathcal{T}. \tag{9.5}$$

The main difference with the original bilevel formulation (2.16)-(2.20) lies in the constraints (9.5). In this model, we ignore the time-dependent constraints related to pump aging, but we may enforce interdependency relations between the activity of valves and pumps on each period. The head loss variables h_t are also dropped as they are not needed in this formulation. Here, the set $\mathcal{E}'(H_t, D_t, x_t)$ is defined as the projection of $\mathcal{E}(H_t, D_t, x_t)$ from (q_t, h_t) to q_t.

The proposed algorithm has a more general scope than pump scheduling. In [110], we describe the algorithm in the context of a discrete-time dynamic optimal control problem with discrete control and storage. The variable split concerns the control variables and the storage state variables. They respectively correspond to the scheduling/flow variables (x, q) and the tank level variables H in pump scheduling. The property of easily solving the equilibrium subproblems is given as an additional assumption on the control problem in question. This is formalized below.

Assumption (A0). The steady-state sub-problems with known initial state variables

$$(\mathcal{P}_t(H_t)) : \min_{x_t, q_t}\{f_t(x_t, q_t) \mid q_t \in \mathcal{E}'(H_t, D_t, x_t), x_t \in \mathcal{X}_t \subseteq \{0,1\}^{\mathcal{A}}\}$$

are easy for all $t \in \mathcal{T}$ and $H_t \in \mathbb{R}^{\mathcal{C}}$, and for any linear function f_t.

In the context of pump scheduling, the subproblems $(\mathcal{P}_t(H_t))$ are linearly separable, when considering the network partition $(\mathcal{G}^b)_{b\in B}$ along the tanks, as described in Section 2.3.3:

$$(\mathcal{P}_t(H_t)) : \sum_{b\in B} \min_{x_t^b, q_t^b}\{f_t^b(x_t^b, q_t^b) \mid q_t^b \in \mathcal{E}'(H_t^b, D_t^b, x_t^b), x_t^b \in \mathcal{X}_t^b \subseteq \{0,1\}^{\mathcal{A}^b}\}.$$

As the subnetworks $\mathcal{G}^b$ usually contain a reasonable number $|\dot{\mathcal{A}}^b|$ of pumps and valves, we can envisage enumerating all possible configurations $X \in \mathcal{X}_t^b$, and evaluating them, independently, by computing (e.g., with a Newton method) the unique solution in $\mathcal{E}'(H_t^b, D_t^b, X)$, which we denote $q(H_t^b, D_t^b, X)$. The steady-state subproblem is then solved as:

$$P_t(H_t) : \sum_{b\in B} \min_{X\in\mathcal{X}_t^b} f_t^b(X, q(H_t^b, D_t^b, X)).$$

Hence, assumption **(A0)** holds in the context of pump scheduling when the number of pumps and valves is limited in every subnetwork.

This assumption also holds in other contexts of dynamic control problems. Typically, $(\mathcal{P}_t(H_t))$ involves simulating $\mathcal{E}'(H_t, D_t, X)$ over each allowed steady-state decision $X \in \mathcal{X}_t$, and knowing the initial state H_t while ignoring the outcome (and the final state H_{t+1}). Because the *nonlinear network equilibrium problem* $\mathcal{E}'$ arises in many other contexts, ranging from electric circuits or thermodynamic systems to traffic congestion (see [13, p. 350]), we think that model $(\mathcal{P}')$ with assumption **(A0)** is quite general. Thus, there is broad applicability for the decomposition method to be presented below.

The optimization problem $(\mathcal{P}')$ can be decomposed temporally after dualizing the storage time-coupling constraints (9.3) as in Lagrangian relaxation or after penalizing their violations as in the following model. Given ℓ_1 penalty and multipliers $\rho \in \mathbb{R}_+^{kT}$, we define

$$(L_\rho) : \min_{x,q,H}\{l(x, q, H, \rho) : (9.2), (9.4), (9.5)\},$$

$$\text{with } l(x, q, H, \rho) = \sum_{t\in\mathcal{T}} \big(c_t(x_t, q_t, H_t) + \sum_{j\in\dot{\mathcal{C}}} \rho_{tj} d_{tj}(q, H)\big),$$

$$\text{and } d_{tj}(q, H) = |H_{(t+1)j} - (H_{tj} + q_{tj}^{\mathcal{C}})| \quad \forall t \in \mathcal{T}, j \in \dot{\mathcal{C}}.$$

Still, such relaxed models, even if separable as T independent subproblems, may remain difficult to solve because of the complexity of optimizing over each subsystem $\mathcal{E}'(H_t, D_t, x_t)$ when H_t is variable.

Instead, we propose solving approximately (L_ρ) with a variable-splitting approach based on a storage/control split for leveraging Assumption **(A0)**. The penalty ρ is updated iteratively and the algorithm stops at the first feasible solution of $(\mathcal{P}')$ within a given tolerance on constraints (9.3) and (9.4). Conceptually, the algorithm iterates over the tank level profiles H, which are feasible regarding the capacities, and attempts to derive a matching schedule (x, q) by gradually reconciling the storage state $H_t + q_t^{\mathcal{C}}$ after time t, with the state assumed H_{t+1} at time $t+1$.

The steps of the proposed algorithm are given in Algorithm 9.

Algorithm 9 Partial storage/control splitting for $(\mathcal{P}')$

1: **Input:** $i = 0$, tank profiles $H^0 \in \mathbb{R}^{\mathcal{T}\dot{\mathcal{C}}}$, penalty ρ^0, tolerance $\varepsilon, \varepsilon' > 0$
2: **Output:** a feasible solution (x, q, H) of $(\mathcal{P}')$
3: **for** $k = 0, 1, \ldots$ **do**:
4: **while** $\|H^{i+1} - H^i\|_\infty \geq \varepsilon'$ **do**:
5: $(x^{i+1}, q^{i+1}) \in \arg\min_{(x,q)} \{l(x, q, H^i, \rho^k) : (9.2), (9.5)\}$
6: $H^{i+1} \in \arg\min_H \{l(x^{i+1}, q^{i+1}, H, \rho^k) : (9.4)\}$
7: **if** $d_{tj}(q^{i+1}, H^{i+1}) < \varepsilon \ \forall t \in \mathcal{T}, j \in \dot{\mathcal{C}}$ **then**:
8: **return** $(x^{i+1}, q^{i+1}, H^{i+1})$
9: **end if**
10: $i \leftarrow i + 1$
11: **end while**
12: update the penalty term ρ^k
13: **end for**

This algorithm has the same framework as PADM when considering the variable split (x, q) and H. However, we do not dualize/penalize here the nonlinear coupling constraint (9.2). Indeed, the first step at each iteration (Line 5) solves (L_ρ) with respect to the control variables (x, q) and, according to Assumption **(A0)**, solving this nonlinear problem is easy. In return, solving (L_ρ) with respect to the state variables H in the second step is much harder as this corresponds to a sequence of possibly infeasible inverse problems, namely: find the initial state H_t for a given control (x_t, q_t) for each period t. Instead, we propose to relax the coupling equilibrium constraints (9.2) in the second subproblem (Line 6) and keep the storage capacities (9.4) as the only constraints to satisfy. It results in a simple linear program with T variables and only box constraints.

As we deliberately maintain the nonlinear coupling relation (9.2) in the first subproblem, we lose the guarantee, in general, provided by PADM, to converge to a stationary point. The known theoretical convergence results for ADMM do not hold either with nonconvex-constrained variable sets. However, in doing so, we take advantage of assumption **(A0)** to not impoverish the first subproblem. While possibly losing mild theoretical guarantees (the convergence to solutions that may not always be feasible), we expect to gain strong practical results.

We observed that the efficiency of this algorithm is highly dependent on the initial tank level profiles H^0. For this reason, as we show in the next chapter, we design a deep learning model to predict near-feasible near-optimal profiles.

Chapter 10

A Supervised Deep Learning Model

10.1 A deep learning approach

Deep learning (DL) has significantly impacted various domains, from computer vision to natural language processing. The key feature that sets deep learning apart from shallow machine learning algorithms is the ability to learn hierarchical features from raw data automatically [111]. In contrast, traditional machine learning methods often rely on manual feature engineering, which is not only labor-intensive but may also introduce biases and limitations [112]. Deep learning models, neural networks (NN) with many layers, can learn intricate patterns and structures. This capability is afforded by the nonlinear transformations and hierarchical feature learning across layers [1]. Backpropagation[1] efficiently computes gradients of the loss function with respect to the model parameters, enabling optimization algorithms like stochastic gradient descent to update the weights in the network [113]. The significant challenge in DL is the issue of overfitting, especially given that these models often have numerous parameters. This problem becomes even more pronounced when the amount of training data is limited [114]. Techniques such as dropout, ℓ_1,ℓ_2 regularization, and

[1]Backpropagation, short for 'backward propagation of errors,' is the training algorithm used in neural networks for adjusting the weights. In the output layer, the error between the predicted and the true values is computed. Subsequently, this error signal is propagated backward through the lower layers of the network. In this context, backpropagation can be seen as a descent algorithm that strives to minimize the error with each iteration. The network weights are modified by the learning algorithm in a manner that decreases the error along a descent path.

early stopping are commonly employed to mitigate overfitting. Furthermore, due to their complexity and depth, deep learning models typically require a large volume of labeled data for training. This data-hungry nature of deep learning models can be a significant limitation in scenarios where collecting or labeling data is expensive or impractical.

Despite these challenges, DL offers a unique advantage, especially for complex tasks like predicting the solution of combinatorial optimization problems. Unlike traditional methods that often require a series of hand-engineered steps, deep learning models can learn to approximate complex decision-making rules directly from raw data. This is particularly attractive in combinatorial optimization problems, where the problem size is usually large and computationally challenging [115].

The prediction of an optimal solution differs from typical machine learning problems, in which the target is a single scalar or category. In our case, the output to be predicted (i.e., the optimal tank profiles) can be seen as a (temporal) sequence. As mentioned, each instance differs from the other via tariff and demand profiles. The learning hypothesis may need to capture the local similarities in the input data: the repetitive patterns in small consecutive periods in demand and tariff profiles occurring in different instances. However, the same pattern occurrence in different periods could produce different results. Thus, the learning algorithm should consider both local patterns and their temporal dependencies. The shallow machine learning hypothesis usually cannot capture the complexity of the problem. The feedforward architecture requires flattening the input data, which might fade the importance of temporal dependencies.

Given this context, the focus of this chapter is to introduce a novel DL architecture for the pump scheduling problem, integrating the capabilities of Long Short-Term Memory (LSTM) networks for sequence understanding and Convolutional Neural Networks (CNNs) for spatial pattern recognition [1]. In what follows, we present the rationale for using these artificial neural networks. In Appendix A, we provide technical details of the DL architecture.

LSTMs are a type of Recurrent Neural Network (RNN) that are well-suited for tasks involving sequential data, such as natural language processing and time-series forecasting. RNNs are neural networks having feedback loops, which allow them to learn temporal dependencies in the data. However, RNNs can suffer from a problem known as the vanishing gradient[2] problem, which makes it difficult to train RNNs

[2]The vanishing gradient problem is a challenge in deep learning where gradients (derivatives of the loss function with respect to model parameters) become exceedingly small as they are backpropagated through the layers of a neural network during training. When gradients become too small, it hinders the ability to update the network weights effectively, leading to slow or stagnated training and difficulty in capturing long-range dependencies in sequential data.

on long sequences of data. LSTMs address the vanishing gradient problem by using a gating mechanism to control the flow of information through the network. The gating mechanism consists of three gates: the input gate, the forget gate, and the output gate. These gates control how much information is passed into the LSTM cell, how much information is forgotten from the LSTM cell, and how much information is output from the LSTM cell.

Convolutional Neural Networks (CNNs) are primarily recognized for their efficacy in image data recognition, classification, and other computer vision tasks. Yet, they can also be utilized for sequential data by focusing on the temporal dependencies within the sequence. Typically, one-dimensional convolutional (1Dconv) layer NNs are employed in sequential data. Each 1Dconv consists of several kernels (filters) responsible for learning features in the input data. A kernel, i.e., a weight matrix, slides across the period of the input sequence, identifying local temporal patterns using the element-wise inner operation between kernel weights and input data. Indeed, multiple kernels can extract different types of temporal features. Each kernel sliding over input data can have a different window size. This variety of sizes and the number of kernels will produce different feature maps, enabling the model to recognize various patterns in the sequence.

10.2 Learning the near-optimal state profiles

We design a deep learning architecture to predict multiple (sub-)optimal state profiles (the tank levels) for each instance of the MINLP ($\mathcal{P}'$). The training of the model is devised in the supervised learning framework. Our deep learning algorithm aims at building a hypothesis function $\mathcal{H}$ mapping to each input feature, i.e., demand and tariff profiles $(D, C) \in \mathbb{R}^{\mathcal{TS}} \times \mathbb{R}^{\mathcal{T}}$, its target, i.e., the tank level profile $H(D, C) \in \mathbb{R}^{\mathcal{TC}}$ in an optimal solution of the problem ($\mathcal{P}'$) with input (D, C). The map is built, given a precomputed collection $\{((D_i, C_i), H_i)\}_{i=1}^{N}$ of input/target tuples with $H_i = H(D_i, C_i)$ $\forall i$, by approximately minimizing a loss function, which is the mean square error between each element of the tank profiles as commonly used in regression problems:

$$\mathcal{L}_{loss}(\mathcal{H}(D, C), H(D, C)) = \frac{1}{N} \sum_{i=1}^{N} ||\mathcal{H}(D_i, C_i) - H_i||_2^2.$$

To fulfill this aim, we blend the capabilities of CNNs and Bidirectional LSTM Networks (Bi-LSTM) [116]. The combination of these two kinds of neural networks can be found in several works, e.g., [117, 118]. The capability of Bi-LSTM lies in

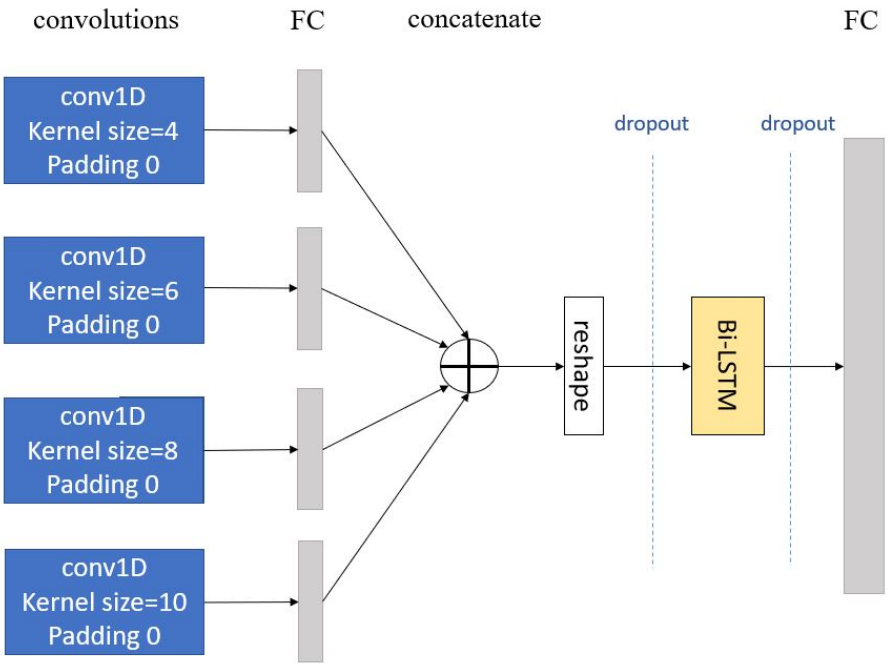

Figure 10.1: Simplified scheme of the proposed CNN-LSTM architecture.

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 $\mathcal{T}$.

Our CNN comprises parallel convolutional layers with different kernel size windows, with their outputs exclusively linked to neighboring areas within the input (i.e., demand and tariff profiles). This arrangement is accomplished by sliding a kernel across the input and the product between the weight matrix and the input data. This design enables the model to acquire filters to identify distinct patterns within the input dataset [1, 119].

Several conv1Ds with different kernel sizes are located parallel to each other with zero padding. The kernel sizes (here $4, 6, 8$, and 10) and the number of kernels of the conv1D layers are chosen relative to the scheduling time horizon (i.e., here we experimented over scheduling problem with $T = 12$). We have considered 32 kernels for each conv1D. To consider the temporal dimension of the target (the output target is related to the number of periods and the number of tanks), the output of the CNNs layers are passed through a hidden layer with ReLU activation functions, and their outputs are concatenated, reshaped and fed into a Bi-LSTM (see Figure 10.1. In the output layer, the prediction H_{tj} for all $t \in \mathcal{T}$ and $j \in \dot{\mathcal{C}}$ results from a fully connected layer with a linear activation function placed after the Bi-LSTM unit.

10.3 Generating multiple starting points for the decomposition algorithm

As mentioned, we propose to restart the decomposition algorithm (Algorithm 9) from different predicted points. To generate multiple warm starting points, we employ Monte Carlo (MC) dropout [120]. As shown in [120], MC dropout approximates Bayesian variational inferences[3].

The dropout method is widely recognized as an effective regularization technique [114]. It works by randomly dropping out (setting to zero) a fraction of the hidden units of a neural network model during training. Each dropout generates a model. In alignment with [120], we apply dropout not only during training but also during the testing phase. The dropout layers are placed both before the Bi-LSTM unit and just before the last fully connected layer that yields the outputs. This approach yields a collection of distinct models arising from masked neurons. As a result, for a given input pair (D, C), we obtain multiple outcomes $\hat{H}$ one from each model.

10.4 Scaling to extrapolate the missing dataset

The main drawback of supervised learning for predicting the solutions to an optimization problem is the need of a sufficient quantity of data to train the learning model. The limitation arises from the computational complexity of the problem and from the requirement to solve the optimization problem on a significant number of instances to build the dataset from which the learning model will be trained (and tested).

In the application context of pump scheduling, we argue that a dataset can be generated from the history of the daily operations of a given hydraulic network. However, this does not hold if no history is available or if, as in the context of this thesis, we consider a data-based alternative to the theoretical MINLP model ($\mathcal{P}'$) instead of the *digital twin* of a real system. Hence, in our case, for generating a dataset, we need to compute optimal solutions of the MINLP ($\mathcal{P}'$) for a large number of instances. However, as shown in our numerical results of Chapter 7, even acquiring a feasible solution is in jeopardy when considering large scheduling horizons (e.g.,

[3] A Bayesian model is a statistical model incorporating Bayesian probability theory to represent uncertainty in data and make probabilistic inferences. These models usually have a prohibitive computational cost, and then, in general, approximations have to be considered. Dropout neural networks are an example of this approximation.

$T = 48$). This limitation is also indicated in [91, 121] and its influence on the learning process.

A recent approach to generalize the learning algorithm for unseen structures consists of using graph neural networks in the context of MILP optimization. In this framework, the MILP structure is encoded into a bipartite [79, 90] (or tripartite [91]) graph with variables on one side and constraints on the other side. This captures the complexity of MILPs well and makes it possible to learn from MILPs with different structures. In particular, for handling travelling salesman problems (TSP) of different scales, the authors in [118] have developed a hierarchical reinforcement learning via graph pointer network to map from the state of the TSP to the optimal solution. Implementing graph neural networks for MINLP optimization is not as direct as for MILPs.

Another option is to emancipate from having pre-solved instances and, consequently, from a supervised learning framework. The self-supervised learning approach is a different path to train neural networks directly without using pre-solved instances [122, 123].

Finally, the learning model can simply be trained on an approximate dataset of approximate solutions computed from an approximate model. In our context, this approach is all the more meaningful in that the solutions obtained from the most accurate learning models might not be feasible anyway (contrary to the TSP, for example, where any permutations of the graph nodes are feasible solutions). When learning from an approximate model, the prediction error accumulates the inaccuracies of the learning model and the approximate model.

Regarding the pump scheduling problem, our numerical experiments showed that the MILP approximations (based either on a polyhedral convex relaxation or a PWL linear approximation of the nonlinear constraints) are neither accurate nor much easier to solve, in particular, for the largest instances (i.e., with large horizons) for which generating a dataset is itself a challenge. For handling these instances, we thus propose to build our approximate dataset with a different approach, i.e., by exploiting the property of elasticity of the scheduling horizon.

Indeed, as we consider daily schedules, the number of periods T only refers to the choice of temporal discretization. Scheduling using a coarse temporal resolution (e.g., $T = 12$) is more tractable, but it has little flexibility. Suppose a hydraulic network includes powerful pumps, tanks with narrowed capacities, or service nodes with a highly dynamic demand. In that case, there is no feasible schedule if the temporal resolution is too low (see, for instance, our numerical results on the Poormond

network). Conversely, a fine temporal discretization (e.g., $T = 48$ periods of a half-an-hour time step) describes better the actual dynamic of the hydraulic networks, but the mathematical models become intractable.

Thus, we propose to build the dataset of a given hydraulic network $\mathcal{G}$ by only considering coarse-time resolution pump scheduling instances (with typically $T = 12$) as described in Chapter 2. The dataset $\mathcal{D}_{\mathcal{G}} = ((D_i, C_i), H_i)_{i=1}^{N}$ is thus made of N instances of the pump scheduling problem on the network $\mathcal{G}$. Each instance $i \in \{1, \ldots, N\}$ is defined by its input data $(D_i, C_i) \in \mathbb{R}^{\mathcal{TS}} \times \mathbb{R}^{\mathcal{T}}$. Note that tariff and demand profiles are originally provided having a fine-time discretization (a half-hour time step). To use this data in the learning model, we first must resample these input data by averaging the consecutive time steps. The target $H_i \in \mathbb{R}^{\mathcal{T}\dot{\mathcal{C}}}$ corresponds to the tank level profiles in an optimal solution of $(\mathcal{P}'(D_i, C_i))$ computed using our branch-and-bound method described in Chapter 4 of Part II. Our deep-learning model is then trained on this dataset alone.

To apply (and experiment) our decomposition to the problem $(\mathcal{P}')$ on an instance $(D', C') \in \mathbb{R}^{\mathcal{T}'\mathcal{S}} \times \mathbb{R}^{\mathcal{T}'}$ having a finer-time discretization $|\mathcal{T}'| > |\mathcal{T}|$, we need demand and tariff profiles (as exogenous input values) and a tank profile (as a starting value) definite over $\mathcal{T}'$. Regarding the initial storage profile, say $H' \in \mathbb{R}^{\mathcal{T}'\mathcal{C}}$, we resample and linear interpolate the tank profile predicted by the deep learning model. As explained above, this model produces multiple approximated profiles of the (sub)-optimal tank profiles. All of these are then downscaled in time and used to initialize Algorithm 9.

Chapter 11

A Physics Informed Deep Learning Model

11.1 A supervised penalty approach

In supervised learning, the classical loss function indicates the distance between predicted and ground truth values. As mentioned in the previous chapter, the (partial) solution predicted by the learning architecture does not guarantee to satisfy the hard constraints of the pump scheduling problem. While some simple restrictions on variables (e.g., box constraints) can be enforced within a DL architecture, in general, there is no straightforward way to integrate all integer and nonlinear constraints within the architecture [123]. The supervised penalty approach (e.g., [124]) is a method for end-to-end learning of optimization problems that integrates the constraints within the learning model. It works by adding penalty terms to the loss function that penalizes violations. The penalty terms are typically weighted by hyperparameters that control the trade-off between minimizing the prediction error and feasibility. During training, the neural network learns to adjust its weights so that the combination of distance of the predicted and ground truth targets and violations are minimized. An approach to integrate optimization problems, in particular, linearly constrained quadratic programming, as a layer in neural networks (OptNet) is introduced in [125]. In [124, 126], a physics-informed neural network (NN) is implemented for the optimal power flow. The loss function of an NN architecture is defined by adding to the classical square error, also a violation measure of the KKT conditions, that has to be minimized. Their numerical experiments, which required High-Performance Computing (Intel Xeon E5-2650 v4 processor and 256 GB RAM) for training their model,

show that the physics-informed neural network is more likely to predict solutions with smaller infeasibility violations. In [123], a framework to predict and correct the solution for continuous nonlinear optimization problems is proposed. The algorithm first predicts a partial solution to an optimization problem. Then, the full set of variables is retrieved by the satisfaction of the equality constraints. Finally, the partial solution is corrected to satisfy the inequality constraints while continuing to satisfy the equality ones.

The major difference in our work is the presence of a discrete feasible set, which impedes the differentiability through backpropagation; this deviates the work from some continuous optimization frameworks, such as [125]. Embedding integrality constraints may make the backpropagation intractable.

Alternatively, in this work, we propose a continuous surrogate model for network analysis problems to embed simultaneously the underlying structure of the problem and not hinder the differentiability of the neural network. To some extent, this is analogous to [127], in which the authors suggest continuous relaxation of the integral constraints and then rounding the fractional solution. However, we propose a new type of relaxation and tightening, which takes leverage from the one-to-one relationship between binary decision variables and storage states in the fixed speed pump scheduling problem, presented as an equilibrium problem in Section 2.3.1.

Given the demand and tariff profiles, we aim to build a model that can accurately predict a (sub-)optimal solution (as before, the state variables). However, the approach proposed here integrates the physical constraints of the problem into the learning algorithm to better guide the output of the learning model. Instead of explicitly entering the constraints of the extended network analysis (see Section 2.3.2), which involves discrete variables, we develop a continuous surrogate model realized via a neural network to map from control arc configurations X to tank storage H.

We design a DL model which includes two main blocks. The first is to predict the binary variables via classification given the load and demand profiles. The following block builds a map from the predicted binary values, together with the load profile, to the tank profile. It is in this second block that physical constraints are taken into account.

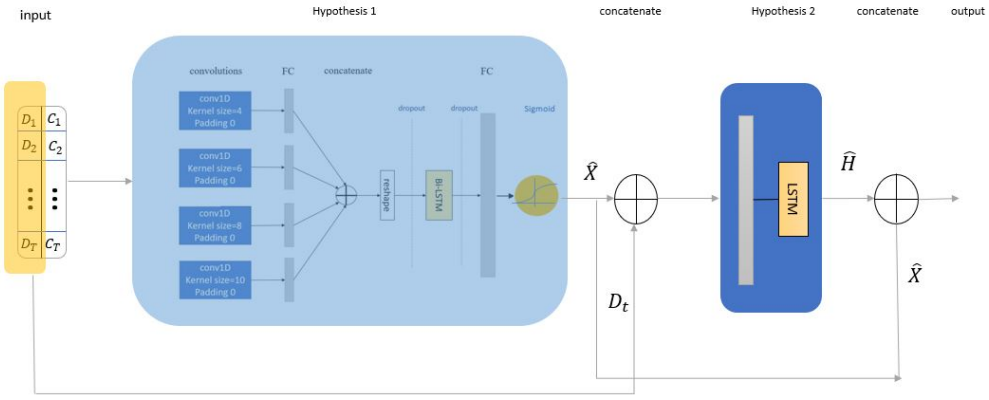

Figure 11.1: Structure of physics informed model

11.2 Training and architecture

To construct a learning hypothesis capturing the constraints of the pump scheduling problem, we propose a neural network consisting of two main sub-blocks. The first block of the proposed model builds a hypothesis function $\mathcal{H}_1$ mapping each input feature, i.e., load and tariff profiles $(D, C) \in \mathbb{R}^{\mathcal{TS}} \times \mathbb{R}^{\mathcal{T}}$, to its target, i.e., the configuration of the pumps $X \subset [0,1]^{\dot{\mathcal{A}} \times \mathcal{T}}$. The second sub-block is responsible for mapping the predicted configuration to the associated profile of the tanks $H(L, X) \subset \mathbb{R}^{\dot{\mathcal{C}} \times T}$. This is done by building a hypothesis function $\mathcal{H}_2$.

For the first hypothesis $\mathcal{H}_1$, we have modified the CNN-LSTM architecture previously developed. The binary configuration is predicted using a supervised classification method. Therefore, we replaced the shape of the output layer with a sigmoid activation function to produce values inside the interval $[0, 1]$, and the number of outputs is based on the number of periods (e.g., 12 for VZ12) and control arcs (e.g., 4 for VZ12).

The second hypothesis $\mathcal{H}_2$ is a surrogate model realized as a neural network to map each configuration to its associated tank profile. The hypothesis $\mathcal{H}_2$ approximates the underlying structure of the pump scheduling problem: it replicates the extended analysis problem, where the demand profile and the pump configurations are its input and the tank profile is its output. The second block consists of LSTM units and feed-forward layers with ReLU activation functions. The recurrent architecture of the model is analogous to the dynamic structure of the extended analysis. In Figure 11.1, the way that these models are connected is depicted.

In a typical supervised learning framework, a loss function for this problem can be a combination of binary cross entropy (for the approximation of the binary variables) and quadratic error (for the approximated tank profile). However, for our problem, we define the following loss function:

$$\mathcal{L}_{loss} = \mathcal{L}_{CE}(X, \hat{X}(L, C)) + p[\hat{H}(L, C) - \underline{H}]^{-} + p[\overline{H} - \hat{H}(L, C)]^{-}, \tag{11.1}$$

where $\mathcal{L}_{CE}$[1] denotes the binary cross entropy loss measurement, and parameter p is a hyperparameter to control the compromise between minimization of the binary prediction and violation of the physical constraints. Indeed, our network is designed to track the optimal binary solution. To compensate for the error generated by the first block, which usually leads to infeasible solutions (w.r.t. the tank capacity), we introduce the second term of the loss function. This guides the predicted pump configurations towards those that privilege feasibility for the storage profile at the expense of optimality. Note that differently from the approach described in the previous chapter, here we do not track the optimal tank profile, but we only require feasibility. The shape of the tank profile is dictated by the map $\mathcal{H}_2$ built on the physical constraints.

11.2.1 A surrogate model to represent physical constraints

To explicitly incorporate the physical constraints of the optimization problem into the deep learning model, we introduce a hypothesis $\mathcal{H}_2$ to represent the underlying structure of the problem. In the fixed pump speed version of the problem, for each instance, given the configuration X, only a single corresponding pressure exists in nodes and a flow in arcs (based on the equilibrium problem). All the possible infeasibilities can be detected as violations of the tank capacities. This means that the violation of the arc capacities or pressure of other nodes only occurs if the pressure in tanks is violated (i.e., $h_{jt} \notin [\underline{H}_{jt}, \overline{H}_{jt}]$).

We define the hypothesis function $\mathcal{H}_2$, realized via an LSTM-based neural network, mapping from decision variables X and the demand profile D to its corresponding storage profile H. This is designed to mimic the extended network analysis of the problem.

[1]The Binary Cross-Entropy (also known as Binary Log Loss) is defined as follows:

$$\mathcal{L}_{CE}(X, \hat{X}) = -\frac{1}{N}\sum_{i=1}^{N} X_i \log(\hat{X}_i) + (1 - X_i)\log(1 - \hat{X}),$$

where X_i is the target binary label in a dataset of N samples and $\hat{X}_i$ is the fractional prediction [0,1] of a point belonging to the class 0 or 1.

The training set consists of input features, i.e., demand profiles and (sub-)optimal decision variables $(D, X) \in \mathbb{R}^{\mathcal{TS}} \times \{0,1\}^{\mathcal{T}\dot{\mathcal{A}}}$ and the target, i.e., the tank profile $H(D, C) \in \mathbb{R}^{\mathcal{T}\dot{\mathcal{C}}}$.

We implement two practices to obtain a model $\mathcal{H}_2$ with a good performance. The first is known as *fine tuning* [128] in the deep learning community. In this framework, the training of the complete model $\mathcal{H}$ consists of two phases. We first train the network for building $\mathcal{H}_2$ *offline*. Then, the resulting model $\mathcal{H}_2$ is integrated into the complete architecture. In the training phase of the complete network (i.e., to build $\mathcal{H}$), the weights of the model $\mathcal{H}_2$ are considered *frozen*, and only the weights of the network which models $\mathcal{H}_1$ are updated. The second idea is based on the use of a data augmentation approach. Indeed, while the training of $\mathcal{H}_2$ is implemented by using binary configuration as an input feature beside the demand profile, the output of $\mathcal{H}_1$, which is upstream in the overall architecture, is usually fractional (as it is the output of a sigmoid function). We then train the network for building $\mathcal{H}_2$ over an augmented training set, which also considers fractional input features, a method resembles *data augmentation* in deep learning community. A detailed presentation is given in the following section.

11.2.2 Data augmentation to train surrogate model

As mentioned before, the output of a neural network used for predicting binary variables, that is, model $\mathcal{H}_1$, is generally a fractional value in the interval $[0, 1]$. For this reason, we train the surrogate model $\mathcal{H}_2$ not only with binary values provided in the pre-solved instances but also with fictitious fractional configurations while the target remains the same.

We introduce minor changes (perturbations) to the input data to produce new training examples. This technique effectively increases the size and diversity of the training dataset. Due to the presence of only fixed-speed pumps and checked valves, the perturbation of the input data, i.e., pump status, may not change the inherent target values. As a result, in data augmentation, for each binary configuration in the training data, we add (subtract) a random variable from a uniform distribution in the interval of $[0, 0.5]$ to 0 (from 1) while the target value (the tank profiles) is kept the same. This technique generates 200 more data sets for each binary configuration while the targets are fixed. The resulting MAE and MSE of the model over the test data are 0.11 and 0.027, respectively. This indicates the accuracy of the surrogate model $\mathcal{H}_2$ to mimic the extended analysis problem, even for fractional status.

Chapter 12

Numerical Experiments

In this section, we evaluate the performance of the hybrid approach to find good-quality solutions. As explained, the hybrid approach considers one of the two deep learning models presented in Chapters 10 and 11 and the decomposition algorithm discussed in Chapter 9. The computational time to acquire the first feasible solution and the quality of the solution are two major indicators to judge the efficacy of the proposed algorithm.

Concerning the proposed DL architectures, we also evaluate the accuracy performance via conventional error measurements for regression and classification problems.

12.1 Numerical results of the hybrid approach: supervised deep-learning and decomposition algorithm

12.1.1 Experimental setup

To train CNN-LSTM, we have set the batch size equal to 32 and a maximum number of epochs equal to 1350. For conv1D units, we have used l_2 regularization with a coefficient equal to 0.02 and selected 32 filters. The number of hidden layers of Bi-LSTM

is set to 16. For the experimental comparison, we designed a conventional feedforward (FFW) neural network with a proportional depth and number of parameters. In particular, the FFW network consists of 8 hidden layers with $ReLU$ activation functions and pyramid architecture. We have also considered regularization with a coefficient equal to 0.01 and dropout layers with a rate between 0.2 and 0.5 after each hidden layer to mitigate overfitting and generate several starting points for the decomposition algorithm. We train FFW within a maximum number of epochs set to 250. CNN-LSTM and FFW have 45k and 63k parameters, respectively.

The Adam optimizer [129] with default initial learning rate of 10^{-3} is used to minimize the regularized Mean Square Error loss function.

Due to the complexity of the model, the relatively low number of training data, and the need for diversification, we have selected a high dropout rate (0.75) for layers of input and output of the Bi-LSTM (one layer before a fully connected layer, the outcome of the network). We took 80 Monte Carlo dropout samples and computed the component-wise mean.

Regarding the setup of the decomposition method, Algorithm 9 is run sequentially (but it can be parallelized) from at most 35 initial points H^0 until a feasible solution of $(\mathcal{P}')$ is reached. The first trial is the component-wise mean from deep learning. Other trials are randomly picked from the 80 other generated samples.

The feasibility tolerance ε_a (line 7 of Algorithm 9) is set to 10^{-6} similar to the branch-and-check algorithm. The other parameter values were chosen after a short numerical evaluation process. For the stopping criteria, we set $A = 5, B = 85$ and $\varepsilon_b = 10^{-3}$. For the penalty update policy, we increase the penalty term as a function of the iteration number, the time value, and the tank generating the constraint violation. Precisely, we set (at line 14)

$$(\rho_{ti})^{a+1} = \begin{cases} 5\xi e^{(\frac{-a}{10})}\rho_{ti}^{a} + 1 \text{ if } d_{ti}(y^{b+1}, s^{b+1}) > \varepsilon_a \\ 2\xi e^{(\frac{-a}{10})}\rho_{ti}^{a} + 1 \text{ otherwise} \end{cases} \tag{12.1}$$

for each $t \in \mathcal{T}$ and each tank $i \in \dot{\mathcal{C}}$, with ξ being randomly generated from the uniform distribution in the interval $[0.75, 1]$. The updating policy deviates from previous works in [109, 130] in which the penalty terms at each iteration are uniformly increased. Our approach is instead related to the interdependency of constraints across the periods. Indeed, this approach provides the necessary differentiated guidance to the decomposition to balance between making a constraint feasible at one period versus its cascading effects on other periods. Finally, we evaluate our hybrid algorithm under different uniform values for the penalty ρ^0 initialization (line 1). In particular, we consider $\rho^0 = 2$ and $\rho^0 = 50$.

We evaluate the performance of our hybrid algorithm (HA) in the operational context of pump scheduling on the *van Zyl* hydraulic network [44]. The hybrid algorithm is evaluated on the case $T = 12$, denoted also in this chapter as VZ12. The scaling approach discussed in Section 10.4 is instead applied to the same network but with a finer time discretization, i.e., $T = 24$ and $T = 48$. We use the notation VZ24 and VZ48, respectively.

For comparison and also to generate the dataset, we use the dedicated branch-and-check global optimizer (BC) from [6] based on the MILP solver Gurobi (v10.0.1). As this solver struggles with proving optimality or even finding a feasible solution for a significant number of the largest instances, we implemented a heavy but efficient preprocessing, including bound-tightening and cut generation techniques discussed in the previous chapters and tailored for the *van Zyl* network.

All algorithms are implemented in Python, and experiments are executed on an Intel(R) Xeon(R) 6148 2.40GHz and 128 GB memory. The DL models CNN-LSTM and FFW are built using Tensorflow API version 2.12.0 on Jupyter notebook in Google Colab with GPU A100. All the implementations can be found at `https://github.com/amirhtavakoli94/hybrid_pmpscheduling.git`

12.1.2 Performance of the supervised learning

Prediction error. To assess the effectiveness of the CNN-LSTM, we first compute the prediction error between the outcome and the target over the test set, using the classical metrics for regression problems, i.e., Mean Absolute Error (MAE)[1] and Mean Square Error (MSE)[2]. Since the range of possible values for predicted variables is different, we also report the normalized MAE (nMAE) with respect to the range of the actual value (i.e., MAE divided by the range of the actual value). Finally, we

[1]The Mean Absolute Error measures the errors between the observation Y and the output of a prediction model $\hat{Y}$. It is formally defined as follows:

$$MAE(Y, \hat{Y}) = \frac{\sum_{i=1}^{N} |Y_i - \hat{Y}|}{N}$$

[2]The Mean Square Error measures the quadratic errors between the observation Y and the output of a prediction model $\hat{Y}$. It is formally defined as follows:

$$MSE(Y, \hat{Y}) = \frac{\sum_{i=1}^{N} (Y_i - \hat{Y})^2}{N}$$

also report the Pearson coefficient (R)[3], which is a coefficient measuring the linear correlation between the ground truth and the predicted profiles.

In Figure 12.1, we illustrate the profiles predicted by CNN-LSTM architecture and the corresponding target in the test data. We also depict the credible interval, which indicates the level of uncertainty around the prediction.

The metrics reported in Table 12.1 illustrate that the CNN-LSTM architecture outperforms the FFW architecture in predicting the ground truth values in all the metrics considered in this work.

	MAE	MSE	nMAE	R
CNN-LSTM	0.53	0.64	9.2%	0.772
FFW	0.89	1.56	12.7%	0.724

Table 12.1
Comparison of the prediction accuracy of CNN-LSTM network and the FFW network over test instances VZ12.

Distance to a feasible solution. As optimal solutions of $(\mathcal{P}')$ are not unique, these metrics only show the ability of the DL model to approach one possible target. As mentioned, the predicted storage profiles do not necessarily correspond to an optimal or feasible solution of $(\mathcal{P}')$. We now measure the potentiality of the predictions to lead to feasible solutions when used as a starting point in Algorithm 9. We evaluate the impact of the initial prediction H^0 in our algorithm (with $\rho^0 = 50$) when computed with either the CNN-LSTM or the FFW architecture. When CNN-LSTM is used, the hybrid algorithm finds a feasible solution for 49 out of the 50 instances in benchmark VZ12, while only nearly half of the instances (27 out of 50) with FFW. As shown in Figure 12.2, for the 26 instances for which both the algorithms can compute 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. This suggests that a better prediction is not only more likely to end up with a feasible configuration but also the quality of the solution could be higher.

[3]The Pearson coefficient applied to N samples in a dataset consisting of the ground true Y and predicted values $\hat{Y}$ is defined as

$$R = \frac{\sum_{i=1}^{N}(Y_i - \bar{Y}_i)(\hat{Y}_i - \bar{\hat{Y}}_i)}{\sqrt{\sum_{i=1}^{N}(Y_i - \bar{Y}_i)^2}\sqrt{\sum_{i=1}^{N}(\hat{Y}_i - \bar{\hat{Y}}_i)^2}},$$

where $\bar{Y}_i$ is the sample mean of Y; and analogously for $\bar{\hat{Y}}_i$.

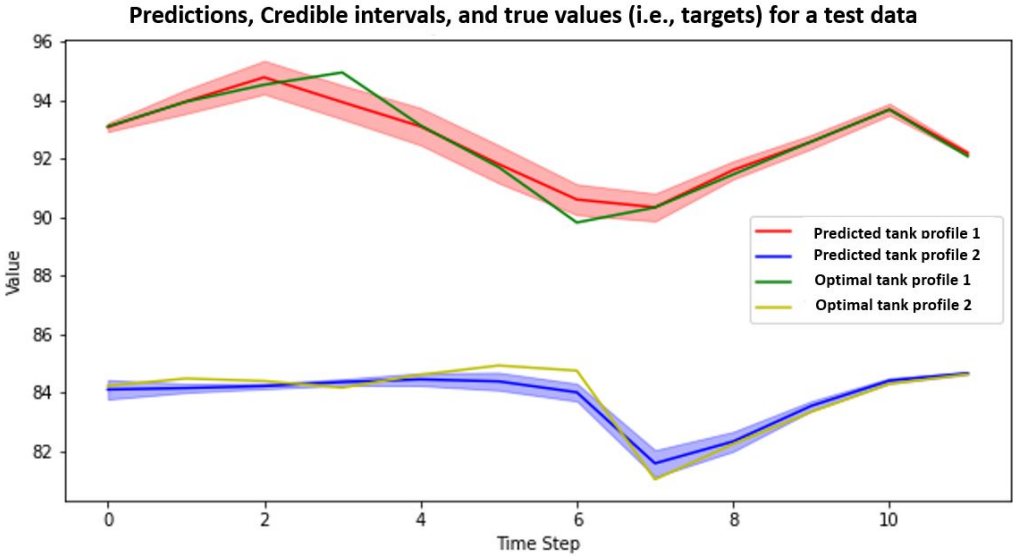

Figure 12.1: CNN-LSTM prediction and credible interval w.r.t the ground truth storage profile.

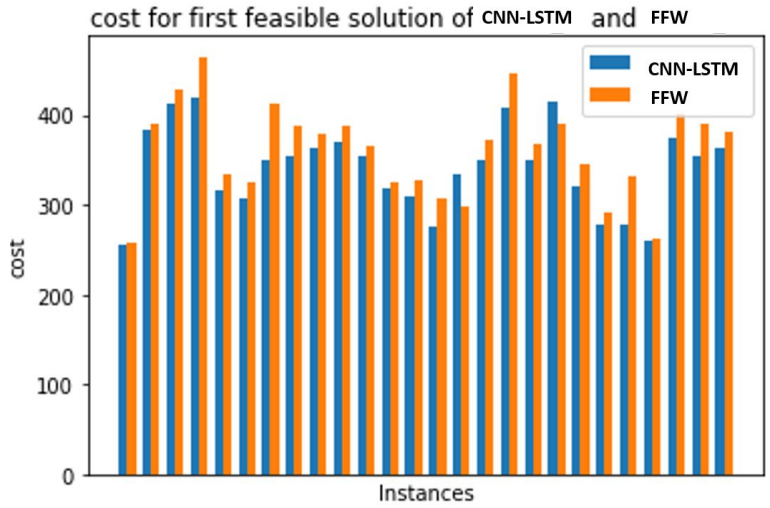

Figure 12.2: Comparison of the cost associated with the solution computed by the hybrid approach using FFW or LSTM-CNN over the test instance VZ12.

12.1.3 Performance of the hybrid algorithm

The experiments above provide a first insight into the efficiency of the proposed hybrid algorithm. They suggest that better predictions in the first phase may increase the

		#solved	Med	Mean	std	Min	Max
VZ12	HA50	49	114	254	359	6	1570
	HA2	44	114	305	438	6	1577
	BC	48	39	121	160	1	681
	BCpre	50	125	124	4	116	137
VZ24	HA50	50	183	285	281	18	1257
	HA2	50	169	279	304.	16	1711
	BC	5	425	1097	1215	272	3117
	BCpre	50	755	809	268	601	2430
VZ48	HA50	50	309	776	1294	37	7069
	HA2	49	322	1014	1435	31	5548
	BC	1	-	-	-	-	-
	BCpre	32	1892	2517	1371	1397	6404

Table 12.2
Performance: computation time in seconds.

chance of finding feasible solutions of good quality.

We now validate the capability of HA to compute good feasible solutions on the simple benchmark VZ12, for which the branch-and-check solver is usually able (with preprocessing) to even provide feasible solutions with optimality proof. We compare the hybrid algorithm with two penalty values, $\rho = 50$ (HA50) and $\rho = 2$ (HA2), with the branch-and-check with (BCpre) and without (BC) advanced preprocessing. Each algorithm stops after the first computed feasible solution and within a limit of 1800 seconds.

In Table 12.2, for each algorithm, we report statistics on the computational time (in seconds) over the instances for which the algorithm computes a feasible solution. In Table 12.3, we report statistics, on these same subsets of instances, about the solution quality: the estimated optimality gap, measured as the deviation in % of the solution cost to the best known lower bound computed with BCpre. First, note that branch-and-check always quickly reaches high-quality feasible solutions, at least with preprocessing (which requires a minimum of 100 seconds).

Without preprocessing, the results are comparable with the hybrid algorithm. The hybrid algorithm is globally outperformed, but it already performs well on this benchmark. With $\rho = 50$, it solves all except one instance in half an hour (half of them are solved in less than 2 minutes). With $\rho = 2$, the number of solved instances drops to 44, but the quality of the solutions improves significantly, as the violations weigh less in the objective of (L_ρ).

		#solved	Med	Mean	std	Min	Max
VZ12	HA50	49	6.6	6.6	4.1	0.0	21.2
	HA2	44	4.3	4.6	2.7	0.0	11.3
	BC	48	4.9	5.4	2.9	1.6	12.5
	BCpre	50	3.5	4.3	2.7	0.4	12.4
VZ24	HA50	50	9.6	9.5	4.0	3.3	23.4
	HA2	50	7.6	8.4	3.1	3.4	16.3
	BC	5	11.7	11.1	2.2	7.2	12.6
	BCpre	50	6.5	7.5	6.0	2.4	39.6
VZ48	HA50	50	8.9	9.8	3.9	3.8	21.0
	HA2	49	10.2	10.3	3.9	4.4	19.7
	BC	1	-	-	-	-	-
	BCpre	32	6.4	6.4	1.5	3.4	8.9

Table 12.3
Performance: estimated optimality gap in %.

12.1.4 Performance of the scaling approach

Tables 12.2 and 12.3 provide the same comparative results for the cases $T = 24$ (VZ24) and $T = 48$ (VZ48), where the computational time limits are increased, respectively, to 3600 and 7200 seconds to cope with the higher complexity. The scaling mechanism is now activated in the hybrid algorithm to initialize ADM. Although we do not directly predict these initialization points through a learning strategy, meaning that we do not implement a model to predict the profile for the cases $T = 24$ and $T = 48$, we still refer to the warm-started PADM as the hybrid algorithm. The reason is that we exploit the prediction results obtained with a coarser-time discretization ($T = 12$), and we resample these results (i.e., the storage level profiles) by linear interpolation.

The hybrid algorithm can now compute a feasible solution for all the instances within an average 10% estimated optimality gap. Note that it is a rough overestimation, as the lower bound is probably far below the actual optimum value for these instances. HA50 is more robust, even if the results are improved with HA2 in a few instances (particularly in VZ24). In comparison, the performance of the branch-and-check algorithm falls dramatically. Without a tailored preprocessing phase, the algorithm cannot compute even one feasible solution for most of the instances within the maximum given time. Preprocessing strongly improves the performance of this algorithm and allows computing feasible solutions for all instances for $T = 24$. The solutions are also often substantially better than with the hybrid algorithm, but the average computational time is about six times larger, see Figure 12.3. In VZ48, for

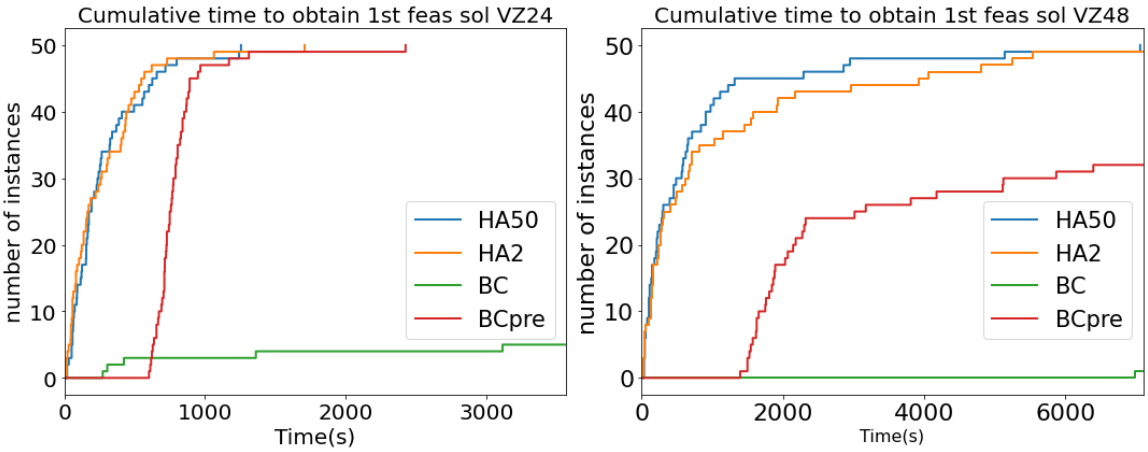

Figure 12.3: Cumulative instances to find a first feasible solution over benchmarks VZ24 (left) and VZ48 (right).

those 18 remaining instances for which we could not find a feasible solution through the branch-and-check algorithm, the median time to acquire a feasible solution with the hybrid algorithm is 500 seconds.

12.2 Numerical results of the hybrid approach: physics informed deep-learning and decomposition algorithm

12.2.1 Experimental setup

Although the physical constraints are explicitly integrated into the proposed deep learning architecture, the final output is not guaranteed to be feasible for the original optimization problem ($\mathcal{P}'$). For this reason, we use an approach similar to the previous chapter. Hence, the predicted output is used to warm-start the decomposition algorithm (Algorithm 9).

A similar setup has been used for experimental results in the decomposition algorithm. The policy update is given in (12.1), and the initial penalty for the decomposition algorithm is set to $\rho^0 = 50$.

To train the supervised penalty model $\mathcal{H}$, the maximum number of epochs is set

to 300 number. We have considered several penalty coefficients, particularly $p = 0, 0.05, 0.1$ in the loss function (11.1). Furthermore, with the penalty value, $p = 0.05$ we also consider more restricted tank capacities with the margin equal to 0.1 of the actual constraint (i.e., $p[\hat{H}(L,C) - 1.05\underline{H}]^- + p[0.95\overline{H} - \hat{H}(L,C)]^-$). We indicate this last as the robust case.

Similarly, we have used the MC dropout method to generate more than one initial start point for the decomposition algorithm. All hyperparameters are identical to the previous configurations.

All algorithms are implemented in Python, and experiments are executed on an Intel(R) Core(TM) i7-10810U 1.10GHz and 32 GB memory. The deep learning models CNN-LSTM and surrogate model are built using Tensorflow API version 2.12.0 on Jupyter notebook in Google Colab with GPU A100. All the implementations can be found at `https://github.com/amirhtavakoli94/hybrid_pmpscheduling`

12.2.2 Performance of the supervised penalty approach

Prediction error. The first indicator of the capability of the supervised penalty approach $\mathcal{H}$ is its accuracy in approximating the (sub-)optimal solutions. We have already discussed the accuracy of the surrogate model $\mathcal{H}_2$ in Section 11.2.1, so here we focus on the accuracy of the $\mathcal{H}$ for predicting binary variables. This means that we look at the output of $\mathcal{H}_1$, but we consider the learning procedure of the overall network $\mathcal{H}$. The accuracy here is measured through the binary classification error (the closer the error value is to 1, the more accurate the model is).

In Table 12.4, we illustrate the effect of the three values of the penalization parameter p considered and the robust case. The results suggest that increasing the penalty parameter in the model decreases the accuracy of the approximation of the (sub-)optimal binary solution.

12.2.3 Performance of the hybrid penalty approach

We assume that the penalty term integrated into the loss function of the DL model worsens the accuracy of the predictions of the binary variable in favor of binary configurations, leading to tank profiles within the physical limits. To prove this

classifier	Accuracy
CNN-LSTM-p0	0.84
CNN-LSTM-p0.05	0.75
CNN-LSTM-p0.1	0.74
CNN-LSTM-p0.05-rob	0.74

Table 12.4
Prediction accuracy of the physics-informed network for predicting the binary variable over the test instances VZ12.

numerically, we investigate the performance of the hybrid algorithm when warm-started with the physics-informed network. In the tables below, this version of the hybrid algorithm combined with physics-informed deep learning is denoted as PIHA. As before, the metrics to evaluate the efficacy of each configuration are the number of feasible solutions, the time required to obtain the first feasible, and their quality. The tables below show that in terms of the number of computed feasible solutions, the best performance of PIHA is with $p = 0.05$. Only the branch-and-check algorithm with extensive preprocessing (BCpre) could find all the feasible solutions. However, when it comes to the computational time, Table 12.5 shows that PIHA is, on average, ten times faster than BCpre. The reader can compare these results with Table 12.2. On the other hand, since the physics-informed DL model is not trained to predict optimal profiles, the solutions obtained by PIHA are not necessarily of good quality.

The physics-informed deep learning architecture is designed to predict feasible solutions to the optimal problem ($\mathcal{P}'$). Although this is not guaranteed, in general, the predictions are (quasi-)feasible. When they are used to initialize the PADM algorithm, this latter algorithm requires a few iterations to terminate with a feasible solution. We believe this approach could be interesting in the preprocessing phase of a global optimization algorithm, such as the branch-and-check, to have a good initialization point.

		#solved	Med	Mean	std	Min	Max
VZ12	PIHA-p0	49	59	131	152	2	510
	PIHA-p0.05	50	10	33	71	2	392
	PIHA-p0.1	49	11	45	92	2	438
	PIHA-p0.05-rob	50	11	27	39	2	169

Table 12.5
Performance: computation time in seconds.

		#solved	Med	Mean	std	Min	Max
VZ12	PIHA-p0	49	4.1	4.8	2.9	0.0	12.8
	PIHA-p0.05	50	10.5	11.1	5.6	3.2	22.3
	PIHA-p0.1	49	12.1	10.9	6.4	2.1	23.7
	PIHA-p0.05-rob	50	15.8	16.1	6.9	5.6	31.6

Table 12.6
Performance: estimated optimality gap in %.

Chapter 13

Conclusions and Prospective for Future Work

This thesis explores various ways to improve optimal operations of the drinking water network distribution while the hard constraints of the problem are still intact. We have shown that the MINLP formulation of the pump scheduling problem quickly becomes intractable. On the other hand, solutions derived from approximation or relaxation usually violate the physical constraints. From these circumstances, we have presented different preprocessing techniques to be coupled with a branch-and-check algorithm. Our results suggest a substantial improvement in the performance of the branch-and-check algorithm, both in closing the optimality gap and finding the first feasible solution after applying the proposed preprocessing. These methods are not only effective on the performance of the branch-and-check algorithm, but also efficient and relatively quick to be generated. The computational effort and its impact on the branch-and-check algorithm with some different configuration settings exhibit the fact that relaxations and definitions of each (conditional) bound tightening have been superior to any arbitrary selection. Based on previous works in flow network design and graph theory, we have also considered simplifications in the graphs by means of supernodes and cutset-based inequalities. This approach is not only useful for the cut generation, but also can be considered as new relaxations in the branch-and-check algorithm.

To our knowledge, our results are the best-reported over the real-scale *Poormond* network in the literature. However, the proposed method has some evident limitations. For instance, the formulation obtained after heavy bound tightening cannot find a feasible solution for *van Zyl* with $T = 48$. Moreover, the dedicated methods to efficiently generate cuts and bound tightening still require considerable computational

effort. We have introduced the graph partition, but except for conditional bound tightening (probing and its extended version materialized with disjunctive programming), we have not used it at the preprocessing stage in particular for steady-state relaxation. Therefore, considering this property is a straightforward way to mitigate the computational effort in the bound-tightening procedure. Apart from implications derived from the underlying structure of the problem, one might take leverage from historical data and learn the best configuration settings for bound tightening and cut generation. Generating some of the proposed cutting planes requires an intensive computational cost. In addition, the impact of some of these inequalities is very sensitive to the subset of variables we have considered or the condition that they are based on. It might be difficult to draw a clear reasoning for the generation of each subset of them, especially in light of the fact that commercial solvers may generate similar inequalities. We can indicate the recent work in optimization of the power system [131] in which authors have developed a learning algorithm to decide whether to apply heavy bound tightening for some nodes in the power network.

The other part of this work is dedicated to problem-specific heuristics to find high-quality feasible solutions. The proposed heuristic method is devised based on a variable-splitting decomposition algorithm; however, the quality of the solution is highly dependent on the initialization of the algorithm. To enhance the performance of this algorithm, we have designed deep-learning models that provide near (sub) optimal initial points. By exploiting both local patterns (via CNNs) and temporal dependencies (via Bi-LSTM) induced by storage, our deep learning model is shown to extract relevant features from the given input (tariff and demand profiles) and then exploit them to predict near-optimal solutions accurately. This combination of deep learning and decomposition methods successfully found good solutions within a short computational time. As a byproduct of developing a heuristic approach, a considerable number of instances are generated for pump scheduling problems, which may be used as a benchmark for the community. We have also investigated the quality of the initial levels of the tanks on the performance of the heuristics. However, there might be other ways to boost the performance.

Since all storage levels do not have the same importance regarding optimality or feasibility, the success of the decomposition algorithm is related not only to the initialization point but also to the initialization of the penalty parameters and their updating policy, which weighs constraint violations. These parameters can steer the decomposition trajectory. Differently from what was found in the literature, we have proposed to increase more the value of the penalties at a period where the violation occurs. In this way, we try to perturb the convergence towards an infeasible solution. A possible research direction could be investigating an intelligent (or learnable) methodology to update the penalty parameters. This is done in [98], where the dual variables (multipliers) and the coupled variables in the ADMM algorithm are

learned. However, their approach is not directly applicable to our case due to the lack of convergence of our proposed decomposition algorithm. Alternatively, the problem can be addressed by a sequential-based optimization approach such as reinforcement learning or multi-arm bandit.

The main caveat of the supervised learning framework is to collect a high number of training data for hard-to-solve instances. We addressed this difficulty with a tailored scaling method for higher-temporal resolution problems. Our numerical experiments showed a substantial improvement in the computational runtime for finding the first feasible solution for the problem and highlighted the efficiency of the approach. Another possibility might be a modification of the loss function. This can be considered an extension of the proposed supervised penalty approach (physics-informed neural network). In particular, the part of the loss function measuring the error with respect to the true solution can be replaced by the objective function of the optimization problem ($\mathcal{P}'$) itself. This new version of the loss function eliminates the need for pre-solved instances and can be helpful in cases where finding even feasible solutions is computationally demanding. This approach is used in optimal power flow and is known as the self-supervised framework [122].

One major advantage of the decomposition method to obtain good solutions is the possibility of parallelization. Each penalized problem at each period can be solved independently as the coupling constraints are obtained from the second subproblem. In this way, we can take advantage of all the processing capacity of the processors. On the contrary, the advantage of having many processing cores in branch and bound tree search fades. Thus, it would be more convenient to implement the decomposition algorithm in a high-performance machine.

Applying the decomposition based on penalizing the coupling constraints at each period is not the only way to framework the problem. While finding the optimal solution for each small subproblem is relatively easy, reaching a consensus (i.e., no violation) could require many iterations. We can shift complexity towards each subproblem and probably lift a bit of the burden by reducing the number of linking variables. It would be interesting to analyze other formulations derived by aggregating several time intervals. A similar approach can be found in [132]. The coupling temporal constraints are relaxed at some periods instead of all. Therefore, to solve each subproblem, we cannot simply enumerate the possible configuration and its penalized cost, but it might be solved with the branch-and-check algorithm. On the other hand, coupling constraints could be reduced significantly.

Future research could also be dedicated to the integration of the proposed hybrid algorithm into the branch-and-cut algorithm as a primal heuristic. Finding a good solution using a hybrid approach can be used simply in a root node as a warm start.

This may prune many branches with weaker lower bounds and facilitate the convergence. The decomposition can also be activated in some nodes in branch-and-bound search tree nodes. For instance, for some infeasible integer nodes encountered in the branch-and-bound algorithm, the decomposition algorithm can restore feasibility by flipping a few decision variables. However, the current implementation of the branch-and-check within Gurobi does not allow us to develop this method. It could be interesting to investigate an on-the-fly learning algorithm to take control of the activation of this heuristic at some infeasible node.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[2] IEA, "Water energy nexus: Excerpt from the world energy outlook," tech. rep., 2016.

[3] D. Wu, V. Nguyen, M. Minoux, and H. Tran, "An integer programming model for minimizing energy cost in water distribution system using trigger levels with additional time slots," in *RIVF International Conference on Computing and Communication Technologies*, 2021.

[4] 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, pp. 490–501, 2015.

[5] H. Donâncio, L. Vercouter, and H. Roclawski, "The pump scheduling problem: A real-world scenario for reinforcement learning," *arXiv preprint arXiv:2210.11111*, 2022.

[6] 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, pp. 1275–1313, 2021.

[7] L. H. Rossman, M. Woo, F. Tryby, R. Shang, R. Janke, and T. Haxton, *EPANET 2.2 User Manual*, 2020.

[8] J. Naoum-Sawaya, B. Ghaddar, E. Arandia, and B. Eck, "Simulation-optimization approaches for water pump scheduling and pipe replacement problems," *European Journal of Operational Research*, vol. 246, no. 1, pp. 293–306, 2015.

[9] L. H. M. Costa, B. de Athayde Prata, H. M. Ramos, and M. A. H. de Castro, "A branch-and-bound algorithm for optimal pump scheduling in water distribution networks," *Water resources management*, vol. 30, pp. 1037–1052, 2016.

[10] H. Shi and F. You, "Energy optimization of water supply system scheduling: Novel minlp model and efficient global optimization algorithm," *AIChE Journal*, vol. 62, no. 12, pp. 4277–4296, 2016.

[11] B. S. Vieira, S. F. Mayerle, L. M. Campos, and L. C. Coelho, "Optimizing drinking water distribution system operations," *Eur. J. Oper. Res.*, vol. 280, no. 3, pp. 1035–1050, 2020.

[12] B. Tasseff, *Optimization of Critical Infrastructure with Fluids*. PhD thesis, 2021.

[13] R. T. Rockafellar, *Network Flows and Monotropic Optimization*. Athena Scientific, 1999.

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

[15] J. Burgschweiger, B. Gnädig, and M. C. Steinbach, "Optimization models for operative planning in drinking water networks," *Optimization and Engineering*, vol. 10, pp. 43–73, 2009.

[16] F. Pecci, E. Abraham, and I. Stoianov, "Quadratic head loss approximations for optimisation problems in water supply networks," *Journal of Hydroinformatics*, vol. 19, no. 4, pp. 493–506, 2017.

[17] B. J. Eck and M. Mevissen, "Valve placement in water networks: Mixed-integer non-linear optimization with quadratic pipe friction," *Report No RC25307 (IRE1209-014), IBM Research*, vol. 70, 2012.

[18] A. Morsi, B. Geißler, and A. Martin, "Mixed integer optimization of water supply networks," *Mathematical optimization of water networks*, pp. 35–54, 2012.

[19] P. Belotti, P. Bonami, M. Fischetti, A. Lodi, M. Monaci, A. Nogales-Gómez, and D. Salvagnin, "On handling indicator constraints in mixed integer programming," *Comput. Optim. Appl.*, vol. 65, pp. 545–566, 2016.

[20] T. Achterberg, "Scip: solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, pp. 1–41, Jul 2009.

[21] B. Geißler, A. Morsi, L. Schewe, and M. Schmidt, "Solving power-constrained gas transportation problems using an mip-based alternating direction method," *Computers & Chemical Engineering*, vol. 82, pp. 303–317, 2015.

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

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

[24] R. O. Salgado-Castro, *Computer modelling of water supply distribution networks using the gradient method*. PhD thesis, Newcastle University, 1988.

[25] G. Mackle, "Application of genetic algorithms to pump scheduling for water supply," in *1st Int. Conf. Genet. Algorithms Eng. Syst. Innov. Appl. GALESIA*, vol. 1995, (Sheffield, UK), pp. 400–405, IEE, 1995.

[26] J. E. van Zyl, D. A. Savic, and G. A. 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.

[27] M. López-Ibáñez, T. D. Prasad, and B. Paechter, "Ant colony optimization for optimal control of pumps in water distribution networks," *J. of Water Res. Planning and Mgmt.*, vol. 134, no. 4, pp. 337–346, 2008.

[28] H. Mala-Jetmarova, N. Sultanova, and D. Savic, "Lost in optimisation of water distribution systems? A literature review of system operation," *Environ. Model. Softw.*, vol. 93, pp. 209–254, 2017.

[29] R. Z. Ríos-Mercado and C. Borraz-Sánchez, "Optimization problems in natural gas transportation systems: A state-of-the-art review," *Appl. Energy*, vol. 147, pp. 536–555, 2015.

[30] C. D'Ambrosio, A. Lodi, S. Wiese, and C. Bragalli, "Mathematical programming techniques in water network optimization," *Eur. J. Oper. Res.*, vol. 243, no. 3, pp. 774–788, 2015.

[31] 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*, vol. 185, pp. 1702–1711, 2017.

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

[33] 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.

[34] D. Fooladivanda and J. A. Taylor, "Energy-optimal pump scheduling and water flow," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1016–1026, 2017.

[35] G. Bonvin and S. Demassey, "Extended linear formulation of the pump scheduling problem in water distribution networks," in *9th Int. Network Optim. Conf.*, pp. 13–18, 2019.

[36] S. Demassey, "Enhanced branch & check for pump scheduling in water networks," in *MINLP workshop*, 2021.

[37] B. Tasseff, R. Bent, M. A. Epelman, D. Pasqualini, and P. Van Hentenryck, "Exact mixed-integer convex programming formulation for optimal water network design," *ArXiv Prepr. ArXiv201003422*, 2020.

[38] G. Bonvin, S. Demassey, and W. de Oliveira, "Robust design of pumping stations in water distribution networks," in *Optimization of Complex Systems: Theory, Models, Algorithms and Applications. Advances in Intelligent Systems and Computing (WCGO'19)*, pp. 957–967, 2019.

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

[40] C. B. Sanchez, R. Bent, S. Backhaus, S. Blumsack, H. Hijazi, and P. Van Hentenryck, "Convex optimization for joint expansion planning of natural gas and power systems," in *2016 49th Hawaii Int. Conf. Syst. Sci. HICSS*, pp. 2536–2545, IEEE, 2016.

[41] C. Borraz-Sánchez, R. Bent, S. Backhaus, H. Hijazi, and P. V. Hentenryck, "Convex relaxations for gas expansion planning," *Inf. J. Comput.*, vol. 28, no. 4, pp. 645–656, 2016.

[42] F. Wu, H. Nagarajan, A. Zlotnik, R. Sioshansi, and A. M. Rudkevich, "Adaptive convex relaxations for gas pipeline network optimization," in *2017 American Control Conference (ACC)*, pp. 4710–4716, 2017.

[43] H. Nagarajan, M. Lu, E. Yamangil, and R. Bent, "Tightening mccormick relaxations for nonlinear programs via dynamic multivariate partitioning," in *Principles and Practice of Constraint Programming*, pp. 369–387, Springer International Publishing, 2016.

[44] J. E. Van Zyl, D. A. Savic, and G. A. Walters, "Operational optimization of water distribution systems using a hybrid genetic algorithm," *Journal of water resources planning and management*, vol. 130, no. 2, pp. 160–170, 2004.

[45] R. Menke, E. Abraham, P. Parpas, and I. Stoianov, "Approximation of system components for pump scheduling optimisation," *Procedia Eng.*, vol. 119, pp. 1059–1068, 2015.

[46] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "Stl: A seasonal-trend decomposition," *J. Off. Stat*, vol. 6, no. 1, pp. 3–73, 1990.

[47] S. Burer and A. Saxena, "The milp road to miqcp," *Mixed integer nonlinear programming*, pp. 373–405, 2011.

[48] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems," *Mathematical programming*, vol. 10, no. 1, pp. 147–175, 1976.

[49] H. S. Ryoo and N. V. Sahinidis, "A branch-and-reduce approach to global optimization," *Journal of global optimization*, vol. 8, pp. 107–138, 1996.

[50] E. M. Smith and C. C. Pantelides, "Global optimisation of nonconvex minlps," *Computers & Chemical Engineering*, vol. 21, pp. S791–S796, 1997.

[51] E. S. Thorsteinsson, "Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming," in *International Conference on Principles and Practice of Constraint Programming (CP'01)*, vol. 2239 of *Lecture Notes in Computer Science*, pp. 16–30, 2001.

[52] B. Tasseff, "Optimization of Critical Infrastructure with Fluids," 2021.

[53] B. Geissler, A. Martin, A. Morsi, and L. Schewe, "Chapter 6: The MILP-relaxation approach," in *Evaluating Gas Network Capacities*, pp. 103–122, SIAM, 2015.

[54] A. M. Gleixner, T. Berthold, B. Müller, and S. Weltge, "Three enhancements for optimization-based bound tightening," *J Glob Optim*, vol. 67, no. 4, pp. 731–757, 2017.

[55] Y. Puranik and N. V. Sahinidis, "Domain reduction techniques for global NLP and MINLP optimization," *Constraints*, vol. 22, no. 3, pp. 338–376, 2017.

[56] M. W. P. Savelsbergh, "Preprocessing and probing techniques for mixed integer programming problems," *ORSA Journal on Computing*, vol. 6, no. 4, pp. 445–454, 1994.

[57] E. Balas, "Disjunctive programming: Properties of the convex hull of feasible points," *Discrete Appl. Math.*, vol. 89, no. 1-3, pp. 3–44, 1998.

[58] P. Bonami, A. Lodi, A. Tramontani, and S. Wiese, "On mathematical programming with indicator constraints," *Math. Program.*, vol. 151, no. 1, pp. 191–223, 2015.

[59] A. Vecchietti, S. Lee, and I. E. Grossmann, "Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations," *Computers & chemical engineering*, vol. 27, no. 3, pp. 433–448, 2003.

[60] B. Tasseff, R. Bent, C. Coffrin, C. Barrows, D. Sigler, J. Stickel, A. S. Zamzam, Y. Liu, and P. Van Hentenryck, "Polyhedral Relaxations for Optimal Pump Scheduling of Potable Water Distribution Networks," 2022.

[61] O. Günlük and Y. Pochet, "Mixing mixed-integer inequalities," *Math. Program.*, vol. 90, no. 3, pp. 429–457, 2001.

[62] M. Conforti, G. Cornuéjols, G. Zambelli, *et al.*, *Integer Programming*, vol. 271. Springer, 2014.

[63] H. Crowder, E. L. Johnson, and M. Padberg, "Solving large-scale zero-one linear programming problems," *Oper. Res.*, vol. 31, no. 5, pp. 803–834, 1983.

[64] Z. Gu, G. L. Nemhauser, and M. W. Savelsbergh, "Lifted cover inequalities for 0-1 integer programs: Computation," *INFORMS Journal on Computing*, vol. 10, no. 4, pp. 427–437, 1998.

[65] Q. Louveaux and L. A. Wolsey, "Lifting, superadditivity, mixed integer rounding and single node flow sets revisited," *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, pp. 173–207, 2003.

[66] A. Atamtürk, "Flow pack facets of the single node fixed-charge flow polytope," *Operations Research Letters*, vol. 29, no. 3, pp. 107–114, 2001.

[67] A. Atamtürk, "Cover and pack inequalities for (mixed) integer programming," *Ann. Oper. Res.*, vol. 139, no. 1, pp. 21–38, 2005.

[68] R. D. Carr, L. K. Fleischer, V. J. Leung, and C. A. Phillips, "Strengthening integrality gaps for capacitated network design and covering problems," tech. rep., Sandia National Lab.(SNL-NM), Albuquerque, NM (United States); Sandia, 1999.

[69] D. A. Savic and G. A. Walters, "Genetic Algorithms for Least-Cost Design of Water Distribution Networks," *J. Water Resour. Plan. Manag.*, vol. 123, no. 2, pp. 67–77, 1997.

[70] G. McCormick and R. Powell, "Derivation of near-optimal pump schedules for water distribution by simulated annealing," *Journal of the Operational Research Society*, vol. 55, no. 7, pp. 728–736, 2004.

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

[72] B. Ulanicki, J. Kahler, and H. See, "Dynamic optimization approach for solving an optimal scheduling problem in water distribution systems," *Journal of Water Resources Planning and Management*, vol. 133, no. 1, pp. 23–32, 2007.

[73] P. L. M. Bounds, K. Ulanicka, B. Ulanicki, B. Dacre, and G. Cummings, "Optimal scheduling of south-staffordshire water supply system using the finesse package," *Water supply management*, pp. 283–292, 2003.

[74] G. Hajgató, G. Paál, and B. Gyires-Tóth, "Deep reinforcement learning for real-time optimization of pumps in water distribution systems," *Journal of Water Resources Planning and Management*, vol. 146, no. 11, p. 04020079, 2020.

[75] A. M. Alvarez, Q. Louveaux, and L. Wehenkel, "A machine learning-based approximation of strong branching," *INFORMS Journal on Computing*, vol. 29, no. 1, pp. 185–195, 2017.

[76] A. Marcos Alvarez, L. Wehenkel, and Q. Louveaux, "Online learning for strong branching approximation in branch-and-bound," 2016.

[77] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[78] C. Hansknecht, I. Joormann, and S. Stiller, "Cuts, primal heuristics, and learning to branch for the time-dependent traveling salesman problem," *arXiv preprint arXiv:1805.01415*, 2018.

[79] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, and A. Lodi, "Exact combinatorial optimization with graph convolutional neural networks," *ArXiv Prepr. ArXiv190601629*, 2019.

[80] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[81] H. He, H. Daume III, and J. M. Eisner, "Learning to search in branch and bound algorithms," *Advances in neural information processing systems*, vol. 27, 2014.

[82] J. Song, R. Lanka, A. Zhao, A. Bhatnagar, Y. Yue, and M. Ono, "Learning to search via retrospective imitation," *arXiv preprint arXiv:1804.00846*, 2018.

[83] Y. Tang, S. Agrawal, and Y. Faenza, "Reinforcement Learning for Integer Programming: Learning to Cut," *ArXiv190604859 Cs Math Stat*, 2020.

[84] M. Turner, T. Koch, F. Serrano, and M. Winkler, "Adaptive cut selection in mixed-integer linear programming," *Open Journal of Mathematical Optimization*, vol. 4, pp. 1–28, 2023.

[85] R. Baltean-Lugojan, P. Bonami, R. Misener, and A. Tramontani, "Scoring positive semidefinite cutting planes for quadratic optimization via trained neural networks," pp. 1–41.

[86] Z. Huang, K. Wang, F. Liu, H.-L. Zhen, W. Zhang, M. Yuan, J. Hao, Y. Yu, and J. Wang, "Learning to select cuts for efficient mixed-integer programming," *Pattern Recognit.*, vol. 123, p. 108353, 2022.

[87] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. of Op. Res.*, vol. 290, no. 2, pp. 405–421, 2021.

[88] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!," *arXiv preprint arXiv:1803.08475*, 2018.

[89] P. Emami and S. Ranka, "Learning permutations with sinkhorn policy gradient," *arXiv preprint arXiv:1805.07010*, 2018.

[90] V. Nair, S. Bartunov, F. Gimeno, I. Von Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang, *et al.*, "Solving MIPs using neural networks," *ArXiv Prepr. ArXiv201213349*, 2020.

[91] J.-Y. Ding, C. Zhang, L. Shen, S. Li, B. Wang, Y. Xu, and L. Song, "Accelerating primal solution findings for mixed integer programs based on solution prediction," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, pp. 1452–1459, 2020.

[92] D. Yilmaz and İ. E. Büyüktahtakın, "Learning optimal solutions via an LSTM-optimization framework," in *Oper. Res. Forum*, vol. 4, p. 48, Springer, 2023.

[93] D. Masti and A. Bemporad, "Learning binary warm starts for multiparametric mixed-integer quadratic programming," in *2019 18th Eur. Control Conf. ECC*, pp. 1494–1499, IEEE, 2019.

[94] L. Anderson, M. Turner, and T. Koch, "Generative deep learning for decision making in gas networks," *Math. Methods of Operations Research*, vol. 95, no. 3, pp. 503–532, 2022.

[95] N. Guha, Z. Wang, M. Wytock, and A. Majumdar, "Machine learning for AC optimal power flow," *ArXiv Prepr. ArXiv191008842*, 2019.

[96] A. S. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," in *Int. Conf. SmartGridComm*, pp. 1–6, IEEE, 2020.

[97] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 630–637, 2020.

[98] T. W. Mak, M. Chatzos, M. Tanneau, and P. Van Hentenryck, "Learning regionally decentralized AC optimal power flows with ADMM," *IEEE Trans. on Smart Grid*, 2023.

[99] J. Douglas and H. H. Rachford, "On the numerical solution of heat conduction problems in two and three space variables," *Transactions of the American mathematical Society*, vol. 82, no. 2, pp. 421–439, 1956.

[100] J. Eckstein, "Parallel alternating direction multiplier decomposition of convex programs," *Journal of Optimization Theory and Applications*, vol. 80, no. 1, pp. 39–62, 1994.

[101] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "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.

[102] R. E. Wendell and A. P. Hurter, "Minimization of a non-separable objective function subject to disjoint constraints," *Operations Research*, vol. 24, no. 4, pp. 643–657, 1976.

[103] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: a survey and extensions," *Math. Meth. Oper. Res.*, vol. 66, pp. 373–407, 2007.

[104] P. D. Bertsekas and T. J. N., *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[105] 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.

[106] Y. Wang, W. Yin, and J. Zeng, "Global convergence of admm in nonconvex nonsmooth optimization," *J. Sci. Comput.*, vol. 78, pp. 29–63, 2019.

[107] T. Kleinert and M. Schmidt, "Computing feasible points of bilevel problems with a penalty alternating direction method," *INFORMS J. on Computing*, vol. 33, 2020.

[108] L. Schewe, M. Schmidt, and D. Weninger, "A decomposition heuristic for mixed-integer supply chain problems," *Operations Research Letters*, vol. 48, no. 3, pp. 225–232, 2020.

[109] B. Geißler, A. Morsi, L. Schewe, and M. Schmidt, "Penalty alternating direction methods for mixed-integer optimization: A new view on feasibility pumps," *SIAM J. Optim.*, vol. 27, no. 3, pp. 1611–1636, 2017.

[110] A. Tavakoli, S. Demassey, and V. Sessa, "Strengthening mathematical formulation for global optimization of the operational water network distribution," in *24ème édition du congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision - ROADEF 2023*, 2023.

[111] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[112] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[113] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[114] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[115] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[116] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[117] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv preprint arXiv:1703.04691*, 2017.

[118] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional lstm-cnns-crf," *arXiv preprint arXiv:1603.01354*, 2016.

[119] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE conf. on computer vision and pattern recognition*, pp. 1–9, 2015.

[120] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proc. of the 33rd Int. Conf. Machine Learning*, vol. 48 of *ICML'16*, pp. 1050–1059, JMLR.org, 2016.

[121] W. Chen, S. Park, M. Tanneau, and P. Van Hentenryck, "Learning optimization proxies for large-scale security-constrained economic dispatch," *Electric Power Systems Research*, vol. 213, p. 108566, 2022.

[122] S. Park and P. Van Hentenryck, "Self-supervised primal-dual learning for constrained optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pp. 4052–4060, 2023.

[123] P. L. Donti, D. Rolnick, and J. Z. Kolter, "Dc3: A learning method for optimization with hard constraints," *arXiv preprint arXiv:2104.12225*, 2021.

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

[125] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*, pp. 136–145, PMLR, 2017.

[126] R. Nellikkath and S. Chatzivasileiadis, "Physics-informed neural networks for minimising worst-case violations in dc optimal power flow," in *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 419–424, IEEE, 2021.

[127] B. Wilder, B. Dilkina, and M. Tambe, "Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1658–1665, 2019.

[128] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[129] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[130] B. Geißler, A. Morsi, L. Schewe, and M. Schmidt, "Solving highly detailed gas transport minlps: block separability and penalty alternating direction methods," *INFORMS Journal on Computing*, vol. 30, no. 2, pp. 309–323, 2018.

[131] F. Cengil, H. Nagarajan, R. Bent, S. Eksioglu, and B. Eksioglu, "Learning to accelerate globally optimal solutions to the ac optimal power flow problem," *Electric Power Systems Research*, vol. 212, p. 108275, 2022.

[132] J. Grübel, T. Kleinert, V. Krebs, G. Orlinskaya, L. Schewe, M. Schmidt, and J. Thürauf, "On electricity market equilibria with storage: Modeling, uniqueness, and a distributed admm," *Computers & Operations Research*, vol. 114, p. 104783, 2020.

Appendix A

Background on the selected deep learning architectures

This section provided technical details of the artificial neural networks used in this work. The content is designed for the reader with basic knowledge of deep learning. More details can be found in [1].

A.1 Long-Short Term Memory (LSTM)

As mentioned above, LSTM networks are an improved version of the classical recurrent neural network designed to capture not only short-time but also long-time relations in data. In LSTM architecture, the classical hidden layers of an ordinary neural network are replaced by a more sophisticated cell, which is shown in Figure A.1[1].

This cell has three inputs: the current input vector $x(t)$, the previous state of the short-term memory $c(t-1)$, and the output of the previous cell $h(t-1)$. These three values are passed through the cell, producing the new cell state $c(t)$ and hidden state $h(t)$. The system of gating controlling the flow of information is described as follows.

Input Gate. The gate regulates the flow of new information into the cell. It decides what information from the current input and the previous hidden state should

[1]From https://databasecamp.de/en/ml/lstms

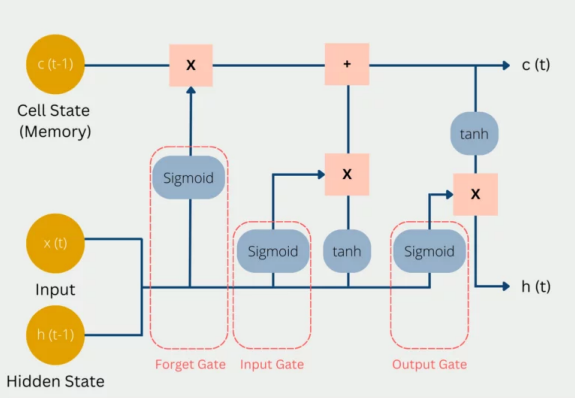

Figure A.1: An LSTM cell.

be stored in the memory cell. This is implemented by

$$i(t) = \text{sigmoid}(U^i x(t) + W^i h(t-1) + b^i),$$

where b^i, U^i, and W^i are biases, input weights, and recurrent weights defining the set of learnable parameters for the input gate.

Forget gate. It determines what information in the memory cell should be discarded or forgotten. It considers the current input and the previous hidden state to calculate a forgetting factor. This is implemented through the following function

$$f(t) = \text{sigmoid}(b^f + U^f x(t) + W^f h(t-1)),$$

where b^f, U^f, and W^f are biases, input weights, and recurrent weights define the set of learnable parameters for the forget gate. In the next step, a potential updated vector for the cell state is computed considering the current input and the previous hidden state by

$$\tilde{c(t)} = \tanh(b^c + U^c x(t) + W^c h(h-1)).$$

Finally, the cell state is updated by the following equation

$$c(t) = f(t) \odot c(t-1) + i(t) \odot \tilde{c(t)},$$

where $\odot$ denotes the element-wise product. It is $c(t)$ that enables the effective learning of long-term dependencies. This arises from the linear interactions within the remaining LSTM cell, which allow it to preserve and store information without significant alteration over extended sequences of time steps.

Output gate It decides what information to expose as the output by using the

following equation:

$$o(t) = \text{sigmoid}(b^o + U^o x(t) + W^o h(t-1)),$$

where b^o, U^o, and W^o are the learnable parameters for the output gate. The hidden state is then computed as

$$h(t) = \tanh(c(t)) \odot o(t).$$

The architecture of the LSTM network used in this thesis is bidirectional. This means that the output of the network at time t depends on the whole input sequence x. In this context, a prediction $o^{(t)}$ not only needs information from the past, that is from $t = 1$ to $t-1$, but it also looks at the future. To implement this kind of network, two kinds of LSTM networks are combined in a way that one moves forward in time and the other moves backward.

A.2 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are deep learning models specifically designed for processing and analyzing visual data. The main difference with traditional neural networks is that they use the mathematical operation called convolution instead of general matrix multiplication. So as shown in Figure A.2 the convolution involves two terms, an *input* (e.g., an image) and a so-called *kernel* (i.e., the learnable weight matrix). The output of the convolution operation is sometimes referred to as the *feature map*.

CNNs are characterized by sparse interactions, also known as sparse connectivity. This sparsity is achieved by using smaller kernels than the input data. This results in a more selective and localized pattern detection mechanism. For instance, in the case of images with a thousand pixels, the CNN can detect small, meaningful features and allows only to retain a small 'portion' of the images.

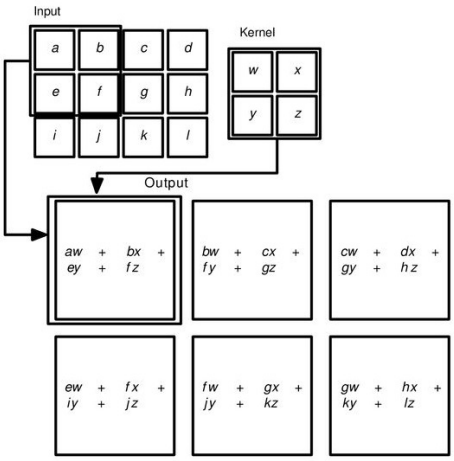

Figure A.2: An example of 2D CNN [1]

A layer of a CNN usually consists of three operations. The first performs several convolutions in parallel with different kernel sizes. Depending on the size and the number of kernels, different aspects and properties of input data can be extracted, and different feature maps can be created. The outputs are then sent to a set of nonlinear activation functions. Indeed, since convolution is a linear operation (and images are far from linear), non-linearity layers are often placed directly after the convolutional layer to introduce non-linearity to the activation map. The final operation is the application of a so-called pooling function (such as max pooling), which provides a summary statistic of the nearby outputs.

When the kernel window slides over the input data, the number of steps to perform is known as *stride*. When we use kernels smaller than the input, we can observe that the size of the corresponding output reduces. To maintain the dimension of output as in input, we use a technique called *padding*. Padding is a process of adding zeros to the input matrix symmetrically.