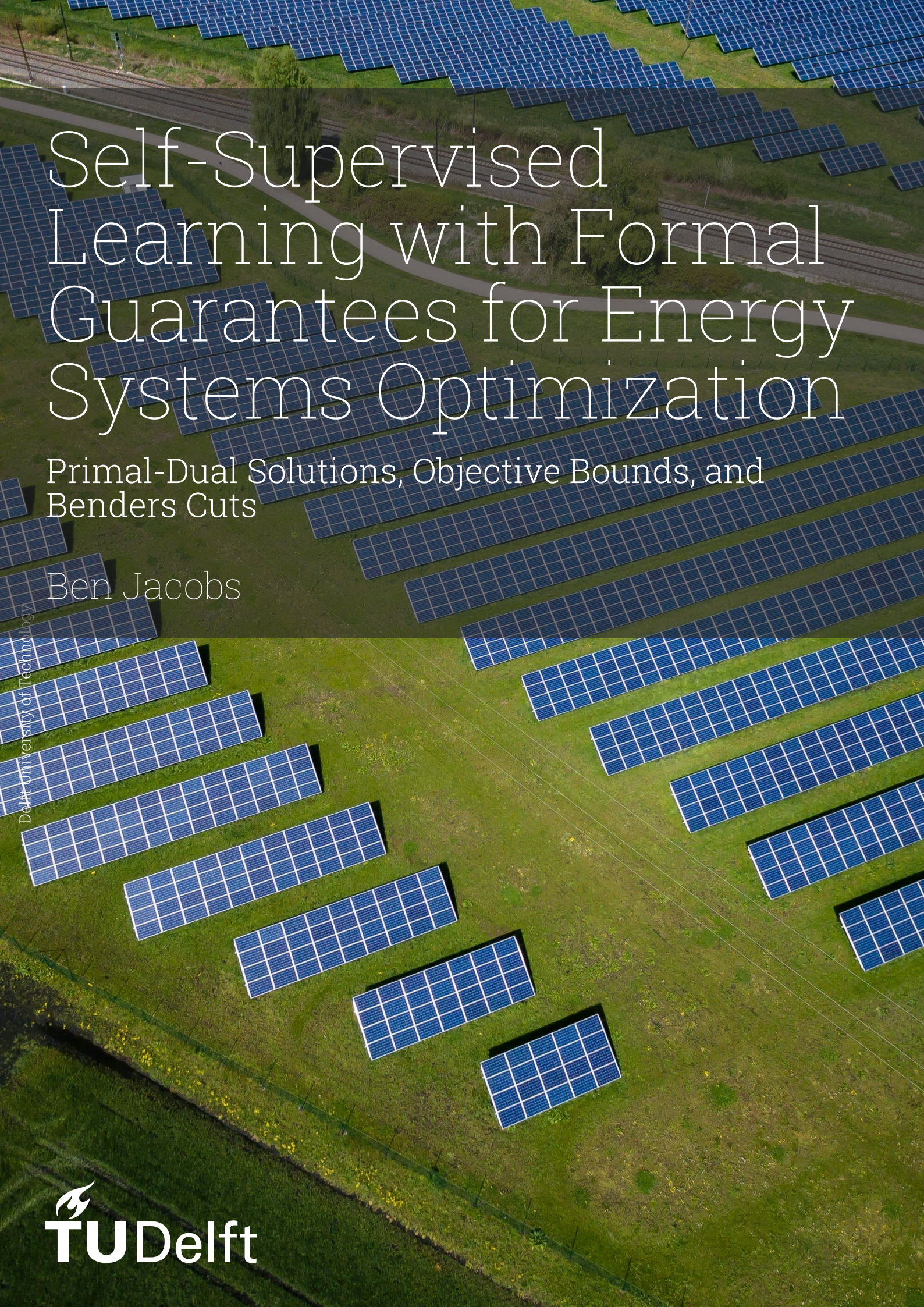

Self-Supervised Learning with Formal Guarantees for Energy Systems Optimization

Primal-Dual Solutions, Objective Bounds, and Benders Cuts

Ben Jacobs

Delft University of Technology

TU Delft

Self-Supervised Learning with Formal Guarantees for Energy Systems Optimization

Primal-Dual Solutions, Objective Bounds, and Benders Cuts

by

Ben Jacobs

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on June 23, 2025.

Student Number:	4713761	
Project Duration:	November 15, 2024 — June 23, 2025	
Supervisors:	Prof. dr. M. M. de Weerdt	TU Delft, supervisor
	Dr. G. Neustroev	TU Delft, daily supervisor
Thesis Committee:	Dr. S. H. Tindemans	TU Delft, external committee member

Cover: Photo by Jeremy Bishop on Unsplash
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This thesis marks the completion of my master's studies in Computer Science and Engineering at Delft University of Technology. I would like to express my sincere gratitude to all who have supported me throughout this process, whether through direct involvement or by providing encouragement from a distance.

First and foremost, I am deeply grateful to my supervisors at Delft University of Technology: *Dr. Greg Neustroev*, for his hands-on involvement, encouragement, and the many things I learned from him, and *Prof. Dr. Mathijs de Weerdt*, for his experienced perspective and thoughtful guidance, which provided valuable direction throughout the project.

I am also thankful to *Maaike Elgersma* for her valuable collaboration, helpful suggestions, and continuous support during the project. Furthermore, I would like to thank *Dr. Germán Morales España* for sharing his practical insights and real-world experience, which offered important perspective and helped shape the focus of this work, and *Stefan Strömer* for the insightful discussion on Benders decomposition. In addition, I wish to thank *Dr. Simon Tindemans* for his time and effort in reviewing and assessing my work.

Finally, I am grateful to my family and friends for their support and encouragement throughout my studies and beyond, especially my parents, *Marco* and *Grayson*, and in particular *Lotte*.

Ben Jacobs
Delft, June 2025

Abstract

The transition towards renewable energy requires long-term energy system planning, which depends on solving constrained optimization (CO) problems. These CO problems are becoming increasingly complex, particularly due to the variability introduced by renewable energy sources. Traditional optimization methods struggle to scale with this growing model complexity. In contrast, machine learning approaches allow faster solution computation, but offer only approximate solutions with no guarantees on solution quality, thereby limiting reliability and interpretability in planning applications.

This research addresses these limitations by exploring the use of neural networks to predict feasible solution pairs for the primal and dual formulations of the CO problem, enabling approximate solutions accompanied by bounds on suboptimality. Self-supervised primal-dual learning (PDL) is adapted and extended to produce paired feasible solutions for both the primal and dual formulations of an economic dispatch problem. Feasibility in the primal network is enforced through differentiable repair and completion layers, including novel domain-specific extensions that adjust supply-demand balancing priorities, which were found to be essential for predictive accuracy. This then exposes a structural limitation of PDL: while repair and completion layers are essential for primal learning, they prevent the learning of meaningful dual predictions. To address this, the dual network is trained using a modified loss function that directly optimizes the dual problem, enabling the use of a completion layer adopted from the literature to ensure dual feasibility. An additional novel classification-based layer that incorporates prior knowledge on the dual variables further improves the dual prediction quality when applied to the economic dispatch problem. Finally, the trained networks are integrated into Benders decomposition, a technique that breaks CO problems into easier, independent problems. This enables a hybrid approach: approximate solutions with bounds on suboptimality can be obtained swiftly, whereas exact solving remains available if necessary, retaining all theoretical convergence and optimality guarantees and potentially reducing computational time.

The proposed framework is evaluated on a generation expansion planning problem, where the economic dispatch subproblems are solved iteratively using the trained networks. The results demonstrate a theoretically grounded and empirically validated proof of concept for producing solutions with quality guarantees using learning-based methods, which is generally applicable to any problem compatible with Benders decomposition where the resulting subproblem admits a conic formulation. However, the results do not show conclusive speed-ups due to a mismatch between the training data and the data encountered during Benders iterations. By demonstrating how neural networks can be used to generate approximate, feasible solutions accompanied by theoretical guarantees on solution quality, this research contributes to the advancement of scalable and reliable constrained optimization methods for energy systems.

Contents

1

Introduction

1.1. Motivation

Climate change is widely recognized as one of the most pressing global challenges. In response, 196 countries signed the Paris Agreement in 2015, committing to limit global temperature rise to 1.5°C [24]. The International Renewable Energy Agency identifies *electrification*[1] as a key strategy for achieving this target in its pathway to 1.5°C [25]. It emphasizes that successful electrification requires forward-looking planning and the modernization and expansion of existing infrastructure. Conversely, a lack of integrated planning is noted as a barrier to effective policy and decision-making. These insights highlight the need for robust planning methods that enable policymakers to make informed, long-term decisions supporting the transition from fossil fuels to renewable energy sources.

To support effective energy system planning, various tools are employed to help decision-makers evaluate the quality of different plans. Among these, *constrained optimization* stands out as a powerful approach for mathematically modeling objectives, such as minimizing cost or maximizing value, subject to constraints like generator capacities and energy demand, thereby enabling the identification of optimal configurations. In a survey by Bhowmik et al. [8], constrained optimization techniques are applied in 17.47% of the reviewed papers on optimal planning for sustainable development, representing the largest share among all analyzed methods.

However, traditional constrained optimization techniques face computational limitations as problem complexity grows. At the same time, energy systems are becoming more complex, necessitating a trade-off between the level of detail that can be modeled and the computational effort required to solve the resulting problem [52]. This trade-off is not limited to time-critical applications, such as automatic generation control [37], where loads must be balanced in real time, but also affects planning models. In such models, constrained optimization can quickly become intractable as the level of modeling detail increases. The integration of weather-dependent renewable energy sources, in particular, introduces uncertainty and temporal variability, posing new challenges to computational tractability that cannot be resolved by advances in computing power alone [33]. These limitations highlight the urgency to improve the tractability of constrained optimization methods to enable effective and scalable energy system planning.

For this reason, improving the computational performance of constrained optimization methods is an active area of research. A variety of approaches have been developed over the years, including exact methods, heuristics, and relaxation-based approaches [39], as well as decomposition techniques [13], each aiming to improve scalability or accelerate convergence. In recent years, the integration of machine learning techniques into constrained optimization methods has gained traction due to its success in neighboring fields.

Current research at the intersection of constrained optimization and machine learning can be broadly divided into two main approaches. The first is *machine-learning-augmented optimization*, where learning

[1] Electrification refers to the widespread adoption of electricity in place of fossil fuels.

is used to support or accelerate components of traditional solving methods, while still relying on exact algorithms to ensure optimality. The second is *end-to-end constrained optimization learning*, where a machine learning model is trained to directly predict the solution to the optimization problem [32]. While the former typically preserves guarantees of optimality, the latter yields fast, approximate solutions that may deviate from the true optimum.

These approximate predictions are valuable in scenarios where solving constrained optimization problems exactly is computationally intractable. However, a key limitation remains: they do not provide bounds on suboptimality. While machine learning methods often generalize well to unseen data sampled from similar distributions, their black-box nature makes performance on structurally different or out-of-distribution instances inherently unpredictable. This uncertainty is particularly problematic in long-term energy systems planning, where solutions close to the true optimum may be acceptable, but a deviation far from the true optimum could result in poor investment decisions or infeasible plans. To ensure their trustworthiness in such applications, it is essential that these models incorporate mechanisms for interpretability and uncertainty quantification to enable effective integration into decision-making processes [17].

In *linear programming*, a constrained optimization technique applied to problems with linear components, *duality theory* provides a natural framework for estimating suboptimality: a feasible primal objective value serves as an upper bound, and a feasible dual objective value offers a lower bound on the true optimal value. Consequently, jointly predicting primal and dual solutions enables an assessment of solution quality. Moreover, many decomposition techniques, such as Benders decomposition [20], rely on both primal and dual information to guide convergence, which underscores a natural synergy between primal-dual solution pairs and decomposition techniques. This thesis extends the use of learning-based methods to predict primal-dual solution pairs and explores how such predictions can be integrated into Benders decomposition to improve scalability while retaining theoretical optimality guarantees for energy system planning.

1.2. Related Work and Research Gap

The related work covers approaches for predicting primal-dual solution pairs in constrained optimization, relevant decomposition techniques, and their integration. Section 1.2.1 first presents machine learning methods for constrained optimization, including both primal and dual methods. Section 1.2.2 then discusses decomposition techniques and highlights their potential for integration with machine learning. Next, Section 1.2.3 provides an overview of prior work that enhances Benders decomposition with machine learning techniques. Finally, Section 1.2.4 outlines the research gap addressed in this thesis.

1.2.1. Machine Learning for Energy Systems Optimization

In recent years, the success of machine learning has sparked growing interest in its application to constrained optimization problems. A review by Kotary et al. [32] highlights its effectiveness in linear programs, while also identifying the lack of formal guarantees on feasibility and performance as key open challenges, further strengthening the motivation for the research presented in this thesis.

Notable work on the intersection between learning and constrained optimization focused on making optimization problems differentiable within neural networks, enabling end-to-end learning. A prominent example is OptNet [3], which introduced quadratic programming layers as differentiable components in neural network models. This approach was later extended to a broader class of convex optimization problems, including linear programs, by Agrawal et al. [1]. These methods embed optimization problems as layers that are solved to optimality during the *forward pass*[2]. Gradients can then be computed using the implicit function theorem, allowing the model to be trained via *backpropagation*[3]. In this setting, the output of the neural network typically represents parameters of the optimization problem, rather than the solution itself. As a result, the optimization must still be solved to optimality at *inference time*[4], resulting in a significant computational cost. Since the aim of the present work is to enable faster

[2]The forward pass is the stage in which the neural network processes input data to produce an output.

[3]Backpropagation computes gradients through the network layers in reverse and uses them to update the model parameters during training.

[4]Inference time refers to the phase when the trained model is applied to new data.

inference, such approaches are not considered. Instead, the focus lies on directly predicting solutions.

Furthermore, machine learning approaches for constrained optimization can be categorized into *supervised* and *self-supervised* methods. Supervised methods require pre-solving instances to create input-output pairs, which can be time-consuming and computationally expensive [32]. These methods train models by minimizing the difference between predicted and known optimal solutions. In contrast, self-supervised methods eliminate the need for pre-computed solutions, typically by directly optimizing the objective function while penalizing constraint violations during training. Moreover, while supervised models often produce highly infeasible outputs due to a lack of constraint enforcement, self-supervised approaches can incorporate constraints directly into the learning process, improving the feasibility of predicted solutions [18]. The scope of the present research is thus limited to self-supervised approaches.

Many such self-supervised machine learning approaches involve learning to predict the primal solution to the constrained optimization problem, while enforcing feasibility through correction steps or by adding penalties in the *loss function*[5]. A promising example of primal learning is presented by Donti, Rolnick, and Kolter [18], who propose DC3, which combines *completion* and *correction* steps to enforce feasibility: the network predicts a subset of decision variables, then completes the solution to satisfy equality constraints, and finally applies gradient steps along the set of points that exactly satisfy these equality constraints to enforce the inequality constraints. Similarly, Li, Kolouri, and Mohammadi [36] propose a method that reparameterizes the optimization problem such that all predictions made by the network are mapped to feasible solutions of the original problem by design.

Additionally, self-supervised dual learning approaches have been proposed, such as the works by Klamkin, Tanneau, and Van Hentenryck [29], Tanneau and Van Hentenryck [55], and Qiu, Tanneau, and Van Hentenryck [49]. These methods ensure the feasibility of solutions to the dual problem by predicting a subset of the dual variables and completing the remainder using a closed-form expression that guarantees feasibility by construction.

Although current primal-learning and dual-learning methods show promising results, naively combining separately trained primal and dual methods disregards the intrinsic relationship between their outputs as characterized by duality theory. Coincidentally, Park and Van Hentenryck [47, 46] introduce self-supervised *primal-dual learning* (PDL), a method that alternately trains two neural networks: one to predict primal solutions and another to predict dual solutions. The training algorithm follows the trajectory of the augmented Lagrangian method [48, 22, 53]. As such, the predicted dual variables are incorporated into the loss function of the primal network, resulting in smoother convergence. On the other hand, primal outputs are used for updating targets in the dual loss function. The approach has been shown to outperform primal-only learning methods on various benchmark problems [47]. In a subsequent publication [46], the method is extended to incorporate differentiable repair and completion layers, achieving feasible primal predictions with objective values within 2% of the known optimum on a large-scale optimal power flow problem in milliseconds.

Another noteworthy approach is proposed by Kotary and Fioretto [31], where a network is trained to predict dual solutions, and primal solutions are recovered by solving a subproblem using an exact solver. While this method is shown to yield accurate solutions and incorporates duality structurally, the reliance on an exact solver makes it computationally more intensive at inference time; thus, it falls outside the scope of this thesis.

1.2.2. Decomposition Techniques for Energy System Planning Problems

Decomposition techniques are widely used to improve the computational tractability of large-scale linear programs by dividing the original problem into smaller subproblems that are solved iteratively. Common methods include Benders decomposition [20], Dantzig–Wolfe decomposition [16], and cross decomposition [58].

In Benders decomposition, *complicating variables* that couple different parts of the problem are temporarily fixed, allowing decomposition into a master problem and many independent subproblems. The solutions to these subproblems are then used to construct *cuts*, which are additional constraints

[5] A loss function quantifies the quality of predictions, which is then used to train a neural network.

that cut off a part of the search space known to be suboptimal. These cuts are added to the master problem and guide the iterative process toward optimality. This approach makes Benders decomposition particularly well-suited for energy system planning problems such as generation expansion planning, as its structure naturally separates long-term investment decisions from short-term operational ones. It has been widely recognized in the literature for this purpose, with early work highlighting its effectiveness specifically in energy system planning [14], and a more recent review emphasizing its general applicability to planning problems [50].

Dantzig–Wolfe decomposition [16] is effective when *complicating constraints* link otherwise independent subproblems. These constraints are enforced in the master problem, while the subproblems are solved independently. Unlike Benders decomposition, which adds constraints (rows) to the master problem, Dantzig–Wolfe adds variables (columns). Accordingly, Benders is often referred to as row generation, and Dantzig–Wolfe as column generation. Dantzig–Wolfe is well suited to scheduling and routing problems, where local decisions can be optimized independently and coordinated globally.

Cross decomposition [58] combines the principles of Benders and Dantzig–Wolfe decomposition. Dantzig–Wolfe decomposition may be applied to the master problem while Benders decomposition is used to solve the resulting subproblems, or vice versa, depending on the structure of the problem.

The repetitive nature of decomposition techniques, where similar subproblems are solved repeatedly, makes them suitable for integration with machine learning, which is known for its ability to generalize across instances from the same distribution. Because solving the subproblems often takes significantly more time than solving the master problem, the total computational time can be greatly improved by reducing the computational time required to solve the subproblems. An example is given by Morabit, Desaulniers, and Lodi [44], who demonstrate a 30% computational gain by leveraging machine learning to select columns during column generation, a method based on Dantzig–Wolfe decomposition. A machine learning model that can accurately predict the solutions to the subproblems directly, removing the need for a conventional solver for the subproblems, could lead to more significant computational gains.

The research presented in this thesis focuses on Benders decomposition due to its particular applicability to energy systems planning problems.

1.2.3. Machine-Learning-Enhanced Benders Decomposition

In recent years, several works have combined machine learning with Benders decomposition across various domains beyond the scope of this thesis. Many of these focus on accelerating convergence while preserving optimality, typically by selecting valuable cuts or enhancing their effectiveness through learning techniques [59, 43, 9, 34, 27, 41]. These strategies remain compatible with predicting primal-dual solution pairs to the subproblems of Benders decomposition.

Other approaches reduce computational cost using subproblem solution approximations [40, 42]. Mitrai and Daoutidis [40] employ surrogate models to estimate subproblem objective values and dual variables, generating Benders cuts without explicitly solving the subproblems. Although feasibility is enforced through the structure of the master problem, the surrogate models do not guarantee feasible solutions to the subproblems. Thus, no bounds on suboptimality can be given for the resulting Benders solution.

A follow-up study by Mitrai and Daoutidis [42] builds on this idea by using the surrogate-model predictions to identify promising instances of the complicating variables. For these points, exact subproblems are solved to compute valid Benders cuts, which are then used to warm start the exact decomposition procedure.

1.2.4. The Research Gap

The joint training of the primal and dual networks of PDL makes it a compelling candidate for predicting primal-dual solution pairs. However, research into the method remains limited, with current findings, to the best of our knowledge, only reported by Park and Van Hentenryck [47, 46]. New challenges will likely arise when applying PDL to different problem classes, such as generation expansion planning. More importantly, the results reported in earlier work focused on the quality of primal solutions, while the quality of the dual solutions has been largely overlooked. Since dual solutions are

essential to provide bounds on suboptimality, this limitation must be addressed to accurately assess the applicability of PDL in this context.

Furthermore, although the combination of Benders decomposition and machine learning has been examined in prior research, limited attention has been given to the integration of predicted primal-dual solution pairs within the iterative solution of Benders subproblems. Existing approaches that approximate Benders cuts are unable to provide valid bounds on suboptimality, thereby limiting their applicability in contexts where solution quality guarantees are required. Moreover, these methods rely on the assumption that feasibility can be enforced through the structure of the master problem. This assumption generally does not hold in energy systems planning, where the subproblems typically determine feasibility.

The research gap is thus twofold:

1. Research on PDL is limited and overlooks dual solution quality, which is crucial for valid bounds on suboptimality.
2. The integration of approximate primal-dual solution pairs into Benders decomposition remains largely unexplored, particularly regarding their impact on feasibility, optimality bounds, and convergence guarantees in energy systems planning contexts.

Addressing this gap will clarify whether PDL can be used to predict high-quality primal-dual pairs that yield valid optimality bounds, and how such approximate solutions can be effectively integrated into Benders decomposition to accelerate energy systems planning without sacrificing theoretical guarantees.

Non-convex settings pose additional theoretical and computational challenges, particularly due to the lack of strong duality and the potential absence of well-defined dual solutions. These issues make it difficult to derive meaningful optimality bounds or guarantees. The scope of this thesis is therefore limited to linear programs and mixed-integer linear programs through the use of Benders decomposition, where dual solutions are well-defined and can be used to provide provable bounds on optimality. This limitation does not hinder applicability to many energy systems planning problems, however, as these are often formulated as linear operational problems with integer investment decisions [30].

1.3. Research Questions

The following research question is formulated to address the research gap identified in the previous section:

> *Can self-supervised primal-dual learning efficiently generate high-quality primal and dual solutions with verifiable optimality guarantees, and can these be integrated into Benders decomposition to accelerate energy systems planning?*

To guide the research throughout this thesis, the main research question is broken down into the following subquestions:

1. How does the primal solution quality of PDL vary with problem structure?
2. Can the quality of dual predictions from PDL support meaningful and reliable bounds on suboptimality?
3. What extensions to PDL improve the feasibility and optimality of primal and dual solutions?
4. Can PDL be effectively integrated with Benders decomposition, and under what conditions can theoretical performance guarantees be preserved?

1.4. Contributions

The contributions made in this thesis are summarized below.

- The quality of primal solutions produced by PDL, with differentiable extensions that ensure feasibility, is evaluated in a new *economic dispatch*[6] application domain along with novel problem-

[6] Economic dispatch is a fundamental subproblem embedded within generation expansion planning.

specific enhancements.

- Inherent limitations in the dual prediction quality of PDL are identified and addressed for the economic dispatch problem by employing an alternative dual loss function and differentiable feasibility-enforcing extensions, including a novel classification-based approach that incorporates prior information on dual variables.
- A learning-enhanced Benders decomposition method, leveraging feasible primal and dual solutions and supported by formal theoretical guarantees, is proposed and empirically evaluated on a generation expansion planning problem.

1.5. Outline

The remainder of this thesis is structured as follows.

Chapter 2 introduces the necessary theoretical background, including concepts relevant to deep learning, constrained optimization, PDL, and Benders decomposition. Next, Chapter 3 introduces the formulations of the convex quadratic programming benchmark, economic dispatch, and generation expansion planning problems considered throughout this thesis. Additionally, the chapter formalizes and motivates the metrics used to measure solution quality.

The following chapters present results that align with answering the subquestions. Subquestion 1 is addressed in Chapter 4 by evaluating the performance of PDL under structural variations introduced into a convex quadratic programming benchmark, extending the evaluation beyond settings considered in previous work. Next, Chapter 5 continues addressing Subquestion 1 by evaluating PDL on an economic dispatch problem, a previously unexplored domain. In addition, Subquestion 3 is addressed by incorporating additional differentiable layers in the primal network to improve feasibility and optimality. Then, Chapter 6 turns to Subquestion 2 by evaluating the quality of dual solutions learned through PDL on both the convex quadratic programming benchmark and the economic dispatch problem. It also contributes to Subquestion 3 by introducing and testing additional differentiable extensions to the dual network. Following this, Chapter 7 addresses Subquestion 4 by integrating predicted primal-dual solution pairs into Benders decomposition and establishing theoretical conditions under which predicted solutions preserve convergence guarantees. The chapter further presents experimental results on a generation expansion planning problem, comparing inexact and hybrid Benders variants in terms of convergence, solution quality, and computational efficiency.

Finally, Chapter 8 concludes the thesis by answering the research question and relating the findings to a broader context. It discusses the limitations of the proposed methods and outlines avenues for future research.

2

Theoretical Background

This chapter provides the theoretical foundation for the remainder of this thesis. The concepts introduced here form the basis for the discussions and analyses in subsequent chapters and are essential for understanding the methods explored in this work. Since this thesis investigates ideas at the intersection of deep learning and constrained optimization, it introduces the necessary background for both domains.

Section 2.1 introduces the relevant background and terminology of deep learning. Section 2.2 presents the fundamentals of constrained optimization, focusing on linear programming, which forms the core of this thesis. Since solving constrained problems directly can be challenging, Section 2.3 presents the Lagrangian formulation, which enables the application of unconstrained optimization techniques. Building on this, Section 2.4 introduces the dual problem formulation, which provides an alternative perspective on the optimization problem and is fundamental for many solving methods. In Section 2.5, the augmented Lagrangian method is discussed, which improves upon the standard Lagrangian approach by addressing some of its limitations in constrained optimization. This method forms the basis of PDL, a method proposed by Park and Van Hentenryck [47, 46], which is the primarily researched method in this thesis. This method is formally introduced in Section 2.6. Finally, Section 2.7 outlines the basics of Benders decomposition. The integration of PDL with Benders decomposition is a key focus of this thesis.

For readers interested in a broader understanding of optimization theory, the book by Michel [39] provides a comprehensive overview, forming the basis of many explanations in this chapter. For decomposition techniques, the reader is referred to the book by Conejo et al. [13].

2.1. Deep Learning

Machine learning models aim to approximate unknown functions that map inputs to outputs without being explicitly programmed. These models learn by identifying patterns and relationships within data. In classical machine learning, the mapping from inputs to outputs is often assumed to be relatively simple or linear, as in models like linear regression or decision trees. However, many real-world tasks involve complex, highly nonlinear relationships that such models cannot capture effectively. Deep learning, a subfield of machine learning, addresses this limitation by using neural networks composed of multiple layers of computation. These additional layers, hence the term "deep", enable the modeling of highly flexible, nonlinear functions.

2.1.1. Multi-Layer Perceptron

Various neural network architectures exist, each tailored to different types of data and tasks. One of the most fundamental and widely used architectures is the *multi-layer perceptron*, also referred to as a fully connected feedforward network. A multi-layer perceptron is composed of layers of so-called "neurons", where each neuron applies a linear transformation followed by a nonlinear activation function:

$$\mathbf{x}^{(l)} = \sigma(\mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}), \tag{2.1}$$

where $\mathbf{x}^{(l)}$ is the output vector of layer l, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the learnable weight matrix and bias vector, respectively, and are referred to as the *learnable parameters*. Furthermore, σ is a nonlinear activation function. The input to the first layer is the problem feature vector, $\mathbf{x}^{(0)} = \mathbf{z}$, and the output of the final layer is the prediction $\hat{\mathbf{x}}$.

Activation functions such as *ReLU*[1] or *sigmoid*[2] are essential because they introduce nonlinearity into the model. Without them, stacking multiple linear layers would be equivalent to a single linear transformation, which limits expressiveness. Nonlinearities enable the network to model complex patterns, which cannot be solved using a purely linear model.

2.1.2. Forward Pass and Loss Functions

A *forward pass* refers to the process of propagating an input through the network to produce an output. Given an input $\mathbf{z}$, the network output $\hat{\mathbf{x}}$ is computed by sequentially applying the layer operations, as is done in (2.1). Once an output is obtained, its quality is assessed using a *loss function*. In supervised methods, the loss compares the output to a known target. In contrast, self-supervised methods directly model the quality of a given solution without requiring a target, for example, by computing its objective value from a known objective function.

2.1.3. Backpropagation

Neural networks are trained by adjusting their learnable parameters to minimize the loss function. This is achieved through *backpropagation*, which computes the gradient of the loss with respect to each learnable parameter using the chain rule of calculus. The gradients are propagated backward, from the loss function to the first layer, which is why the method is called backpropagation.

These gradients are then used by an *optimization algorithm* to update the learnable parameters. The optimizer takes steps in the negative direction of the gradient to reduce the loss. The size of these steps is controlled by the *learning rate*. Parameters that guide the training process, such as the learning rate, are referred to as *hyperparameters*. These hyperparameters are not learned during training but are instead manually tuned to ensure good performance.

Each computation from input $\mathbf{z}$ to output $\hat{\mathbf{x}}$ must be *differentiable* to allow gradients to flow through all components of the model during training.

2.1.4. Training, Validation and Testing

To evaluate the performance of a deep learning model, the available data is typically divided into three disjoint subsets: a *training set*, a *validation set*, and a *test set*. The training set is used to iteratively update the model's parameters through backpropagation. A complete pass through the training data is referred to as an *epoch*. The validation set is used during training to tune hyperparameters and monitor performance in order to prevent *overfitting*, a phenomenon where the model captures noise or specific patterns in the training data, thereby failing to generalize to new data. The test set is held out entirely during training and validation, and is only used after training is complete to provide an unbiased assessment of the model's performance on unseen data.

2.2. Constrained Optimization

Constrained optimization is the process of finding the optimal solution to a mathematical problem while satisfying the given constraints. The optimal solution is one that either minimizes or maximizes the objective function, for example, minimizing the cost or maximizing the profit. A constrained optimization problem is generally written as:

[1] ReLU (Rectified Linear Unit) is a simple activation function that returns the input if it is positive and zero otherwise

[2] Sigmoid is a smooth, S-shaped activation function that squashes input values into the range between 0 and 1.

$$\underset{\mathbf{x}}{\text{minimize}} \quad f(\mathbf{x}) \tag{2.2a}$$

$$\text{subject to} \quad g_i(\mathbf{x}) \leq 0; \qquad i = 1, \dots, M, \tag{2.2b}$$

$$h_j(\mathbf{x}) = 0; \qquad j = 1, \dots, O. \tag{2.2c}$$

Here, $f : \mathbb{R}^Q \to \mathbb{R}$ is the objective function that operates on the vector of decision variables $\mathbf{x} \in \mathbb{R}^Q$, whose elements can be either continuous, integer, binary, or a combination of these. The expressions $g_i(\mathbf{x}) \leq 0$ and $h_j(\mathbf{x}) = 0$ define the inequality and equality constraints, respectively, which must be satisfied. Here, $g : \mathbb{R}^Q \to \mathbb{R}^M$ and $h : \mathbb{R}^Q \to \mathbb{R}^O$ are vector-valued functions collecting multiple scalar constraints. A solution that adheres to all constraints is *feasible*, whereas one that violates at least one constraint is *infeasible*. An inequality constraint is referred to as *active* or *tight* if it is satisfied exactly at the boundary. A maximization problem can be transformed into a minimization problem by minimizing $-f(\mathbf{x})$. A solution $\mathbf{x}^\star$ is *optimal* if it minimizes the objective function, that is,

$$f(\mathbf{x}^\star) \leq f(\mathbf{x}), \quad \forall\, \mathbf{x} \neq \mathbf{x}^\star. \tag{2.3}$$

This representation of the problem is known as the *primal* problem.

2.2.1. Linear Programming

If the objective function and all constraints are linear, a constrained optimization problem becomes a linear programming problem (LP) [15]. Linear programs in which at least one decision variable is restricted to being an integer are called mixed-integer linear programming.

Due to the linear nature of this problem, the objective function $f(\mathbf{x})$ can be expressed as the dot product of a cost vector $\mathbf{c}$ and the decision variables $\mathbf{x}$. Similarly, the inequality and equality constraint functions can be represented using constraint matrices $\mathbf{G}$ and $\mathbf{H}$, which, after matrix multiplication with $\mathbf{x}$, are compared element-wise to the right-hand-side vectors $\mathbf{b}$ and $\mathbf{d}$. The primal LP problem can thus be written as:

$$\underset{\mathbf{x}}{\text{minimize}} \quad \mathbf{c}^\top \mathbf{x} \tag{2.4a}$$

$$\text{subject to} \quad \mathbf{G}\mathbf{x} \leq \mathbf{b}, \tag{2.4b}$$

$$\mathbf{H}\mathbf{x} = \mathbf{d}. \tag{2.4c}$$

This linear formulation will be used throughout this chapter, although the discussed methods are not necessarily limited to LP. However, in the context of this thesis, they are applied explicitly to LP.

Constrained optimization methods are generally challenging because they must ensure feasibility while optimizing, which introduces complexity in both mathematical formulation and computational implementation. Several methods exist for solving LP problems, with the simplex method [15] being one of the most widely used. It iteratively traverses the extreme points of the feasible region defined by the constraints to identify the optimal solution. Other well-known methods include the dual simplex method [35], which starts with an optimal but infeasible solution and iteratively restores feasibility while maintaining optimality, as well as interior-point methods such as the barrier method [19], which approach the solution from within the feasible region rather than along its boundary.

2.3. Lagrangian Formulation and Optimality Conditions

An LP problem can be reformulated as an unconstrained optimization problem by incorporating the constraints into the objective function through Lagrange multipliers. The resulting function, known as the *Lagrangian*, is defined as:

$$\mathscr{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) \coloneqq \mathbf{c}^\top \mathbf{x} + \boldsymbol{\mu}^\top (\mathbf{G}\mathbf{x} - \mathbf{b}) + \boldsymbol{\lambda}^\top (\mathbf{H}\mathbf{x} - \mathbf{d}), \tag{2.5}$$

where $\boldsymbol{\mu}$ is the vector of Lagrange multipliers for the inequality constraints, and $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers for the equality constraints. These multipliers are also known as the dual variables.

This formulation allows us to handle constraints implicitly through the Lagrangian multipliers, enabling the use of potentially less computationally intensive unconstrained optimization methods.

However, minimizing the Lagrangian function alone does not necessarily lead to the optimal decision variables $\mathbf{x}^\star$. For the optimal solution of the Lagrangian formulation to coincide with the solution of the original LP problem, certain conditions must hold. These are known as the *Karush–Kuhn–Tucker* (*KKT*) *conditions*, which provide necessary and sufficient conditions for optimality in constrained optimization:

1. **Stationarity**: The gradient of the Lagrangian must vanish with respect to the decision variables:

$$\nabla_{\mathbf{x}} \mathscr{L}(\mathbf{x}^\star, \boldsymbol{\mu}, \boldsymbol{\lambda}) = 0. \tag{2.6}$$

   This ensures no further improvement in the objective function is possible at $\mathbf{x}^\star$ while respecting the constraints. It should be noted that the gradient with respect to $\mathbf{x}$ is a constant in LP, simplifying the stationarity condition.

2. **Primal feasibility**: The solution must satisfy the original problem constraints:

$$\mathbf{G}\mathbf{x}^\star \leq \mathbf{b}, \quad \mathbf{H}\mathbf{x}^\star = \mathbf{d}. \tag{2.7}$$

   This guarantees that $\mathbf{x}^\star$ is a valid solution within the feasible region.

3. **Dual feasibility**: The Lagrange multipliers associated with the inequality constraints must be non-negative:

$$\boldsymbol{\mu} \geq 0. \tag{2.8}$$

   This condition ensures the Lagrangian remains bounded below, as negative $\boldsymbol{\mu}$ would allow it to decrease without limit by over-satisfying the inequality constraints.

4. **Complementary slackness**: If a constraint is not tight, its corresponding Lagrange multiplier must be zero:

$$\boldsymbol{\mu}^\top(\mathbf{G}\mathbf{x}^\star - \mathbf{b}) = 0. \tag{2.9}$$

   This means that an inequality constraint at the optimal solution is either active ($\mathbf{G}\mathbf{x}^\star = \mathbf{b}$, allowing $\boldsymbol{\mu}$ to be positive) or inactive ($\mathbf{G}\mathbf{x}^\star < \mathbf{b}$, forcing $\boldsymbol{\mu} = 0$). Because $\mathbf{H}\mathbf{x}^\star - \mathbf{d} = 0$, $\boldsymbol{\lambda}$ is allowed to take on all values. This condition ensures that the dual variables only influence the optimization when necessary, that is, when the solution is infeasible.

When these conditions are satisfied, the solution $\mathbf{x}^\star$ is optimal for both the original LP problem and the Lagrangian formulation. In practice, when optimizing the Lagrangian formulation, the Lagrange multipliers $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ must be carefully chosen to ensure that the KKT conditions hold.

2.4. Dual Problem

The Lagrangian function can be rewritten in the following form:

$$\mathscr{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \mathbf{x}^\top(\mathbf{c} + \mathbf{G}^\top\boldsymbol{\mu} + \mathbf{H}^\top\boldsymbol{\lambda}) - \mathbf{b}^\top\boldsymbol{\mu} - \mathbf{d}^\top\boldsymbol{\lambda}. \tag{2.10}$$

A function f_D parameterized by the dual variables $(\boldsymbol{\mu}, \boldsymbol{\lambda})$, can be derived by minimizing the Lagrangian over $\mathbf{x}$:

$$f_D(\boldsymbol{\mu}, \boldsymbol{\lambda}) \coloneqq \inf_{\mathbf{x}} \mathscr{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}). \tag{2.11}$$

However, for the Lagrangian to be bounded below with respect to $\mathbf{x}$, the coefficient of $\mathbf{x}$ in the Lagrangian must vanish. This is the case when:

$$\mathbf{c} + \mathbf{G}^\top\boldsymbol{\mu} + \mathbf{H}^\top\boldsymbol{\lambda} = 0. \tag{2.12}$$

When adding this as an equality constraint on the function f_D, the condition can be substituted back into the Lagrangian (2.5), removing the dependence on $\mathbf{x}$ and resulting in the following definition for f_D:

$$f_D(\boldsymbol{\mu}, \boldsymbol{\lambda}) = -\mathbf{b}^\top \boldsymbol{\mu} - \mathbf{d}^\top \boldsymbol{\lambda}. \tag{2.13}$$

Additionally, for any feasible primal solution $\mathbf{x}$, we have $\mathbf{G}\mathbf{x} \leq \mathbf{b}$ and $\mathbf{H}\mathbf{x} - \mathbf{d} = 0$. Since $\boldsymbol{\mu} \geq 0$ by (2.8), it follows that $\boldsymbol{\mu}^\top(\mathbf{G}\mathbf{x} - \mathbf{b}) \leq 0$. When substituting this into the Lagrangian (2.5), this yields:

$$\mathscr{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) \leq \mathbf{c}^\top \mathbf{x}. \tag{2.14}$$

Since $-\mathbf{b}^\top \boldsymbol{\mu} - \mathbf{d}^\top \boldsymbol{\lambda} = \inf_{\mathbf{x}} \mathscr{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda})$, we obtain:

$$-\mathbf{b}^\top \boldsymbol{\mu} - \mathbf{d}^\top \boldsymbol{\lambda} \leq \mathbf{c}^\top \mathbf{x}. \tag{2.15}$$

Thus, a feasible solution to $f_D(\boldsymbol{\mu}, \boldsymbol{\lambda})$ provides a lower bound on the primal problem's optimal value. Our goal is to find the best possible lower bound, which is achieved by maximizing $f_D(\boldsymbol{\mu}, \boldsymbol{\lambda})$. The dual objective function f_D is unbounded for negative values of $\boldsymbol{\mu}$ following (2.8), which is why a nonnegativity constraint is added for $\boldsymbol{\mu}$. The resulting optimization problem is thus formulated as:

$$\begin{aligned}
\underset{\boldsymbol{\mu}, \boldsymbol{\lambda}}{\text{maximize}} \quad & -\mathbf{b}^\top \boldsymbol{\mu} - \mathbf{d}^\top \boldsymbol{\lambda} && \text{(2.16a)} \\
\text{subject to} \quad & \mathbf{G}^\top \boldsymbol{\mu} + \mathbf{H}^\top \boldsymbol{\lambda} = -\mathbf{c}, && \text{(2.16b)} \\
& \boldsymbol{\mu} \geq 0 && \text{(2.16c)}
\end{aligned}$$

This optimization problem is known as the *dual* problem formulation. From (2.15), the following theorem follows directly:

Theorem 1 (Weak Duality)**.** *Let* $\mathbf{x}$ *be any feasible solution to the primal problem, and* $(\boldsymbol{\mu}, \boldsymbol{\lambda})$ *be any feasible solution to the dual problem. Then the value of the dual objective provides a lower bound to the value of the primal objective:*

$$-\mathbf{b}^\top \boldsymbol{\mu} - \mathbf{d}^\top \boldsymbol{\lambda} \leq \mathbf{c}^\top \mathbf{x}. \tag{2.17}$$

In linear programs, this bound is tight at optimality, resulting in strong duality:

Theorem 2 (Strong Duality)**.** *If the primal linear program has an optimal solution* $\mathbf{x}^\star$*, then the dual also has an optimal solution* $(\boldsymbol{\mu}^\star, \boldsymbol{\lambda}^\star)$*, and the optimal objective values are equal:*

$$\mathbf{c}^\top \mathbf{x}^\star = -\mathbf{b}^\top \boldsymbol{\mu}^\star - \mathbf{d}^\top \boldsymbol{\lambda}^\star. \tag{2.18}$$

Combining both theorems gives:

$$\mathbf{b}^\top \boldsymbol{\mu} - \mathbf{d}^\top \boldsymbol{\lambda} \leq \mathbf{b}^\top \boldsymbol{\mu}^\star - \mathbf{d}^\top \boldsymbol{\lambda}^\star = \mathbf{c}^\top \mathbf{x}^\star \leq \mathbf{c}^\top \mathbf{x}. \tag{2.19}$$

This highlights several important properties of duality. First, optimal dual variables quantify the sensitivity of the optimal primal objective to changes in the constraint right-hand sides, as seen directly from the dual objective function. Second, any pair of feasible primal and dual solutions provides bounds around the optimal value. The distance between the primal and dual solutions thus provides a sense of the optimality of both solutions if strong duality holds.

In some cases, the dual formulation may offer computational advantages, especially when it has fewer variables or a simpler structure than the primal.

2.5. The Augmented Lagrangian Method

Understanding PDL first requires an understanding of the augmented Lagrangian method (ALM) [22, 48, 53], an iterative approach to solving constrained optimization problems. It transforms the problem into a sequence of unconstrained problems by modifying the Lagrangian function and updating the Lagrange multipliers at each iteration.

Since the standard Lagrangian does not inherently enforce feasibility, ALM incorporates a penalty term, augmenting the Lagrangian, to penalize constraint violations. The penalty weight increases over iterations, gradually ensuring feasibility while optimizing the objective function.

The penalty term takes on the following form:

$$\frac{\rho}{2}(\|\max(\mathbf{Gx}-\mathbf{b},\mathbf{0})\|_2^2+\|\mathbf{Hx}-\mathbf{d}\|_2^2), \tag{2.20}$$

where ρ is the scalar weighing of the penalty term. Constraint violations are penalized quadratically, ensuring that both negative and positive deviations from the equality constraints $\mathbf{Hx}-\mathbf{d}=0$ are penalized. The element-wise max operator ensures that satisfied inequality constraints $\mathbf{Gx}-\mathbf{b}\leq 0$ are not penalized.

The full augmented Lagrangian is then given by:

$$\mathscr{L}_\rho(\mathbf{x},\boldsymbol{\mu},\boldsymbol{\lambda})=\mathbf{c}^\top\mathbf{x}+\boldsymbol{\mu}^\top(\mathbf{Gx}-\mathbf{b})+\boldsymbol{\lambda}^\top(\mathbf{Hx}-\mathbf{d})+\frac{\rho}{2}(\|\max(\mathbf{Gx}-\mathbf{b},\mathbf{0})\|_2^2+\|\mathbf{Hx}-\mathbf{d}\|_2^2). \tag{2.21}$$

At each iteration k, the unconstrained optimizer finds a solution $\hat{\mathbf{x}}_k$ given the Lagrange multiplier estimates $\boldsymbol{\mu}_k$ and $\boldsymbol{\lambda}_k$, along with the penalty weight ρ_k:

$$\hat{\mathbf{x}}_k=\min_{\mathbf{x}}\mathscr{L}_{\rho_k}(\mathbf{x},\boldsymbol{\mu}_k,\boldsymbol{\lambda}_k). \tag{2.22}$$

At the end of each iteration, the Lagrange multiplier estimates are updated using the following update rules:

$$\begin{aligned}\boldsymbol{\mu}_{k+1}&=\max(\boldsymbol{\mu}_k+\rho_k(\mathbf{G}\hat{\mathbf{x}}_k-\mathbf{b}),\mathbf{0}),\\ \boldsymbol{\lambda}_{k+1}&=\boldsymbol{\lambda}_k+\rho_k(\mathbf{H}\hat{\mathbf{x}}_k-\mathbf{d}).\end{aligned} \tag{2.23}$$

Here, the element-wise max operator restricts the multiplier for the inequality constraints $\boldsymbol{\mu}$ to nonnegative values, whereas the multiplier for the equality constraints $\boldsymbol{\lambda}$ remains free of sign.

If the degree of violation has not decreased sufficiently at the end of the iteration, increasing the penalty weight ρ_k helps enforce feasibility. The measure of constraint violation at iteration k, following [47], which adopts the violation update rule from [4], is defined as[3]:

$$v_k=\max\left(\left\|\max(\mathbf{G}\hat{\mathbf{x}}_k-\mathbf{b},\frac{\boldsymbol{\mu}_k}{\rho_k})\right\|_\infty,\|\mathbf{H}\hat{\mathbf{x}}_k-\mathbf{d}\|_\infty\right), \tag{2.24}$$

where $\|\cdot\|_\infty$ denotes the infinity norm, which returns the maximum absolute value of the elements in a vector.

If the violation v_k does not decrease sufficiently, the penalty weight ρ_k is increased to force the solution towards the feasible space. The update rule for ρ_k is:

$$\rho_{k+1}=\begin{cases}\min(\eta\rho_k,\rho_{\max}), & \text{if } v_k>\tau v_{k-1},\\ \rho_k, & \text{otherwise.}\end{cases} \tag{2.25}$$

[3] In the original paper by Park and Van Hentenryck [47], the expression in (2.24) erroneously uses $\boldsymbol{\lambda}_k$ in place of $\boldsymbol{\mu}_k$.

Here, $\eta > 1$ is a scalar that controls the growth rate of the penalty weight, and $\rho_{\max}$ is the maximum allowed penalty weight. The tolerance scalar $\tau \in (0,1)$ determines how large of a decrease in violation is deemed sufficient.

If the violation is sufficiently small, $v_k \leq \epsilon$ for some predefined small threshold ϵ, the algorithm is considered converged and terminates.

In convex problems, such as LP problems, the ALM is proven to converge to a solution that satisfies the KKT conditions as $\rho \to \infty$. However, excessively large values of ρ can cause ill-conditioning, making the optimization landscape harder to navigate [39]. As a result, ALM is highly sensitive to the trajectory of ρ over iterations, requiring careful tuning of its update parameters.

2.6. Self-Supervised Primal-Dual Learning

PDL was first introduced by Park and Hentenryck [47, 46]. PDL is a machine learning method that trains on multiple instances of the same constrained optimization problem, learning to approximate the optimal primal and dual decision variables. The goal of PDL is to generalize beyond the training set so that it can predict near-optimal decision variables even for previously unseen problem instances.

$$P_\phi(\mathbf{z}) = \hat{\mathbf{x}}, \tag{2.26}$$

$$D_\psi(\mathbf{z}) = (\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}}). \tag{2.27}$$

Here, P_ϕ denotes the primal network with learnable network parameters ϕ, and D_ψ denotes the dual network with learnable network parameters ψ. Both networks take as input $\mathbf{z}$, a feature vector that encodes instance-specific information. In [47], $\mathbf{z}$ is the right-hand-side vector of the equality constraints.

As a self-supervised learning method, PDL does not require previously solved training examples. Its training algorithm is based on the ALM, following the same iterative structure. The outer iterations mirror the ALM, while the inner iterations train the primal and dual networks to approximate the corresponding decision variables. The remainder of this section details the PDL training algorithm, referencing its ALM counterparts in Section 2.5.

2.6.1. Primal Learning

The augmented Lagrangian is the loss function ℓ_P of the primal network P_ϕ. The network updates its parameters ϕ based on this loss, learning to predict the decision variables $\hat{\mathbf{x}}$ that minimize the augmented Lagrangian, similar to the ALM from (2.21):

$$\ell_P(\hat{\mathbf{x}} \mid \hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\lambda}}_k) = \mathscr{L}_{\rho_k}(\hat{\mathbf{x}}, \hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\lambda}}_k). \tag{2.28}$$

The Lagrange multiplier estimates are provided by the *frozen* dual network D_ψ^k, which is not updated and copies its parameters from the previous iteration.

2.6.2. Dual Learning

As in the ALM, the Lagrange multiplier estimates are updated each iteration for PDL. These serve as the targets for the dual network D_ψ. The dual loss function ℓ_D becomes:

$$\ell_D(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}} \mid \hat{\mathbf{x}}_k, \hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\lambda}}_k) = \|\boldsymbol{\mu} - \max(\boldsymbol{\mu}_k + \rho(\mathbf{G}\hat{\mathbf{x}}_k - \mathbf{b}), \mathbf{0})\|_2^2 + \|\boldsymbol{\lambda}_k + \rho(\mathbf{H}\hat{\mathbf{x}}_k - \mathbf{d})\|_2^2, \tag{2.29}$$

with the primal network frozen to provide $\hat{\mathbf{x}}_k$. This way, the dual network learns to predict the updated multiplier estimates from (2.5).

2.6.3. Penalty weight

The update rule for the penalty weight ρ is exactly the same as in the ALM, equation (2.25). However, the computation of the violation of the current iteration is changed slightly from equation (2.24) by taking the maximum violation of all the training samples:

$$v_k = \max_{\mathbf{z} \sim \mathscr{D}} v_k^{(\mathbf{z})}. \tag{2.30}$$

Here, $\mathscr{D} = \{\mathbf{z}^{(i)}\}_{i=1}^{S}$ refers to the set of Straining samples. Taking the maximum ensures that the worst-case violation is minimized across all samples.

2.7. Benders Decomposition

This section explains Benders decomposition [20], a technique that is researched in combination with PDL later in this thesis. Benders decomposition is well-suited for problems with complicating variables, which increase the computational complexity of an optimization problem, often due to their presence in many constraints or the introduction of non-convexities. When these variables are fixed, the remaining structure of the problem simplifies and can be decomposed into smaller, independent subproblems. For instance, fixing integer variables in a mixed-integer linear program reduces it to a linear program, which is computationally easier. In planning models, investment decisions often act as complicating variables. Once fixed, the operational decisions can be optimized efficiently as independent subproblems.

Consider the following generalized mixed-integer linear program with complicating variables:

$$\begin{aligned}
\underset{\mathbf{u},\,\mathbf{y}}{\text{minimize}} \quad & (\mathbf{c}^u)^\top \mathbf{u} + (\mathbf{c}^y)^\top \mathbf{y} && \text{(2.31a)}\\
\text{subject to} \quad & \mathbf{G}^u \mathbf{u} + \mathbf{G}^y \mathbf{y} \leq \mathbf{b}, && \text{(2.31b)}\\
& \mathbf{H}^u \mathbf{u} + \mathbf{H}^y \mathbf{y} = \mathbf{d}, && \text{(2.31c)}\\
& \mathbf{u} \in \mathbb{N}, && \text{(2.31d)}
\end{aligned}$$

where $\mathbf{u}$ is the vector of complicating variables, and $\mathbf{y}$ is the vector of non-complicating variables. Additionally, $\mathbf{c}^u$, $\mathbf{c}^y$ are the cost vectors operating on the complicating and non-complicating variables, respectively. Similarly, $\mathbf{G}^u$ and $\mathbf{G}^y$ are the inequality constraint matrices operating on the complicating and non-complicating variables, and $\mathbf{H}^u$ and $\mathbf{H}^y$ are the equality constraint matrices operating on the complicating and non-complicating variables.

2.7.1. Problem Decomposition

In (2.31), the complicating variables $\mathbf{u}$ and non-complicating variables $\mathbf{y}$ can be separated into a master problem and a subproblem. The master problem is then defined as:

$$\begin{aligned}
\underset{\mathbf{u}}{\text{minimize}} \quad & (\mathbf{c}^u)^\top \mathbf{u} + \theta(\mathbf{u}) && \text{(2.32a)}\\
\text{subject to} \quad & \mathbf{u} \geq 0, && \text{(2.32b)}
\end{aligned}$$

where $\theta(\mathbf{u})$ is the value function representing the optimal cost of the subproblem for a given instance of the complicating variables $\mathbf{u}$. The resulting subproblem is defined as $\theta(\mathbf{u}) =$

$$\begin{aligned}
\underset{\mathbf{y}}{\text{minimize}} \quad & (\mathbf{c}^y)^\top \mathbf{y} && \text{(2.33a)}\\
\text{subject to} \quad & \mathbf{G}^y \mathbf{y} \leq \mathbf{b} - \mathbf{G}^u \mathbf{u}, && \text{(2.33b)}\\
& \mathbf{H}^y \mathbf{y} = \mathbf{d} - \mathbf{H}^u \mathbf{u}. && \text{(2.33c)}
\end{aligned}$$

Here, it is important to note that the complicating variables $\mathbf{u}$ have become a parameter to the subproblem (2.33), and have therefore moved to the right-hand side of the constraints (2.33b)–(2.33c). This subproblem can typically be decomposed further into many independent subproblems, since the dependency on the complicating variables has been removed.

2.7.2. Approximating the Value Function using Benders Cuts

To solve the master problem, the minimum of the value function $\theta(\mathbf{u})$ must be determined. Ideally, the entire curve of $\theta(\mathbf{u})$ would be known, so that its minimum could be found directly. While each function value can be computed by solving the subproblem (2.33) for a fixed $\mathbf{u^{(k)}}$, naively evaluating every possible instance quickly becomes computationally intractable. Hence, a more efficient approximation strategy is required.

Given that $\theta(\mathbf{u})$ is convex, as proven in the book by Conejo et al. [13], its curve can be approximated using subgradients evaluated at selected points. These subgradients provide linear underestimators of the function. As more subgradients are collected at various values of $\mathbf{u^{(k)}}$, they collectively form a piecewise-linear approximation from below for $\theta(\mathbf{u})$.

A subgradient of a convex function $f(\mathbf{u})$ at a point $\mathbf{u}^{(k)}$ is any vector ∇ satisfying:

$$f(\mathbf{u}) \geq f(\mathbf{u}^{(k)}) + \nabla^\top(\mathbf{u} - \mathbf{u}^{(k)}) \quad \text{for all } \mathbf{u}. \tag{2.34}$$

Recall that dual variables, like derivatives, describe the sensitivity of the optimal value of the objective function to changes in the right-hand side of the constraints. In the context of Benders decomposition, this sensitivity is with respect to the complicating variables, which appear on the right-hand side of the subproblem constraints.

Following (2.16), the dual formulation of the subproblem $\theta(\mathbf{u})$ is defined as:

$$\underset{\boldsymbol{\mu} \geq 0, \boldsymbol{\lambda}}{\text{maximize}} \quad \boldsymbol{\mu}^\top (\mathbf{b} - \mathbf{G}^u \mathbf{u}) + \boldsymbol{\lambda}^\top (\mathbf{d} - \mathbf{H}^u \mathbf{u}) \tag{2.35a}$$

$$\text{subject to} \quad (\mathbf{G}^y)^\top \boldsymbol{\mu}^y + (\mathbf{H}^y)^\top \boldsymbol{\lambda} = \mathbf{c}^y. \tag{2.35b}$$

Given the optimal dual variables $\boldsymbol{\mu}^\star, \boldsymbol{\lambda}^\star$ for point $\mathbf{u}^{(k)}$, hereafter defined $\boldsymbol{\mu}^{(k)}$ and $\boldsymbol{\lambda}^{(k)}$, strong duality (Theorem 2) guarantees that the optimal value of the primal subproblem is equal to that of its dual. Thus, the subproblem value function satisfies:

$$\theta(\mathbf{u}^{(k)}) = \left(\boldsymbol{\mu}^{(k)}\right)^\top \left(\mathbf{b} - \mathbf{G}^u \mathbf{u}^{(k)}\right) + \left(\boldsymbol{\lambda}^{(k)}\right)^\top \left(\mathbf{d} - \mathbf{H}^u \mathbf{u}^{(k)}\right). \tag{2.36}$$

Since the dual variables represent the sensitivity of the optimal value of the subproblem with respect to the right-hand side of the constraints, and in this case the right-hand side depends on $\mathbf{u}$, the dual solution provides a subgradient of the value function $\theta(\mathbf{u})$ at point $\mathbf{u}^{(k)}$. The right-hand side constants $\mathbf{b}$ and $\mathbf{d}$ are omitted from the subgradient expression because they do not influence how the function changes with respect to the complicating variables $\mathbf{u}$. Thus, the equation can be reformulated as:

$$\theta(\mathbf{u}) \geq \theta(\mathbf{u}^{(k)}) - \left(\boldsymbol{\mu}^{(k)}\right)^\top \mathbf{G}^u \left(\mathbf{u} - \mathbf{u}^{(k)}\right) - \left(\boldsymbol{\lambda}^{(k)}\right)^\top \mathbf{H}^u \left(\mathbf{u} - \mathbf{u}^{(k)}\right), \tag{2.37}$$

which, when substituting $\theta(\mathbf{u}^{(k)})$ using (2.36), reduces to

$$\theta(\mathbf{u}) \geq \left(\boldsymbol{\mu}^{(k)}\right)^\top (\mathbf{b} - \mathbf{G}^u \mathbf{u}) + \left(\boldsymbol{\lambda}^{(k)}\right)^\top (\mathbf{d} - \mathbf{H}^u \mathbf{u}). \tag{2.38}$$

This equation is known as a *Benders cut*, and can be used to approximate the value function $\theta(\mathbf{u})$ from below.

2.7.3. Benders iterations

To find the minimum of the value function $\theta(\mathbf{u})$, it is not necessary to approximate the function accurately over its entire domain. In regions far from the optimum, the approximation only needs to be accurate enough to guide the optimization in the right direction. However, as the solution approaches the minimum, a more precise approximation becomes essential to identify the optimal point correctly.

This motivates the iterative structure of Benders decomposition. At each iteration, the accumulated Benders cuts provide a piecewise linear underestimator of $\theta(\mathbf{u})$. A new candidate solution $\mathbf{u}^{(k)}$ is obtained by solving the master problem, which minimizes the sum of the complicating costs and the current approximation of the value function. This candidate is then evaluated by solving the subproblem, which provides both the exact subproblem value and a new cut. The new cut is added to the master problem, improving the approximation for future iterations.

The resulting master problem is thus:

$$\underset{\mathbf{u},\,\alpha}{\text{minimize}} \quad (\mathbf{c}^u)^\top \mathbf{u} + \alpha \tag{2.39a}$$

$$\text{subject to} \quad \left(\boldsymbol{\mu}^{(k)}\right)^\top (\mathbf{b} - \mathbf{G}^u \mathbf{u}) + \left(\boldsymbol{\lambda}^{(k)}\right)^\top (\mathbf{d} - \mathbf{H}^u \mathbf{u}) \leq \alpha \quad k = 1, \dots, \nu, \tag{2.39b}$$

$$\mathbf{u} \in \mathbb{N}. \tag{2.39c}$$

This master problem minimizes over an additional auxiliary variable α, which is constrained from below by the ν Benders cuts. The optimal solution will therefore result in α taking on the minimum value of the current approximation of the value function $\theta(\mathbf{u})$ given by the cuts.

The algorithm is considered to have converged when the value α, obtained by minimizing the master problem, is sufficiently close to the true value obtained by solving the subproblem. That is,

$$\theta(\mathbf{u}^\star) - \alpha^\star \leq \epsilon, \tag{2.40}$$

for a specified tolerance $\epsilon \geq 0$.

2.7.4. Algorithm

The following pseudocode summarizes the Benders decomposition algorithm:

Algorithm 1 Benders Decomposition

Initialize iteration index $k \leftarrow 0$
Initialize cut set $\mathscr{C} \leftarrow \varnothing$
Initialize guess $\mathbf{u}^{(0)}$
repeat
    Solve subproblem for $\mathbf{u}^{(k)}$ to obtain:
        Optimal subproblem value $\theta(\mathbf{u}^{(k)})$
        Optimal dual variables $\boldsymbol{\mu}^{(k)}, \boldsymbol{\lambda}^{(k)}$
    Add Benders cut to $\mathscr{C}$:
        $\left(\boldsymbol{\mu}^{(k)}\right)^\top (\mathbf{b} - \mathbf{G}^u \mathbf{u}) + \left(\boldsymbol{\lambda}^{(k)}\right)^\top (\mathbf{d} - \mathbf{H}^u \mathbf{u}) \leq \alpha$
    Solve master problem with cuts $\mathscr{C}$ to obtain:
        New decision $\mathbf{u}^{(k+1)}$
        Approximation value $\alpha^{(k+1)}$
    $k \leftarrow k + 1$
until $\theta(\mathbf{u}^{(k)}) - \alpha^{(k)} \leq \epsilon$
return $\mathbf{u}^{(k)}$

3

Problem Formulations

The problem formulations used to support the experimental analysis throughout this thesis are outlined in this chapter. Section 3.1 introduces a quadratic programming problem that serves as a benchmark, both to reproduce PDL and to provide a valid basis for assessing generalization. Section 3.2 describes a generation expansion planning problem, which is the target for applying PDL in combination with Benders decomposition. Section 3.3 details the economic dispatch problem, which arises as the subproblem in the Benders decomposition of the generation expansion problem. Since PDL is applied to predict solutions to this subproblem, it serves as the basis for evaluating and refining PDL. Finally, Section 3.4 outlines the metrics used throughout this thesis to assess prediction quality.

Datasets for each problem are generated by perturbing selected components of a problem variant, thus generating many different instances. The specific perturbations applied to generate datasets are detailed in the corresponding experimental chapters.

3.1. Quadratic Programming Benchmark

The quadratic programming benchmark is identical to the one used by Park and Van Hentenryck [47] to evaluate PDL, which was adopted from a work by Donti, Rolnick, and Kolter [18]. This problem will be used to validate our implementation of PDL on a problem where its performance is known. Subsequently, it will serve as a benchmark for exploring the effect of changes to the problem formulation itself. The problem is defined as follows:

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^Q}{\text{minimize}} \quad & \tfrac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{r}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{G}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{H}\mathbf{x} = \mathbf{d}. \end{aligned} \tag{3.1}$$

In this formulation, $\mathbf{x} \in \mathbb{R}^Q$ are the decision variables, $\mathbf{Q} \in \mathbb{R}^{Q \times Q}$ denotes a diagonal, positive semi-definite matrix with entries sampled uniformly from the interval $[0,1)$. The constraint matrices $\mathbf{G} \in \mathbb{R}^{M \times Q}$ and $\mathbf{H} \in \mathbb{R}^{O \times Q}$ are generated with entries drawn from the standard normal distribution. The RHS of the equality constraints, $\mathbf{d}$, is sampled uniformly from the interval $[-1,1]$. The RHS of the inequality constraints is defined as

$$\mathbf{b}_i = \sum_j \left| (\mathbf{G}\mathbf{H}^\dagger)_{ij} \right|, \tag{3.2}$$

where $\mathbf{H}^\dagger$ denotes the Moore–Penrose pseudoinverse of $\mathbf{H}$. This formulation ensures the feasibility of the problem instances, which is explained in more detail by Donti, Rolnick, and Kolter [18].

Donti, Rolnick, and Kolter [18] and Park and Van Hentenryck [47] both construct datasets of problem instances by varying the right-hand side of the equality constraints, $\mathbf{d}$, while keeping all other components of the problem fixed. Thus, in both cases, each instance corresponds to a unique value of $\mathbf{d}$. This thesis generally follows the same formulation for generating instances, unless explicitly stated otherwise.

3.2. Generation Expansion Planning

Generation expansion planning is an optimization problem that aims to determine the optimal installation of power generation units to meet forecasted demand while minimizing costs over a specified planning horizon. Typically, four questions are answered [45]:

- what types of generators to invest in;
- how many units of each type to install;
- where to locate them;
- and, in some cases, when to build them.

Many variations of the generation expansion planning problem exist, differing in the length of the planning horizon, the granularity of the topology of the power grid, the level of physical detail modeled in the underlying electrical processes, the treatment of uncertainty, and the geographic scale. This thesis considers a simplified deterministic version in which all investments are made once at the beginning of the planning period. The objective is to minimize both investment and operational costs. Operational costs include generation costs and a penalty for unmet demand, computed over the entire planning horizon. The specific problem formulation used in this thesis is introduced in this section.

It is worth noting that, given a fixed installed capacity, the problem reduces to an economic dispatch problem, which optimizes generation to meet demand at minimum operational cost. The economic dispatch problem is formalized in Section 3.3.

The generation expansion planning problem considered in this thesis is based on the formulation used by Arnoldus [5]. Although relatively simple, it captures key elements commonly found in generation expansion planning problems, such as investment and operational decisions, as well as the inclusion of renewable energy, as outlined in reviews such as [7, 45].

The problem consists of four main components: nodes $n \in \mathcal{N}$, generators $g \in \mathcal{G}$, transmission lines $l \in \mathcal{L}$, and time steps $t \in \mathcal{T}$. Each node represents a group of consumers with a corresponding energy demand. A set of generators $\mathcal{G}_n \subseteq \mathcal{G}$ is associated with each node and is available to supply energy. Each generator is associated with exactly one node. Generators are defined by their technology type, including but not limited to oil-fired plants, gas-fired plants, wind farms, and solar panels. Additionally, energy can be exchanged between nodes that are connected by a transmission line l. Energy can be exchanged bidirectionally through transmission lines; however, the transmission lines are modeled in a directed manner. A positive line flow indicates energy traveling in the default direction, whereas a negative line flow represents energy traveling in the opposite direction. Each node has a possibly empty set of incoming lines $\mathcal{L}_n^{\text{in}} \subseteq \mathcal{L}$ and outgoing lines $\mathcal{L}_n^{\text{out}} \subseteq \mathcal{L}$.

The objective of the generation expansion planning problem is to determine which generator types should be installed at each node and how many units of each type in order to minimize the total cost. The installation of generators incurs investment costs, while their operation results in additional operating costs. The total cost is computed over a planning horizon, defined by the set of time steps $\mathcal{T} = \{1, \dots, T\}$, where each time step can represent a distinct unit of time, such as an hour, a day, or a month. An example illustrating a solution to a generation expansion planning problem is given in Figure 3.1.

The components of the generation expansion planning problem are explained below. The complete problem formulation is shown in (3.5). The constraints in the rest of this chapter are formulated in standard form, with all inequalities expressed as $\leq$ and constant right-hand sides, to maintain consistency with the formulation used throughout this thesis and in the PDL algorithm. Furthermore, lowercase symbols denote decision variables, while uppercase symbols denote input parameters. The

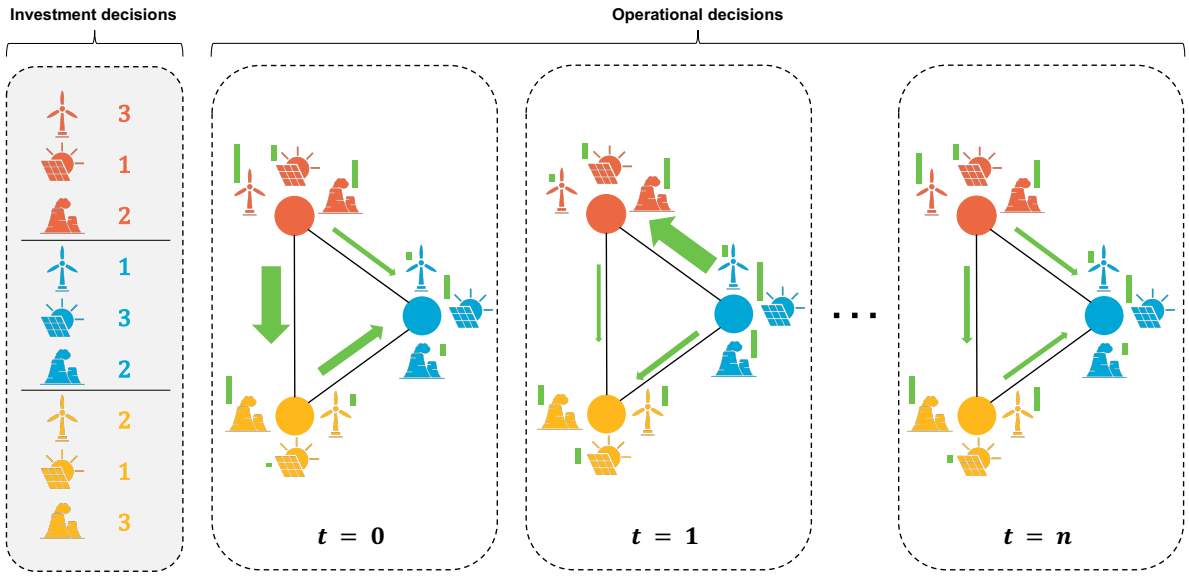

Figure 3.1: Example of a solution to a generation expansion planning problem. The left panel shows the investment decisions at each node, indicating the number of units installed per generator type. The right panels illustrate the corresponding operational decisions: generator outputs and power flows across the network over n time steps.

specific input parameters used can be found in the codebase [26], an overview of which is provided in Appendix A.

3.2.1. Objective function

The optimization is performed over four types of decision variables: the number of units to be installed for each generator $\mathbf{u} \in \mathbb{N}^{|\mathcal{G}|}$, the power production of each generator at each time step $\mathbf{p} \in \mathbb{R}^{|\mathcal{G}|\times|\mathcal{T}|}$, the amount of energy transmitted on each line per time step $\mathbf{f} \in \mathbb{R}^{|\mathcal{L}|\times|\mathcal{T}|}$, and the amount of unmet demand at each node and time step $\mathbf{e} \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{T}|}$. This unmet demand is penalized in the objective function to encourage meeting all demand.

As explained earlier, the objective function consists of investment costs and operating costs. The investment costs are computed as follows:

$$\sum_{g\in\mathcal{G}} C_g^{\text{inv}} \cdot P_g^{\max} \cdot u_g, \tag{3.3}$$

where C_g^{inv} [€/MW] denotes the investment cost per megawatt for generator g, $P_g^{\max}$ [MW] is the capacity of a single unit, and u_g is the number of units installed at generator g.

The operating costs are given by:

$$W \cdot \left(\sum_{g\in\mathcal{G},t\in\mathcal{T}} C_g^{\text{var}} \cdot p_{g,t} + \sum_{n\in\mathcal{N},t\in\mathcal{T}} C^{\text{e}} \cdot e_{n,t} \right). \tag{3.4}$$

Here, W denotes the period weight, which in this case corresponds to the number of hours represented by each time step t. The variable production cost is given by C_g^{var} [€/MWh], representing the cost of producing one megawatt-hour of electricity by generator g. The variable $p_{g,t}$ [MW] denotes the amount of energy produced by generator g at time step t. The cost of unmet demand is represented by C^{e} [€/MW], and $e_{n,t}$ [MW] denotes the amount of demand that is not served at node n at time step t. The inclusion of the unmet demand slack variable e is commonly used in Benders decomposition to prevent infeasible subproblems when the chosen investment decisions are insufficient.

3.2.2. Constraints

This section describes the different types of constraints enforced during the minimization process. Constraints (3.5b)-(3.5h) impose bounds on the decision variables, while the remaining constraints involve

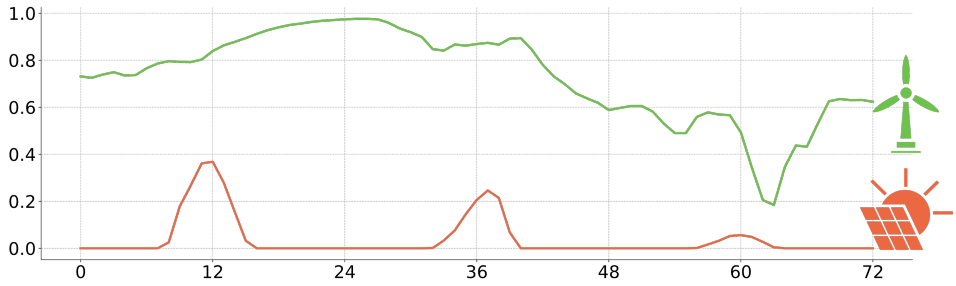

Figure 3.2: Example of the available generation capacity profiles $A_{g,t}$ for a wind-based generator (green) and a solar-based generator (orange). The data corresponds to 72 hourly time steps sampled from the case study considered throughout this thesis.

more complex relationships.

The generation production $p_{g,t}$ [MW] of generator g at time step t is constrained from below by zero, as specified in Constraint (3.5b), and from above by $A_{g,t} \cdot P_g^{\max} \cdot u_g$, as defined in Constraint (3.5c). The parameter $A_{g,t} \in [0,1]$ represents the fraction of the maximum generation capacity that is available at generator g during time step t. This value typically varies for renewable energy sources, which depend on external conditions such as wind or sunlight, but is equal to one for conventional (non-renewable) generators. An example illustrating possible profiles of available generation is shown in Figure 3.2. It should be noted that in Constraint (3.6c), the term defining the capacity is part of the constraint matrix, as it includes the investment decision variable u_g, and therefore appears on the left-hand side of the constraint.

The line flow $f_{l,t}$ [MW] through transmission line l at time step t is bounded below by the negative import capacity $-\underline{F}_l$ [MW] in Constraint (3.5d), and above by the export capacity $\overline{F}_l$ [MW] in Constraint (3.5e).

The unmet demand $e_{n,t}$ [MW] at node n at time step t is bounded below by 0 through Constraint (3.5f) and above by the demand $D_{n,t}$ [MW] at node n at time step t in Constraint (3.5g). The demand typically varies over time.

The number of units installed or *investments*, $u_g \in \mathbb{R}$, at generator g is constrained to be nonnegative, as specified in Constraint (3.5h). There is no upper bound on this variable. In typical generation expansion planning problems, u_g is restricted to integer values, as fractions of generation units are physically impossible, resulting in a mixed-integer linear programming problem.

Lastly, Constraint (3.5i) enforces the energy balance at each node n and time step t, ensuring that local supply equals local demand. The balance includes total generation production $\sum_{g \in \mathcal{G}_n}$ [MW], net transmission $\sum_{l \in \mathcal{L}_n^{\text{in}}} f_{l,t} - \sum_{l \in \mathcal{L}_n^{\text{out}}} f_{l,t}$ [MW], and the slack variable for unmet demand $e_{n,t}$ [MW]. The inclusion of unmet demand guarantees that the problem remains feasible, provided all decision variables satisfy their respective bounds.

Inter-period constraints, including those related to ramping and storage, are omitted to avoid excessive complexity, as the concepts introduced in this work are applied for the first time in this domain and are intended to serve as a proof of concept for future extensions.

The complete problem is formulated below.

$$\underset{\mathbf{u},\mathbf{p},\mathbf{f},\mathbf{e}}{\text{minimize}} \quad \underbrace{\sum_{g\in\mathscr{G}} C_g^{\text{inv}} \cdot P_g^{\max} \cdot u_g}_{\text{investment costs}} + W \cdot \underbrace{\left(\sum_{g\in\mathscr{G},t\in\mathscr{T}} C_g^{\text{var}} \cdot p_{g,t} + \sum_{n\in\mathscr{N},t\in\mathscr{T}} C^{\text{e}} \cdot e_{n,t} \right)}_{\text{operating costs}} \tag{3.5a}$$

subject to

$$-p_{g,t} \le 0 \qquad \forall g \in \mathscr{G}, t \in \mathscr{T}, \tag{3.5b}$$

$$p_{g,t} - A_{g,t} \cdot P_g^{\max} \cdot u_g \le 0 \qquad \forall g \in \mathscr{G}, t \in \mathscr{T}, \tag{3.5c}$$

$$-f_{l,t} \le \underline{F}_l \qquad \forall l \in \mathscr{L}, t \in \mathscr{T}, \tag{3.5d}$$

$$f_{l,t} \le \overline{F}_l \qquad \forall l \in \mathscr{L}, t \in \mathscr{T}, \tag{3.5e}$$

$$-e_{n,t} \le 0 \qquad \forall n \in \mathscr{N}, t \in \mathscr{T}, \tag{3.5f}$$

$$e_{n,t} \le D_{n,t} \qquad \forall n \in \mathscr{N}, t \in \mathscr{T}, \tag{3.5g}$$

$$-u_g \le 0 \qquad \forall g \in \mathscr{G}, \tag{3.5h}$$

$$\sum_{g\in\mathscr{G}_n} p_{g,t} + \sum_{l\in\mathscr{L}_n^{\text{in}}} f_{l,t} - \sum_{l\in\mathscr{L}_n^{\text{out}}} f_{l,t} + e_{n,t} = D_{n,t} \qquad \forall n \in \mathscr{N}, t \in \mathscr{T}, \tag{3.5i}$$

$$u_g \in \mathbb{Z}. \tag{3.5j}$$

Since both the demand $D_{n,t}$ and available generation $A_{g,t}$ vary over time, each subset of timesteps, accompanied with an investment decision $\mathbf{u}$ defines a distinct instance of the economic dispatch problem introduced in the next section.

3.3. Economic Dispatch

When the investment decisions are fixed, that is, when u_g is known, the generation expansion planning problem reduces to an economic dispatch problem. This problem arises from the subproblems generated by applying Benders decomposition to the generation expansion planning problem. In this setting, the objective is to minimize operating costs by determining the optimal generation $p_{g,t}^\star$, line flows $f_{l,t}^\star$, and unmet demand $e_{n,t}^\star$ for all nodes $n \in \mathscr{N}$ and time steps $t \in \mathscr{T}$.

The formulation of the economic dispatch problem, as embedded within the generation expansion planning problem, is provided below, retaining only the relevant components.

$$\underset{\mathbf{p},\mathbf{f},\mathbf{e}}{\text{minimize}} \quad \sum_{g\in\mathscr{G},t\in\mathscr{T}} C_g^{\text{var}} \cdot p_{g,t} + \sum_{n\in\mathscr{N},t\in\mathscr{T}} C^{\text{e}} \cdot e_{n,t} \tag{3.6a}$$

subject to

$$-p_{g,t} \le 0 \qquad \forall g \in \mathscr{G}, t \in \mathscr{T}, \tag{3.6b}$$

$$p_{g,t} \le A_{g,t} \cdot P_g^{\max} \cdot u_g \qquad \forall g \in \mathscr{G}, t \in \mathscr{T}, \tag{3.6c}$$

$$-f_{l,t} \le \underline{F}_l \qquad \forall l \in \mathscr{L}, t \in \mathscr{T}, \tag{3.6d}$$

$$f_{l,t} \le \overline{F}_l \qquad \forall l \in \mathscr{L}, t \in \mathscr{T}, \tag{3.6e}$$

$$-e_{n,t} \le 0 \qquad \forall n \in \mathscr{N}, t \in \mathscr{T}, \tag{3.6f}$$

$$e_{n,t} \le D_{n,t} \qquad \forall n \in \mathscr{N}, t \in \mathscr{T}, \tag{3.6g}$$

$$\sum_{g\in\mathscr{G}_n} p_{g,t} + \sum_{l\in\mathscr{L}_n^{\text{in}}} f_{l,t} - \sum_{l\in\mathscr{L}_n^{\text{out}}} f_{l,t} + e_{n,t} = D_{n,t} \qquad \forall n \in \mathscr{N}, t \in \mathscr{T}, \tag{3.6h}$$

$$u_g \in \mathbb{Z}. \tag{3.6i}$$

The period weight W has been removed from the objective function since it has no influence on the optimal primal decision variables. However, the optimal dual decision variables will have to be scaled by a factor of $\frac{1}{W}$ to match those from the original generation expansion planning problem.

3.4. Evaluation Metrics

This thesis will use various metrics to evaluate the quality of predicted solutions. This section explains the necessity of each metric and its computation.

3.4.1. Optimality Gap

The optimality gap gives a sense of how close the objective value of a prediction is to the known optimal objective value, thereby serving as a direct measure of quality. Given predicted decision variables $\hat{\mathbf{x}}$, known optimal decision variables $\mathbf{x}^\star$ and objective function f, the optimality gap is calculated as follows:

$$\Delta_O \coloneqq \frac{f(\hat{\mathbf{x}}) - f(\mathbf{x}^\star)}{|f(\mathbf{x}^\star)|} \cdot 100\%. \tag{3.7}$$

This formulation applies to both the primal and dual optimality gaps when substituting the corresponding objective functions and decision variables.

3.4.2. Duality Gap

When both primal and dual predictions are available, we can assess their optimality without knowing the exact optimal objective value. By strong duality (Theorem 2), the primal and dual objective values coincide at the optimum of a linear program. In a minimization problem, the primal objective approaches the optimum from above, while the dual objective approaches from below. For a feasible, suboptimal primal-dual pair, the optimal value lies between them. Thus, the duality gap provides a measure of optimality. The formula for the duality gap is given below:

$$\Delta_D \coloneqq \frac{|f(\hat{\mathbf{x}}) - f_D(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}})|}{|f(\hat{\mathbf{x}})|} \cdot 100\%. \tag{3.8}$$

Here, f_D denotes the dual objective function, and $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\lambda}}$ are the predicted dual decision variables. When the solutions are infeasible, the dual objective value may exceed the primal objective value. To avoid ambiguity, both objective values will be reported alongside the duality gap.

3.4.3. Constraint Violations

Objective values alone do not provide a complete picture of solution quality, as they do not consider constraint violations and thus do not provide a measure of feasibility. A good solution exhibits both a small optimality gap and minimal constraint violations.

We distinguish between violations of equality and inequality constraints, as equality constraints are generally more challenging because they must be satisfied exactly. To quantify violations, we report the mean violation over all constraints and problem instances to understand overall feasibility, and the mean of the maximum violation per instance to capture the worst violation per instance. A solution may satisfy most constraints but still be far from feasible if a single constraint is violated significantly, which might not be reflected in the mean.

Given a set of outputs $\hat{\mathscr{X}} = \left\{\hat{\mathbf{x}}^{(s)} = P_\phi(\mathbf{z}^{(s)}) \;\middle|\; \mathbf{z}^{(s)} \in \mathscr{D}\right\}$, computed over the input features $\mathbf{z}$ from the training set $\mathscr{D}$, the formulas for the equality constraint violation metrics are as follows:

$$V^{\text{eq}}_{\text{mean}} \coloneqq \underset{s,i}{\text{mean}}\left(|\mathbf{H}^\top \hat{\mathbf{x}}^{(s)} - \mathbf{d}|_i\right) \qquad \text{and} \qquad V^{\text{eq}}_{\text{max}} \coloneqq \underset{s}{\text{mean}}\left(\max_i\left(|\mathbf{H}^\top \hat{\mathbf{x}}^{(s)} - \mathbf{d}|_i\right)\right), \tag{3.9}$$

where $\mathbf{H}$ and $\mathbf{d}$ respectively are the equality constraint matrix and equality right-hand side. Samples are indexed by s, and specific constraint violations by i. The inequality constraint violation metrics are computed by:

$$V^{\text{ineq}}_{\text{mean}} \coloneqq \underset{s,i}{\text{mean}}\left((\mathbf{G}^\top \hat{\mathbf{x}}^{(s)} - \mathbf{b})^+_i\right) \qquad \text{and} \qquad V^{\text{ineq}}_{\text{max}} \coloneqq \underset{s}{\text{mean}}\left(\max_i\left((\mathbf{G}^\top \hat{\mathbf{x}}^{(s)} - \mathbf{b})^+_i\right)\right), \tag{3.10}$$

where $\mathbf{G}$ and $\mathbf{b}$ respectively are the inequality constraint matrix and inequality right-hand side, and $(\cdot)^+ = \max(\cdot, 0)$. Again, samples are indexed by s, and specific constraint violations by i.

3.5. Conclusion

This chapter presented the problem formulations that support the experimental analysis conducted in this thesis. A quadratic programming benchmark was introduced to reproduce prior results, provide a controlled setting for analyzing PDL, and test modifications in a setting where the method is known to perform well. The generation expansion planning problem was formulated as the primary case study, serving as the target for applying PDL in combination with Benders decomposition. The associated economic dispatch problem, which arises as the subproblem in the Benders decomposition, was introduced as the core task to which PDL is applied. As such, it serves as the foundation for evaluating and refining PDL throughout this thesis. Finally, a set of evaluation metrics was defined to assess solution quality in terms of optimality, feasibility, and duality. These formulations provide the basis for the experiments and analyses presented in the following chapters.

4

Evaluating Primal-Dual Learning under Variations in Problem Structure

This chapter presents an exploratory analysis on the performance of PDL by introducing variations to the convex quadratic programming benchmark problem (3.1). In previous work by Park and Van Hentenryck [47, 46], PDL is evaluated on a range of optimization problems by perturbing specific problem components to generate datasets of problem instances. PDL is then trained to predict solutions for previously unseen instances.

The types of variations explored in these works differ between problem types. In the initial publication [47], the right-hand-side constants of the equality constraints are varied for a convex and a non-convex quadratic programming benchmark. In contrast, a different quadratically constrained quadratic programming benchmark considered in the same work varies the linear component of the objective function. A subsequent publication [46] introduces a security-constrained optimal power flow[1] problem, in which a linear cost vector in the objective function is similarly varied between instances. PDL is shown to perform well in all of these scenarios.

However, no prior work has evaluated PDL on problems where the *constraint matrix* varies between instances. This case is worth considering, as such variations arise in many real-world problem formulations, such as in the generation expansion planning problem presented in (3.5). In this formulation, both the demand $\mathbf{D}$ and the available generation $\mathbf{A}$ typically differ between the instances of the problem. The available generation $\mathbf{A}$ is incorporated into the constraint matrix, while $\mathbf{D}$ appears on the right-hand side of all equality and some inequality constraints.

Moreover, the performance of PDL has not been evaluated on problems where the *quadratic* component of the objective function varies between instances. Such variation is common, for example, in economic dispatch problems where generation costs are modeled as quadratic functions [51, 38, 12].

The remainder of this chapter evaluates the performance of PDL under two types of structural variation between data instances: changes to elements of the inequality constraint matrix and changes to elements of the quadratic term of the objective function. Section 4.1 describes the experimental setup, detailing how these structural variations are introduced into the convex quadratic programming benchmark. Section 4.2 presents the corresponding results, which are then discussed in Section 4.3. Finally, Section 4.4 concludes the findings.

4.1. Experimental Setup

As no source code is provided alongside the original PDL publication [47], the method was first reimplemented from scratch. Our reproduction of PDL is evaluated on the original convex and non-convex quadratic programming benchmarks, yielding results that closely match those reported in the origi-

[1] Security-constrained optimal power flow is a class of energy systems optimization problems.

nal work, thereby validating the implementation. A comprehensive overview of the reproduction is provided in Appendix B.

4.1.1. Construction of Original Problem Instances

The original dataset $\mathscr{D}$, which is the same as those used in [47], is constructed by generating multiple instances of the convex quadratic programming benchmark problem introduced in 3.1 through variation of the right-hand side of the equality constraints $\mathbf{d}$. The right-hand sides of the equality constraints $\mathbf{d}$ then serve as the input to both the primal and dual networks:

$$\mathscr{D} = \left\{\mathbf{z}^{(i)} = \mathbf{d}^{(i)}\right\}_{i=1}^{S}. \tag{4.1}$$

The dataset contains $S = 10000$ instances, each with $Q = 100$ decision variables, and $M = 50$ inequality and $O = 50$ equality constraints. The dataset is split into training data, validation data, and test data with respective sizes of 80%, 10%, and 10%.

4.1.2. Construction of Modified Problem Instances

In each experiment, the dimensionality of the input features $\mathbf{z}$, determined by the number of varying elements, is kept constant to ensure that the complexity of the learning task does not increase due to changes in input size. Specifically, 50 elements are varied across instances, matching the number of varying elements in the original problem. The positions of these elements are fixed, so that the same entries are modified in every instance.

Three different strategies are considered for selecting the elements of the inequality constraint matrix $\mathbf{G}$ that are varied across instances:

- *Row*: varying the first 50 entries of the first row.
- *Column*: varying the entire first column, which consists of 50 entries.
- *Random*: Varying a random selection of 50 elements across the entire matrix.

Only a single variation strategy, referred to as *objective* in the results, is used for the quadratic term $\mathbf{Q}$, as it is a diagonal matrix. Again, a random selection of 50 diagonal entries is varied between instances.

For instances where the inequality constraint matrix $\mathbf{G}$ is varied (row, column, random), the input features $\mathbf{z}$ are sampled from the standard normal distribution, following the generation of $\mathbf{G}$. For instances where the quadratic term $\mathbf{Q}$ is varied (objective), the entries of $\mathbf{z}$ are sampled uniformly from [0,1), following the generation of $\mathbf{Q}$.

We hypothesize that the performance of PDL will deteriorate when elements of the constraint matrix are varied rather than when the right-hand side is altered. When the right-hand side varies, the directions of the constraints remain fixed, so although the feasible region's size, aspect ratio, and position may change, its overall orientation and structure are preserved. In contrast, varying the coefficients of the constraint matrix changes the directions of the constraints, thereby fundamentally altering the feasible region and, more importantly, the interactions among the decision variables.

Similarly, we hypothesize that performance worsens when the quadratic term of the objective function varies. Varying the linear term shifts the location of the optimum but preserves the curvature of the objective function. In contrast, changing the quadratic term alters the curvature, reshaping the optimization landscape entirely.

The performance of PDL on these modified problems is compared to its performance on the original convex quadratic programming benchmark problem introduced in Section 3.1. Each experiment is repeated five times to minimize the effect of randomness.

4.1.3. Neural Network Architecture and Hyperparameters

The architecture of the primal and dual neural networks is kept consistent with the original work by Park and Van Hentenryck [47], in which a multi-layer perceptron is used for both.

Two sets of experiments are conducted. In the first, the hyperparameters are identical to those used in the original study. Since these were tuned for the case where only the right-hand side of the constraints is varied, PDL is expected to perform particularly well on that problem type. To enable a fair comparison across all problem variations, a second set of experiments is conducted using a randomly selected set of hyperparameters for each of the five repeats. This random set is held fixed across all problem types within that repeat.

The full details of the neural network architecture and hyperparameters are provided in Appendix C.2.

An overview of the codebase [26] and the link to the repository are provided in Appendix A.

4.2. Results

The results of both experiments are presented in Table 4.1. Additionally, the results of each repeat with random hyperparameters are shown individually in Table 4.2. In each table, the metrics introduced in Section 3.4 are used to evaluate the performance of PDL. The duality gap is not included, as the quality of the predicted dual solutions is not evaluated in this chapter.

Table 4.1: Performance of PDL on variations of a convex quadratic programming benchmark. In the original problem, instances vary only in the right-hand side constants of the equality constraints. The Row, Column, and Random variants introduce variation in the inequality constraint matrix, with elements selected row-wise, column-wise, or randomly, respectively. The Objective variant varies elements of the quadratic objective function. Performance is evaluated using the relative optimality gap (in %), and the maximum and mean violations of the inequality and equality constraints. For all metrics, lower values indicate better performance. Bold values highlight the best result in each column, excluding optimal and predicted objectives, as these cannot be compared across rows directly. To eliminate any bias from hyperparameters tuned for the original problem, results are reported for both the original hyperparameters and randomly sampled ones. Each setting is averaged over five repeats to reduce the effect of randomness; standard deviations are also reported, but standard deviations of 0 are omitted for clarity.

	Objective			Inequality		Equality	
Experiment	Optimal	Predicted	Gap (%)	Max	Mean	Max	Mean
Original hyperparameters							
Original	-15.037	-15.022 ± 0.002	$\mathbf{0.104 \pm 0.016}$	**0.002**	**0.000**	**0.006(0.000)**	**0.002**
Row	-15.570	-15.507 ± 0.076	0.411 ± 0.495	0.115 ± 0.027	0.004 ± 0.001	0.030 ± 0.018	0.009 ± 0.005
Column	-15.358	-13.829 ± 1.112	10.030 ± 7.435	0.050 ± 0.037	0.001 ± 0.001	0.007 ± 0.001	**0.002**
Random	-15.602	-14.667 ± 0.402	5.992 ± 2.579	0.145 ± 0.048	0.003 ± 0.001	0.026 ± 0.035	0.008 ± 0.011
Objective	-15.630	-14.329 ± 0.178	8.366 ± 1.144	0.007 ± 0.003	**0.000**	0.008 ± 0.002	0.003 ± 0.001
Random hyperparameters							
Original	-15.037	-14.944 ± 0.074	$\mathbf{0.620 \pm 0.490}$	$\mathbf{0.019 \pm 0.016}$	$\mathbf{0.001 \pm 0.001}$	0.018 ± 0.014	0.006 ± 0.005
Row	-15.570	-15.151 ± 0.409	2.692 ± 2.627	0.094 ± 0.033	0.005 ± 0.002	0.098 ± 0.016	0.031 ± 0.005
Column	-15.358	-14.777 ± 0.417	3.752 ± 2.698	0.125 ± 0.085	0.006 ± 0.005	0.069 ± 0.038	0.022 ± 0.012
Random	-15.602	-14.764 ± 0.656	5.361 ± 4.195	0.073 ± 0.046	0.003 ± 0.003	0.058 ± 0.045	0.019 ± 0.014
Objective	-15.630	-14.434 ± 0.163	7.687 ± 1.033	0.036 ± 0.038	0.002 ± 0.002	$\mathbf{0.013 \pm 0.005}$	$\mathbf{0.004 \pm 0.002}$

Table 4.1 shows that PDL performs best on the original convex quadratic programming problem when using the original hyperparameters. This is reflected in consistently lower average optimality gaps and constraint violations compared to the modified problem types. With random hyperparameters, the results are more mixed, although the original problem generally continues to perform well across the different metrics.

A more detailed view is provided when considering each individual repeat with random hyperparameters in Table 4.2. In most cases, PDL achieves the lowest metric values on the original problem, as seen by the number of rows where the original configuration produces the best result.

4.3. Discussion

The results show a clear trend: for all problems in which the inequality constraint matrix varies between instances, PDL struggles to learn solutions with low constraint violations. In contrast, when the objective function varies, PDL has difficulty achieving near-optimal objective values. This suggests that the network has more difficulty learning the components of the problem that change across instances, which is to be expected.

Table 4.2: Detailed performance of PDL on the convex quadratic programming benchmark for each individual random-hyperparameter repeat (1–5). Each column reports the optimal and predicted objective values, the relative optimality gap (in %), and the maximum and mean violations of the inequality and equality constraints; lower values indicate better performance. Results for each repeat correspond to a single run with a unique random hyperparameter draw. Standard deviations are omitted since each entry reflects one experiment.

			Ineq. constraint matrix			Q. term
Repeat	Metric	Original	Row	Column	Random	Objective
Repeat 0	Gap (%)	**0.783**	5.702	5.842	8.041	7.494
	Ineq. Max	**0.011**	0.062	0.072	0.057	0.017
	Ineq. Mean	**0.001**	0.003	0.003	0.002	**0.001**
	Eq. Max	**0.011**	0.104	0.105	0.112	0.015
	Eq. Mean	**0.003**	0.033	0.033	0.036	0.005
Repeat 1	Gap (%)	-0.029	**-0.702**	0.136	0.164	6.550
	Ineq. Max	**0.042**	0.140	0.290	0.118	0.106
	Ineq. Mean	**0.004**	0.008	0.016	0.006	0.005
	Eq. Max	0.044	0.106	0.038	0.031	**0.022**
	Eq. Mean	0.014	0.032	0.010	0.009	**0.007**
Repeat 2	Gap (%)	0.485	**-0.089**	0.784	0.488	6.672
	Ineq. Max	**0.033**	0.127	0.122	0.131	0.047
	Ineq. Mean	**0.001**	0.006	0.005	0.007	0.002
	Eq. Max	0.023	0.066	**0.010**	0.027	0.011
	Eq. Mean	0.007	0.021	**0.003**	0.010	**0.003**
Repeat 3	Gap (%)	**0.411**	5.044	5.891	10.227	9.197
	Ineq. Max	**0.001**	0.061	0.070	0.006	0.004
	Ineq. Mean	**0.000**	0.003	0.003	**0.000**	**0.000**
	Eq. Max	**0.007**	0.102	0.095	0.008	0.010
	Eq. Mean	**0.002**	0.032	0.030	0.003	0.003
Repeat 4	Gap (%)	**1.450**	3.505	6.109	7.886	8.522
	Ineq. Max	**0.008**	0.078	0.071	0.056	0.009
	Ineq. Mean	**0.000**	0.004	0.004	0.002	**0.000**
	Eq. Max	**0.005**	0.110	0.100	0.111	0.009
	Eq. Mean	**0.001**	0.034	0.032	0.035	0.003

No significant difference is observed between the three strategies for varying elements in the constraint matrix (row, column, or random selection), indicating that PDL is sensitive to the presence of variation in the constraint matrix itself, regardless of the specific pattern of variation.

Overall, the original quadratic programming benchmark, where only the right-hand side of the equality constraints varies, appears to be the easiest setting for PDL. Even with randomly sampled hyperparameters, while another variation may occasionally yield a better result on a single metric, the original problem generally performs better across the remaining metrics.

It is important to note, however, that the variations introduced in these experiments are likely more extreme than those encountered in many real-world applications. In practice, changes to the constraint matrix may affect only a small subset of entries rather than entire rows or columns, as is the case for the generation expansion planning problem (3.5), where .

Moreover, the hyperparameters used in these experiments may not be optimal for each variation type. A well-tuned PDL model may therefore generalize more effectively in realistic settings, even though variations in the constraint matrix and the quadratic term of the objective function remain inherently more challenging than changes in the right-hand side or the linear term.

4.4. Conclusion

This chapter presented an exploratory analysis demonstrating that PDL performs well when the right-hand side of the constraints or the linear term of the objective function varies between problem instances, but struggles when variation is introduced in the constraint matrix or the quadratic component of the objective function. These findings indicate that problems exhibiting such structural variation are likely more difficult for PDL. However, it cannot be concluded that PDL is incapable of learning effectively in these cases without further investigation.

In particular, the results suggest that directly applying PDL to the generation expansion planning problem introduced in Section 3.2, where the constraint matrix varies across instances, may be difficult without further adaptation. As addressing these challenges is not within the direct scope of this research, the remainder of this thesis focuses on the economic dispatch problem introduced in Section 3.3, where only the right-hand side varies across problem instances.

5

Primal-Dual Learning for Economic Dispatch

The previous chapter identified that directly applying PDL to the generation expansion planning problem (3.5) may be challenging due to variations in the constraint matrix across problem instances. To address this, the current chapter focuses on the simpler economic dispatch problem (3.6), in which only the right-hand side varies between instances, a setup in which PDL is known to perform well. This offers a more controlled setting for evaluating and extending PDL.

Despite its seemingly simple mathematical structure, economic dispatch remains challenging from a learning perspective. Small variations in the right-hand side can lead to discrete shifts in which constraints are active, resulting in non-smooth and piecewise-linear solution behavior that can be difficult for neural networks to capture, for example, when a generator reaches its capacity. This type of abrupt behavior is less common in the previously studied quadratic programming benchmark problem, where variables influence all constraints more evenly and both constraint matrices and right-hand sides are generated from predictable statistical distributions rather than domain-specific structures. As a result, the relationship between input and solution tends to be smoother and more regular in that case. Studying economic dispatch thus provides a more realistic and challenging testbed for evaluating learning-based optimization methods like PDL.

Moreover, when PDL is integrated with Benders decomposition to solve the generation expansion planning problem, the resulting subproblems correspond precisely to instances of economic dispatch. Therefore, any extensions developed in this chapter are directly applicable to the later study involving Benders decomposition. The aim of this chapter is to assess the effectiveness of PDL beyond previously studied problems and to explore improvements that support its use in this context.

The dataset generation process is described in Section 5.1. Results for the plain PDL method as introduced in Section 2.6 are presented and discussed in Section 5.2. Section 5.3 introduces repair and completion layers to simplify the learning process and evaluates the performance of PDL with these layers. A more advanced variant with additional repair mechanisms is proposed and evaluated in Section 5.4. The overall findings are discussed in Section 5.5, and the chapter is concluded in Section 5.6.

5.1. Experimental Setup

We consider a simple, time-independent variant of the economic dispatch problem (3.6), with three nodes, three transmission lines, and six generator types, two per node (Figure 5.1). For each node, one of these generator types is a renewable energy source with a varying supply, whereas the other is a fossil-based generator with a constant maximum supply. The three nodes correspond to Belgium, France, and Germany. Belgium can invest in offshore wind generators and gas generators. France can invest in solar energy generators and nuclear energy generators. Germany can invest in solar energy generators and gas generators. Each node is connected directly to the other nodes through a transmission line.

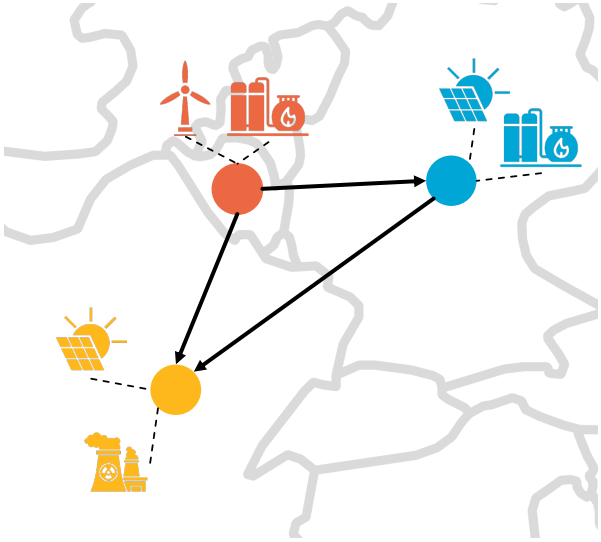

Figure 5.1: Overview of the specific variant of the economic dispatch problem considered: each node represents a country with its generator technologies, and transmission lines fully connect all nodes. Arrows denote the inherent direction of each transmission line, although power may flow in both directions.

5.1.1. Data Generation

Data samples follow the following structure. The vector of demand $\mathbf{D}$ and the vector containing generation availability $\mathbf{A}$ vary across instances. The profiles for these parameters are identical to those used by Arnoldus [5] and can be found in the code linked in Appendix A.

Additionally, since we are considering an economic dispatch problem, the vector containing the units invested in each generator, $\mathbf{u}$, is known. If we solve the economic dispatch problem as a subproblem of Benders decomposition, the investment decisions would be computed based on the cuts from the Benders algorithm. To mimic this interaction, we generate a set of investment decisions $\mathbf{u}$, which vary across instances as described below. Additionally, Benders decomposition would allow the time dimension to be fully decomposed, and therefore each instance considers only a single timestep t. As such, the equations in this chapter are no longer indexed by t.

Each data sample is therefore characterized by the vector of demand $\mathbf{D}$, the vector of available generation $\mathbf{A}$, and the vector of units invested $\mathbf{u}$. However, $\mathbf{A}$ and $\mathbf{u}$ are both present in the upper bound of the generator production, in the right-hand side of Constraint (3.6c):

$$\overline{P}_g = A_g P_g^{\max} u_g. \tag{5.1}$$

The entire right-hand-side term of that constraint is included as input to the model, as passing the separate parts could only complicate learning because the network would have to learn the interaction between inputs.

The dataset $\mathscr{D}$ containing S data samples $\mathbf{z}$ therefore takes the following form:

$$\mathscr{D} = \left\{ \mathbf{z}^{(i)} = \left(\{D_n\}_{n \in \mathscr{N}}, \{\overline{P}_g\}_{g \in \mathscr{G}} \right) \right\}_{i=1}^{S}, \tag{5.2}$$

where each i indicates a distinct sample.

Generation of Investment Decisions

The investment decisions selected during Benders decomposition will often correspond to extreme points because optimal solutions are generally known to lie at the extreme points of the feasible region. For example, in the absence of any cuts, the minimal-cost solution is to make no investments at all, resulting in $\mathbf{u} = \mathbf{0}$.

To ensure that the trained network is well-prepared for these scenarios, the investment decision vectors of the dataset must provide good coverage of the feasible decision space. If, for example, samples are drawn from a normal distribution, the resulting vectors will be concentrated near the origin, which fails to capture the boundary regions.

In order to adequately represent the boundary regions, the investment decision vectors are sampled from a Sobol sequence [54]. A Sobol sequence produces quasi-random samples that more uniformly span a space, especially in high-dimensional settings, when compared to purely random sampling. Note that these sequences work best when sample sizes are powers of two.

The experiments in the remainder of this chapter are performed on a dataset with 2^{15} samples, with a training, validation, and test-set split of 80%, 10%, and 10%, respectively.

5.1.2. Neural Network Architecture and Hyperparameters

The neural network architecture used for both the primal and dual networks remains a multi-layer perceptron. The number of layers is increased in accordance with the second PDL publication by Park and Van Hentenryck [46]. Additionally, following that work, a LayerNorm layer [6] is added before the multi-layer perceptron in the primal network only. Although no justification is provided in the original paper, we find that this layer is necessary to prevent vanishing gradients in computations involving sigmoids, which will be introduced later in Section 5.3. To ensure consistency, LayerNorm is included in all experiments presented here, including those without sigmoids.

Full details on the network architecture, the chosen hyperparameters, and their motivation are provided in Appendix C.3.

Hyperparameters are based on the work by Park and Van Hentenryck [47, 46], and therefore not tuned to any specific problem in this chapter. Hyperparameter tuning is performed later to improve performance prior to combining PDL with Benders decomposition in Chapter 7. Extensive hyperparameter tuning is not necessary to highlight the differences in performance between the PDL configurations introduced in this chapter and is therefore omitted.

An overview of the used codebase [26] and the link to the repository are provided in Appendix A.

5.1.3. Objective Function

To eliminate the incentive for negative generation production and negative unmet demand, the objective function used in the loss function is slightly modified compared to the original objective function in the economic dispatch problem (3.6). In the original formulation, both generation production and unmet demand are multiplied by their respective costs. As a result, if the neural network outputs negative values for these variables, the corresponding cost terms also become negative, leading to an undesired minimizing of the loss function. Although the Lagrangian and penalty terms in PDL are designed to drive the decision variables toward feasibility, we have chosen to address the issue directly at the source. Otherwise, additional conflicting gradients are introduced through both a reward on negative generation and unmet demand in the objective function, and a penalty on negative generation and unmet demand in the loss. Such conflicting gradients are known to complicate the learning process [60]. To resolve this, the objective function is modified to penalize the absolute values of generation production and unmet demand:

$$\sum_{g \in \mathcal{G}, t \in \mathcal{T}} C_g^{\text{var}} \cdot |p_{g,t}| + \sum_{n \in \mathcal{N}, t \in \mathcal{T}} C^{\text{e}} \cdot |e_n|. \tag{5.3}$$

Note that this does not change the value of the optimal objective function. Any negative values for generation production $p_{g,t}$ and unmet demand e_n are still corrected through the Lagrangian and penalty terms of PDL, but the objective function no longer produces conflicting gradients.

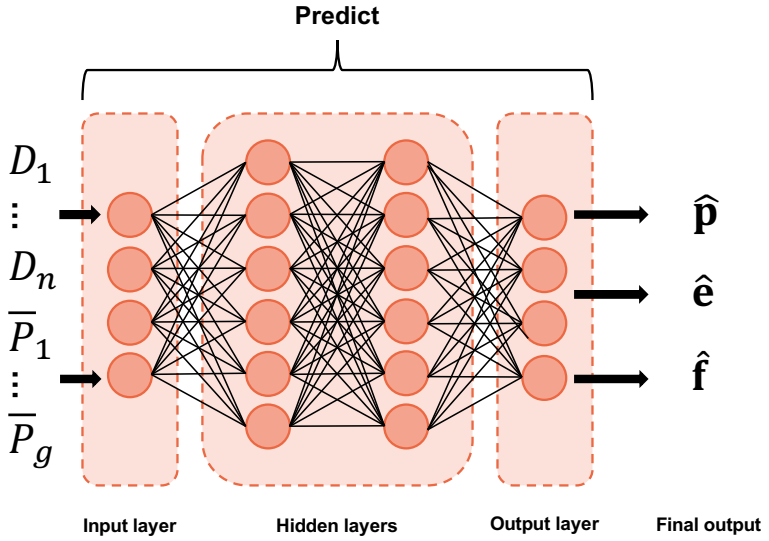

Figure 5.2: Overview of the plain primal network for the economic dispatch problem. The inputs are the demands D and generator capacities $\overline{P}$. Outputs are the predicted decision variables: generator production $\hat{\mathbf{p}}$, transmission line flows $\hat{\mathbf{f}}$, and unmet demand $\hat{\mathbf{e}}$.

5.2. Plain Primal-Dual Learning

This section applies *plain* PDL to the economic dispatch problem, without any extensions, exactly as described in Section 2.6. Given an input $\mathbf{z}$, the primal network predicts all of the decision variables:

$$P_\phi(\mathbf{z}) = (\hat{\mathbf{p}}, \hat{\mathbf{f}}, \hat{\mathbf{e}}). \tag{5.4}$$

An overview of the primal network is given in Figure 5.2.

5.2.1. Results

The performance of plain PDL on the test set is presented in Tables 5.1 and 5.2.

Table 5.1: Performance of plain PDL on the test set of economic dispatch instances in terms of objective value. The known optimal objective value, predicted objective value, and resulting optimality gap (in %) are reported. Results are averaged over five repeats to reduce the effect of randomness; standard deviations are also reported.

Experiment	Optimal obj.	Predicted obj.	Optimality gap (%)
Plain PDL	20 128.1	510 353.4 ± 3340.0	31 362.2 ± 241.5

Table 5.2: Performance of plain PDL on the test set of economic dispatch instances in terms of feasibility. The average maximum and mean violations across all test samples are reported for the inequality and equality constraints. Results are averaged over five repeats to reduce the effect of randomness; standard deviations are also reported.

Experiment	Ineq. max	Ineq. mean	Eq. max	Eq. mean
Plain PDL	653.7 ± 14.1	42.0 ± 1.2	5153.1 ± 31.9	3067.8 ± 17.3

It is clear from Table 5.1 that plain PDL fails to approach optimal solutions. At the same time, Table 5.2 shows that the constraint violations are large, especially for the equality constraints, which correspond to the node balancing constraints 3.6h.

5.2.2. Discussion of Results

Due to the quadratic nature of the penalty term in the loss function (2.28), violations of the node balancing constraint outweigh all other components, including the objective value and the remaining con-

straint violations. As a result, the neural network prioritizes the node balancing constraint while largely ignoring the others. Nevertheless, it fails to reduce the violations of this constraint.

This highlights the challenge of satisfying the node balancing constraints. Although inequality constraints are generally easier to satisfy than equality constraints, as they must be satisfied exactly at the bound, an additional explanation can be given for the difficulty observed here. The node balancing constraint involves all of the predicted decision variables:

$$D_n = \sum_{g \in \mathcal{G}_n} \hat{p}_g + \sum_{l \in \mathcal{L}_n^{\text{in}}} \hat{f}_l - \sum_{l \in \mathcal{L}_n^{\text{out}}} \hat{f}_l \quad + \hat{e}_n, \tag{5.5}$$

whereas the other constraints involve only a single predicted decision variable. The difficulty arises from the need to balance multiple predicted variables simultaneously, where a change in one variable requires a corresponding change in one or more of the others to maintain feasibility. As a result, the gradient updates for these connected variables may conflict, as the gradient for a single variable may improve its own contribution to the balance, but when combined with updates to other variables, it may worsen the overall constraint satisfaction. This makes it difficult for the neural network to learn a feasible solution effectively, highlighting the need for additional methods beyond penalization to enforce constraint satisfaction.

5.3. Primal-Dual Learning with Bound Repair Layers and Completion Layers

As shown in the previous section, even when using PDL, which accounts explicitly for constraint violations in the loss, predicting near-optimal decision variables that satisfy feasibility constraints remains challenging. Improving constraint satisfaction during learning will not only result in predictions closer to the feasible region but also improve the objective value, because violations in the loss function no longer dominate it.

This can be readily achieved by constraining or transforming the output of the neural network to values that are known to be feasible. However, to learn in an end-to-end manner, these operations must be differentiable, allowing for the gradient-based optimization required for neural networks. This is generally non-trivial, as many constraint satisfaction or projection operations are inherently non-differentiable or involve complex computations. A benefit is that these mechanisms reduce the need for post-hoc repair or correction steps, which decouple the learning objective from the final output.

Prior research has proposed various approaches for differentiable constraint satisfaction within neural networks to overcome this challenge. These approaches can broadly be categorized into two classes: implicitly differentiable and explicitly differentiable methods.

Implicitly differentiable methods, introduced in the related work in Section 1.2, address constraint satisfaction by embedding a solver for the constrained optimization problem directly into the network architecture, allowing differentiation through the implicit function theorem. Although these approaches can be effective in certain applications, they are not considered in this thesis due to their high computational cost, as previously discussed.

In contrast, explicitly differentiable methods rely on repair functions that are differentiable by design and can be directly integrated into the computational graph. These methods typically offer lower computational cost, enabling faster training and inference. For this reason, this thesis focuses exclusively on explicitly differentiable repair mechanisms for enforcing feasibility in neural network outputs.

The remainder of this section introduces differentiable repair and completion layers, which enforce constraint satisfaction in different ways. These repair and completion layers have largely been based on the work by Van Hentenryck [57].

5.3.1. Bound Repair Layer

Repair layers ensure feasibility by projecting the unconstrained neural network outputs onto the feasible region. In this work, bound repair layers are employed to enforce variable bounds of the form

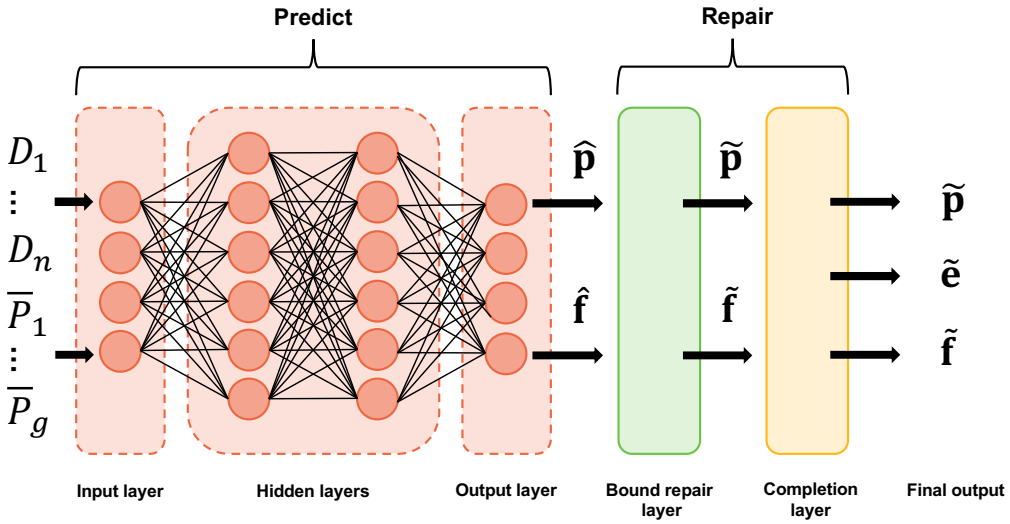

Figure 5.3: Overview of the primal network with bound repair and completion layers for the economic dispatch problem. The inputs remain the same. The neural network only outputs the generator production $\hat{\mathbf{p}}$ and transmission line flows $\hat{\mathbf{f}}$. These are bound-repaired using a sigmoid layer. The unmet demand $\tilde{\mathbf{e}}$ is computed from the repaired $\tilde{\mathbf{p}}$ and $\tilde{\mathbf{f}}$ in the completion layer.

$\underline{\mathbf{x}} \leq \mathbf{x} \leq \overline{\mathbf{x}}$, where $\mathbf{x}$ are decision variables. A common approach is to use a sigmoid activation function, which naturally maps inputs to the interval $(0,1)$. The resulting output can then be rescaled to the original bounds via the following transformation:

$$\tilde{\mathbf{x}} = \underline{\mathbf{x}} + (\overline{\mathbf{x}} - \underline{\mathbf{x}}) \cdot \sigma(\hat{\mathbf{x}}), \tag{5.6}$$

where $\sigma(\cdot)$ denotes the sigmoid function, $\hat{\mathbf{x}}$ is the raw network output, $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ are respectively the lower and upper bound, and $\tilde{\mathbf{x}}$ denotes the repaired prediction. This formulation guarantees that the final output strictly adheres to the specified bounds in a smooth and differentiable manner. However, sigmoid layers can introduce vanishing gradients when their inputs are near the bounds [23]. Therefore, it is important to ensure that the input to the sigmoid does not take on excessively large positive or negative values, where the gradient of the sigmoid becomes insignificant. This is why the earlier-mentioned LayerNorm is essential: it normalizes the input to have zero mean and unit variance, helping stabilize activations and mitigate vanishing gradients.

Similar repair layers using a sigmoid are employed by Park and Van Hentenryck [46]. In the present work, these bound repair layers are applied to Constraints (3.6b)–(3.6e) of the economic dispatch problem. The constraints on unmet demand (3.6f) and (3.6g), are not bound repaired.

5.3.2. Node Balance Completion Layer

Differentiable completion layers are adjacent to repair layers. These completion layers operate by predicting a subset of the decision variables and subsequently deriving the remaining variables through a closed-form expression that ensures constraint satisfaction. In addition to reducing the dimensionality of the neural network's output and therefore reducing the size of the search space, completion layers inherently enforce constraints by design.

Consider an equality constraint of the form

$$x_1 + x_2 = C, \tag{5.7}$$

where x_1 and x_2 are decision variables, and C is a known constant. In this case, the network can be designed to predict either x_1 or x_2, while the other variable is computed deterministically:

$$x_1 = C - x_2, \quad \text{or} \quad x_2 = C - x_1. \tag{5.8}$$

This formulation guarantees that the constraint is always satisfied, regardless of the prediction. While this example illustrates a simple case where the closed-form solution is trivial to compute, in general,

the derived expressions may be arbitrarily complex, as long as the closed-form expression remains differentiable.

A completion layer is useful for the node balancing constraint (3.6h). Specifically, instead of predicting all decision variables $\mathbf{p}, \mathbf{f}, \mathbf{e}$, the network output can be limited to the production vector $\mathbf{p}$ and line flow vector $\mathbf{f}$, and then unmet demand for a specific node e_n can be inferred from them as follows:

$$e_n = D_n - \left(\sum_{g \in \mathcal{G}_n} p_g + \sum_{l \in \mathcal{L}_n^{\text{in}}} f_l - \sum_{l \in \mathcal{L}_n^{\text{out}}} f_l \right). \tag{5.9}$$

This enforces the node balancing constraint, removing the need for the neural network to coordinate multiple outputs to achieve feasibility. Note, however, that feasibility is not ensured for the unmet demand e_n, since it can take on negative values if

$$D_n < \sum_{g \in \mathcal{G}_n} p_g + \sum_{l \in \mathcal{L}_n^{\text{in}}} f_l - \sum_{l \in \mathcal{L}_n^{\text{out}}} f_l, \tag{5.10}$$

and thus violates the bounds on unmet demand (3.6f). These bounds cannot be repaired after completion using a bound repair layer, as the equality constraints would no longer hold.

An overview of the primal network architecture with added bound repair and completion layers is given in Figure 5.3.

5.3.3. Results

The performance of the primal network setup with the added repair and completion layers (5.6), (5.9) on the test set is compared to the performance of plain PDL in Tables 5.3 and 5.4.

In Table 5.3, it can be seen that the optimality gaps are reduced, but remain large. Table 5.4 shows that the repair and completion layers perform as intended. Due to the completion layer, the equality constraints are fully satisfied, with no remaining violations. The inequality constraint violations are also reduced, with only violations on the unmet demand bounds still present.

Table 5.3: Performance of PDL with repair and completion layers (bold) on the test set of economic dispatch instances in terms of objective value. The known optimal objective value, predicted objective value, and resulting optimality gap (in %) are reported. Results are averaged over five repeats to reduce the effect of randomness; standard deviations are also reported.

Experiment	Optimal obj.	Predicted obj.	Optimality gap (%)
Plain PDL	20 128.1	510 353.4 $\pm$ 3340.0	31 362.2 $\pm$ 241.5
Repair Completion PDL	**20 128.1**	**71 529.5 $\pm$ 3706.9**	**3464.0 $\pm$ 253.4**

Table 5.4: Performance of PDL with repair and completion layers (bold) on the test set of economic dispatch instances in terms of feasibility. The average maximum and mean violations across all test samples are reported for the inequality and equality constraints. Results are averaged over five repeats to reduce the effect of randomness; standard deviations are also reported.

Experiment	Ineq. max	Ineq. mean	Eq. max	Eq. mean
Plain PDL	653.7 $\pm$ 14.1	43.0 $\pm$ 1.2	5153.1 $\pm$ 31.9	3067.8 $\pm$ 17.3
Repair Completion PDL	**39.8 $\pm$ 8.5**	**1.8 $\pm$ 0.4**	**0.0**	**0.0**

5.3.4. Discussion of Results

The large optimality gaps are caused by the model's inability to effectively minimize the amount of unmet demand, because such significant optimality gaps cannot be explained by using more expensive generators alone. The cost of unmet demand is heavily penalized and dominates the total cost.

The difficulty in minimizing unmet demand is likely still linked to the node balancing constraints. Although the completion layer ensures that these constraints are satisfied without requiring direct coordination among all predicted variables, the model still needs to coordinate several variables to reduce unmet demand. This remains difficult for the reasons discussed earlier in Section 5.2.

5.4. Primal-Dual Learning with a Generation-Prioritized Layer

To further address the difficulty of minimizing unmet demand, as discussed in the previous section, this work introduces a new repair layer that explicitly adjusts the predicted generator production. The main contribution of this layer is its built-in preference for satisfying demand through generation rather than with the unmet demand slack variable. This way, the neural network does not have to learn how to meet demand from scratch. Instead, the model can concentrate on learning how to satisfy demand in a cost-efficient manner.

5.4.1. Generation-Prioritized Layer

The generation-prioritized layer proposed in this work builds upon a node balance repair mechanism introduced in earlier studies [57, 46] for different problem domains than those considered in this thesis. That prior mechanism was developed for simpler node balances without flows and unmet demand:

$$\sum_{g \in \mathcal{G}_n} p_g = D_n. \tag{5.11}$$

This node balance is repaired by proportionally scaling the production of the generators to meet the demand. If the total production exceeds the demand, the production is scaled down. If demand exceeds total production, production is scaled up. The scale factor is computed in such a way that the production bounds of the generators are respected. The repair is defined as

$$\tilde{p}_g = \begin{cases} (1 - \zeta_n^{\uparrow})\hat{p}_g + \zeta_n^{\uparrow}\overline{P}_g & \text{if } \sum_{g \in \mathcal{G}_n} \hat{p}_g < D_n, \\ (1 - \zeta_n^{\downarrow})\hat{p}_g + \zeta_n^{\downarrow}\underline{P}_g & \text{otherwise,} \end{cases} \quad \forall g \in \mathcal{G}_n, \tag{5.12}$$

where

$$\zeta_n^{\uparrow} = \frac{D_n - \sum_{g \in \mathcal{G}_n} \hat{p}_g}{\sum_{g \in \mathcal{G}_n} \overline{P}_g - \sum_{g \in \mathcal{G}_n} \hat{p}_g}, \qquad \text{and} \qquad \zeta_n^{\downarrow} = \frac{\sum_{g \in \mathcal{G}_n} \hat{p}_g - D_n}{\sum_{g \in \mathcal{G}_n} \hat{p}_g - \sum_{g \in \mathcal{G}_n} \underline{P}_g}.$$

Here, $\underline{P}_g$ and $\overline{P}_g$ denote the lower and upper bounds of the production of a generator. This repair is not applicable to the economic dispatch problem (3.6), because its node balancing constraint (3.6h) takes on another form where there are additional terms for the inflows f_n^{in}, the outflows f_n^{out}, and a slack variable for the unmet demand e_n. It should be noted that e_n is necessary to allow otherwise infeasible solutions, as insufficient investment decisions from Benders decomposition can result in the total production capacity being unable to cover the demand fully.

Furthermore, any repair involving the flows is non-trivial, as altering a single flow impacts the balance of other nodes. Consequently, flows would require a global repair approach, which is complex in itself, considering that solving flow problems represents a distinct class of optimization problems.

In the remainder of this section, we propose modifications to the node balance repair layer to use it to prioritize generation over unmet demand, without interfering with the flows.

If we disregard altering the flows, the effective demand of a node can be calculated as

$$D_n^{\text{eff}} = D_n - \sum_{l \in \mathcal{L}_n^{\text{in}}} f_l + \sum_{l \in \mathcal{L}_n^{\text{out}}} f_l, \tag{5.13}$$

so that the node balancing constraint becomes

$$D_n^{\text{eff}} = \sum_{g \in \mathcal{G}_n} p_g \quad + e_n. \tag{5.14}$$

Modeling the unmet demand e as an expensive generator to use the node balance repair will not work, because, as a slack variable, it must be allowed to take on negative values.

However, since the penalty on unmet demand is higher than the variable costs of the generators, we know that the unmet demand should be minimized. We can add this information to the repair by rewriting the node balance as

$$\min\left(D_n^{\text{eff}}, \sum_{g\in\mathcal{G}_n} \overline{P}_g\right) = \sum_{g\in\mathcal{G}_n} p_g, \tag{5.15}$$

so that the generators cannot overshoot their maximum capacity, and then estimating the unmet demand slack variable through a completion layer. The key insight is that $e_n = 0$ as long as $\sum_{g\in\mathcal{G}_n} \overline{P}_g \geq D_n^{\text{eff}}$.

However, one edge case remains when the imported flows outweigh the demand of a node, when

$$D_n < \sum_{l\in\mathcal{L}_n^{\text{in}}} f_l - \sum_{l\in\mathcal{L}_n^{\text{out}}} f_l, \tag{5.16}$$

resulting in $D_n^{\text{eff}} < 0$, breaking the repair layer. Therefore, the effective demand is clamped from below by zero.

The generation-prioritized layer for the economic dispatch problem is thus specified as

$$\tilde{p}_g = \begin{cases} (1-\zeta_n^{\uparrow})\hat{p}_g + \zeta_n^{\uparrow}\overline{P}_g & \text{if } \sum_{g\in\mathcal{G}_n} \hat{p}_g < \min\left((D_n^{\text{eff}})^+, \sum_{g\in\mathcal{G}_n} \overline{P}_g\right), \\ (1-\zeta_n^{\downarrow})\hat{p}_g + \zeta_n^{\downarrow}\underline{P}_g & \text{otherwise,} \end{cases} \quad \forall g \in \mathcal{G}_n, \tag{5.17}$$

where the corresponding scale factors are defined as

$$\zeta_n^{\uparrow} = \frac{\min\left((D_n^{\text{eff}})^+, \sum_{g\in\mathcal{G}_n} \overline{P}_g\right) - \sum_{g\in\mathcal{G}_n} \hat{p}_g}{\sum_{g\in\mathcal{G}_n} \overline{P}_g - \sum_{g\in\mathcal{G}_n} \hat{p}_g}, \qquad \text{and} \qquad \zeta_n^{\downarrow} = \frac{\sum_{g\in\mathcal{G}_n} \hat{p}_g - \min\left((D_n^{\text{eff}})^+, \sum_{g\in\mathcal{G}_n} \underline{P}_g\right)}{\sum_{g\in\mathcal{G}_n} \hat{p}_g - \sum_{g\in\mathcal{G}_n} \underline{P}_g}.$$

The same completion layer from the previous section is then used to estimate the unmet demand slack variable e_n.

Note that the generator production should be bound-repaired beforehand, as any production outside of the bounds will not be corrected by the node balancing repair layer. For example, if $\hat{p}_g > \overline{P}_g$, then

$$(1-\zeta_n^{\uparrow})\hat{p}_g + \zeta_n^{\uparrow}\overline{P}_g > \overline{P}_g, \quad \forall\, \zeta_n^{\uparrow} \in (0,1), \tag{5.18}$$

and the generator production does not satisfy the upper bound. An overview of the primal network architecture with this generation-prioritized layer is given in Figure 5.4.

5.4.2. Results

The performance of PDL with the generation-prioritized layer is compared to the two other setups from the prior sections in Tables 5.5–5.7.

This version of PDL outperforms the other setups, achieving optimality gaps within 2%, without any constraint violations on the test set.

Table 5.7 reports two types of metrics averaged over the test set. First, it shows the mean absolute difference between the predicted and known optimal decision variables. For reporting line flows, the net flow per node is used instead of individual line flows, since multiple flow patterns can result in the same net flow. Second, it shows the mean percentage of demand that cheaper generators could have met without changing flows.

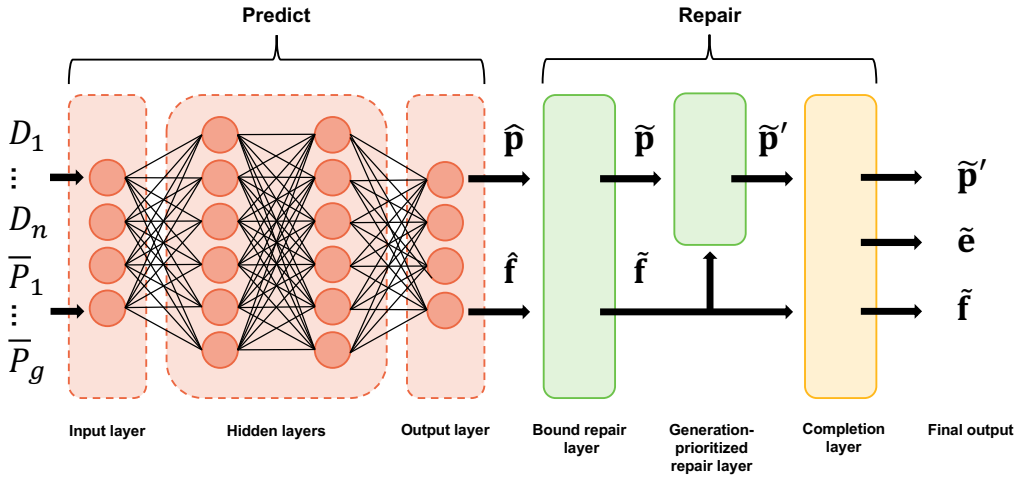

Figure 5.4: Overview of the primal network with bound repair and completion layers, and also the generation-prioritized layer for the economic dispatch problem. The inputs remain the same. Between the bound repair layer and the completion layer, the repaired generator production $\tilde{\mathbf{p}}$ is adjusted to prioritize unmet demand, resulting in $\tilde{\mathbf{p}}'$.

Table 5.5: Performance of PDL with the generation-prioritized layer (bold) on the test set of economic dispatch instances in terms of objective value. The known optimal objective value, predicted objective value, and resulting optimality gap (in %) are reported. Results are averaged over five repeats to reduce the effect of randomness; standard deviations are also reported.

Experiment	Optimal obj.	Predicted obj.	Optimality gap (%)
Plain PDL	20 128.1	510 353.4 ± 3340.0	31 362.2 ± 241.5
Repair Completion PDL	20 128.1	71 529.5 ± 3706.9	3464.0 ± 253.4
Generation-Prioritized PDL	**20 128.1**	**20 166.6 ± 7.7**	**1.9 ± 0.3**

5.4.3. Discussion of Results

The results confirm that the generation-prioritized layer functions as intended. All constraints are satisfied, allowing the primal network to focus on learning cost-efficient generation strategies. Table 5.7 shows that the neural network successfully favors cheaper generators over more expensive ones, as 0% of the effective demand per node can be generated more cheaply. Furthermore, because generation is prioritized over unmet demand by design, any remaining optimality gap can be attributed to a suboptimal effective demand D^{eff} per node, which results from differences in the predicted net flows.

The predicted flows affect the objective value in two main ways. First, they can lead to unnecessary unmet demand, for example, when a node with unused capacity is unable to support another node that is already at full capacity. Second, flows can be inefficient when a node fails to import cheaper energy from another node, resulting in higher overall generation costs.

The difficulty of learning flows can be attributed to several factors. First, flows introduce symmetry, as multiple flow configurations can lead to the same net flow at each node. This creates ambiguity, making it harder for the neural network to learn consistent patterns. Additional symmetry arises when multiple generators have the same exact marginal cost; if two nodes can supply energy to a third node at the same cost, it does not matter which one is used, resulting in equally valid but different solutions. Symmetries are known to introduce saddle points and plateaus in the optimization landscape, thus complicating learning. Furthermore, predicting flows requires coordinated, global adjustments across the network, increasing the complexity of the learning task compared to local, node-wise decisions such as generation or unmet demand.

5.5. Discussion

The results presented in this chapter demonstrate that plain PDL struggles to learn the economic dispatch problem effectively. The underlying difficulty is likely caused by the need to coordinate multiple variable predictions to satisfy constraints and to minimize the objective function. Although direct evidence for this hypothesis is unavailable, a series of modifications were proposed based on this intu-

Table 5.6: Performance of PDL with the generation-prioritized layer (bold) on the test set of economic dispatch instances in terms of feasibility. The average maximum and mean violations across all test samples are reported for the inequality and equality constraints. Results are averaged over five repeats to reduce the effect of randomness; standard deviations are also reported, but standard deviations of 0 are omitted for clarity.

Experiment	Ineq. max	Ineq. mean	Eq. max	Eq. mean
Plain PDL	653.7 ± 14.1	43.0 ± 1.2	5153.1 ± 31.9	3067.8 ± 17.3
Repair Completion PDL	39.8 ± 8.5	1.8 ± 0.4	0.0	0.0
Generation-Prioritized PDL	**0.0**	**0.0**	**0.0**	**0.0**

Table 5.7: The absolute difference between the known optimal decision variables and the predictions from PDL with the generation-prioritized layer. The cheaper available generation reports the percentage of power generation that a cheaper generator could have generated within the same node. Results are averaged over five repeats to reduce the effect of randomness; standard deviations are also reported.

	Absolute differences			
Experiment	Production	Unmet demand	Net flow	Cheaper available generation (%)
Generation-Prioritized PDL	135.925 ± 42.247	9.051 ± 0.435	280.900 ± 84.461	0.000

ition. Adding bound repair layers, completion layers, and the generation-prioritized repair resulted in expected improvements, supporting the hypothesis.

By incorporating all proposed repair mechanisms, the primal network produced fully feasible solutions with average optimality gaps under 2%. This improvement can largely be attributed to the ability of the repair layers to enforce feasibility directly, thus reducing the need to balance the conflicting objectives of minimizing the objective value and satisfying the constraints. However, when all constraints are guaranteed to be satisfied by construction, the Lagrangian and penalty terms of the PDL loss become zero. In such settings, the training objective is reduced to the standard minimization of the primal objective function, and the use of PDL may no longer be justified. Alternative machine learning approaches, which decouple the primal and dual networks, may be more suitable in such cases.

Despite the promising results, the studied economic dispatch variant remains a simplified proof of concept. It contains only three nodes, two generator types per node, and three connecting transmission lines. Realistic energy systems typically involve dozens of nodes, lines, and generators, leading to a much larger number of decision variables and a significantly more complex problem. It is expected that the implementations explored in this chapter will encounter new challenges in such settings, particularly where repair layers cannot be applied as easily.

The second PDL publication by Park and Van Hentenryck [46] reports optimality gaps below 2% on significantly larger problems, with input feature dimensions up to $|\mathbf{z}| = 5000$. In contrast, the economic dispatch variant studied in this chapter has an input feature dimension of only $|\mathbf{z}| = 9$. It is unlikely that the implementation presented in this chapter would directly scale to problems of that size while maintaining the same level of performance. However, a key difference is that their problem setup only predicts generator dispatch and does not require learning line flows, as these are given. This difference is believed to be the primary reason for the discrepancy in scalability. Section 5.4 shows that generation can be predicted near-optimally, and the remaining optimality gap is attributed entirely to inaccuracies in the predicted flows, confirming that line flows form a bottleneck in this setting. Addressing the difficulty in learning line flows presents a promising avenue for future research.

The experiments in this chapter used a fixed set of hyperparameters, selected to match prior work rather than optimized for the economic dispatch problem. While sufficient to demonstrate relative improvements between configurations, further tuning of both hyperparameters and network architecture could lead to better overall performance. Additionally, alternative neural architectures may offer improvements. In particular, graph neural networks offer a promising direction. Graph neural networks can leverage the natural graph topology of power networks, or even model the structure of linear programs directly, treating variables and constraints as nodes in a graph. Since graph neural networks naturally aggregate global information, they are likely a good fit for learning the line flows.

Finally, it is worth noting that this chapter evaluates performance using the mean optimality gap, which

computes the relative difference between predicted and optimal objective values. This relative measure aligns well with Benders decomposition, where consistent performance across all investment decisions is required, regardless of the magnitude of the resulting optimal objective. However, the loss function used during training minimizes the mean predicted objective, which places greater emphasis on instances with higher objective values than those with lower values. This can result in less accurate predictions as Benders decomposition approaches the true minimum. Alternative loss functions that align more closely with the mean optimality gap may improve performance and should be explored in future work.

5.6. Conclusion

This chapter researched the application of PDL to the economic dispatch problem and explored a series of differentiable repair strategies to address feasibility and optimality challenges. Each modification was designed to reduce the challenge of coordinating multiple decision variables by the neural network by enforcing constraints directly through the neural network architecture. The final setup, which combined a bound repair layer, completion layer, and generation-prioritized layer, yielded feasible solutions with optimality gaps of 2%.

While the results demonstrate that feasibility can be effectively enforced through differentiable architectural components, the network's ability to produce meaningful dual variables remains unaddressed. This motivates the next chapter, which focuses on improving the quality and usability of learned dual variables.

6

Evaluating and Extending Dual Learning

As explained in Section 2.6, PDL aims to train two neural networks: one for predicting primal solutions and another for predicting dual solutions. Although accurately predicting only the primal solutions might suffice in some contexts, much of the utility of PDL lies in producing a consistent primal-dual pair. This pairing not only enables us to assess the optimality of a predicted solution via the gap between the primal and dual objectives through strong duality (Theorem 2), but also enables integration with techniques such as Benders decomposition, which rely on both the primal objective value and the dual solutions.

However, the studies by Park and Van Hentenryck [47, 46] evaluate the performance of PDL exclusively in terms of the quality of the predicted primal solutions. In their approach, the dual predictions are used mainly to facilitate smoother learning of the primal variables. The quality of the dual predictions is not directly assessed.

This chapter aims to address this shortcoming by evaluating the quality of the solutions learned by the dual network and by proposing modifications that improve the quality of these learned solutions.

Section 6.1 evaluates the quality of the dual predictions by computing the dual optimality gap and dual constraint violations on both the quadratic programming benchmark and the economic dispatch problem. Section 6.2 proposes an alternative loss function for training the dual network and introduces a dual completion layer to improve feasibility. In Section 6.3, a dual classification layer is proposed, incorporating prior knowledge about dual variables. Section 6.4 details the experimental setup, and Section 6.5 subsequently presents the results. Finally, Section 6.6 discusses the findings, and Section 6.7 concludes the chapter.

6.1. Quality of Dual Variable Predictions

This section assesses the quality of the dual solutions learned by the dual networks that were trained alongside the primal networks investigated in previous chapters.

To perform this assessment, the dual problem formulations must first be derived for both the convex quadratic programming benchmark (3.1) and the economic dispatch problem (3.6). The dual formulation of economic dispatch corresponds to the earlier formulation from Section 2.4. The dual formulation for the convex quadratic programming problem differs, however, as it includes a quadratic term. The dual formulation for the convex quadratic programming problem is as follows:

$$\underset{\boldsymbol{\mu},\boldsymbol{\lambda}}{\text{maximize}} \quad -\frac{1}{2}(\mathbf{r}+\mathbf{G}^\top\boldsymbol{\mu}+\mathbf{H}^\top\boldsymbol{\lambda})^\top\mathbf{Q}^{-1}(\mathbf{r}+\mathbf{G}^\top\boldsymbol{\mu}+\mathbf{H}^\top\boldsymbol{\lambda})-\boldsymbol{\mu}^\top\mathbf{b}-\boldsymbol{\lambda}^\top\mathbf{d} \tag{6.1a}$$

$$\text{subject to} \quad -\boldsymbol{\mu}\leq 0. \tag{6.1b}$$

The complete derivation of the dual problem formulation for the convex quadratic programming benchmark is provided in Appendix D.

The results in this section are obtained from the dual networks that were trained alongside the primal networks presented in Chapters 4 and 5. No additional networks are trained for this section. The reader is referred to the respective earlier chapters for details on the experimental setup.

6.1.1. Results

Quality is measured using the dual optimality gap with respect to the known optimal dual solution and absolute constraint violations. For the convex quadratic programming benchmark, plain PDL is evaluated on the original, unmodified problem. For the economic dispatch, three setups are considered: plain PDL (Section 5.2), PDL with bound repair and completion layers (Section 5.3), and PDL with the generation-prioritized layer (Section 5.4), because the quality of the learned duals depends on the quality of the corresponding primal solutions. Table 6.1 presents the quality of the dual solutions for both the convex quadratic programming and economic dispatch problems.

Table 6.1: Evaluation of the quality of dual predictions using plain PDL on the convex quadratic programming benchmark and the three PDL setups from Chapter 5 on the economic dispatch problem. The convex QP dual (6.1) contains no equality constraints; therefore, those entries are omitted. A negative optimality gap indicates that the predicted optimal value is lower than the known optimal value. Results are averaged over five repeats to reduce the effect of randomness; standard deviations are also reported. Results (mean ± standard deviation) × magnitude are averaged over five repeats to reduce randomness; standard deviations of zero and unit-scale factors are omitted for clarity.

Metric		Convex QP	Economic dispatch		
		Plain	Plain	Bound & compl.	Generation-prioritized
Objective	Predicted	$(-7.8 \pm 6.2) \times 10^{3}$	$(8.3 \pm 0.1) \times 10^{11}$	3.4 ± 6.6	-1.0 ± 2.6
	Gap (%)	$(-5.2 \pm 4.2) \times 10^{4}$	3.0×10^{10}	-10.0×10^{1}	-1.0×10^{2}
Inequality	Max	$(2.3 \pm 0.6) \times 10^{-2}$	$(2.4 \pm 0.2) \times 10^{5}$	$(0.5 \pm 1.0) \times 10^{-3}$	$(4.2 \pm 8.3) \times 10^{-4}$
	Mean	$(2.6 \pm 0.6) \times 10^{-3}$	$(1.6 \pm 0.2) \times 10^{4}$	$(0.5 \pm 1.1) \times 10^{-4}$	$(1.7 \pm 3.5) \times 10^{-5}$
Equality	Max	–	1.1×10^{7}	1.0×10^{1}	1.0×10^{1}
	Mean	–	$(6.4 \pm 0.1) \times 10^{6}$	2.5	2.5

6.1.2. Discussion of Results

The results show several issues with the predictions learned by the dual networks. The predicted dual objective values differ by a few orders of magnitude from the known optimal values, as indicated by the large mean optimality gaps. This is particularly notable in the convex quadratic programming benchmark, where PDL has been verified to function as intended (Appendix B). Since the primal network performs well on this problem, the poor quality of the corresponding dual predictions highlights a fundamental weakness in PDL's ability to learn accurate dual variables. In the economic dispatch problem, the dual objective predicted by plain PDL exceeds the optimal value, which, according to Theorem 1, indicates dual infeasibility. This is further supported by the observed constraint violations.

Violations of the inequality constraints indicate that the predicted dual variables associated with these constraints, $\hat{\boldsymbol{\mu}}$, can take on negative values. This leads to undesirable behavior in the primal loss function 2.28, where the Lagrangian term for an inequality constraint becomes negative whenever a dual variable is negative and the corresponding constraint is violated. Although the targets for $\boldsymbol{\mu}$ are clamped to zero from below in the dual loss function 2.29, the dual network still produces negative predictions for the predicted inequality-constraint dual variables $\hat{\boldsymbol{\mu}}$.

While the Augmented Lagrangian Method (ALM) enforces the KKT conditions directly, this is no longer the case when a neural network approximates the dual variables. Therefore, it would be reasonable to restrict the Lagrangian terms in the PDL loss to nonnegative values. This would prevent the loss

from being minimized in unintended ways, such as by reducing the Lagrangian terms through negative duals. The low constraint violations observed in the convex quadratic programming benchmark suggest that this issue becomes less significant when the primal predictions are nearly optimal.

Moreover, the economic dispatch setups with repair layers produce dual objective values close to zero, with dual optimality gaps of nearly 100%. This occurs because the targets for the dual decision variables in the dual loss function (2.29) remain nearly zero, since the dual target updates adopted from the ALM, as specified in 2.5, depend directly on constraint violations. When a constraint exhibits no violations, its corresponding dual variable receives no signal during training and therefore cannot be meaningfully learned.

This highlights a fundamental challenge in learning dual solutions using PDL with repair layers. Although the repair and completion layers have been shown in Chapter 5 to be essential to achieve near-optimal primal predictions, these same layers prevent the dual network from learning meaningful dual solutions through PDL. To use PDL with Benders decomposition, where both accurate primal and dual predictions are required, modifications to the algorithm are necessary to ensure that the dual variables are learned correctly.

6.2. Dual Learning with Completion Layer and Objective-Based Loss Function

This section proposes an alternative approach for the dual network that enables dual learning in the presence of primal repair layers.

The dual targets present in the original dual loss function (2.29) originate from the ALM, of which the theory relies on the presence of constraint violations [22, 48, 53]. As demonstrated in Chapter 5, repair mechanisms are essential for primal learning on the economic dispatch problem, resulting in constraints for which there are never violations. Therefore, loss functions relying on these update rules cannot be applied in such repaired systems without fundamentally limiting either primal or dual learning.

Given that the dual problem itself is an optimization problem, an alternative approach is to train the dual network by directly optimizing the dual objective function, similar to the training of the primal network. This avoids the need for explicit target values and allows the integration of dual repair and completion layers to enforce feasibility during training. As noted by Michel [39], the ALM performs better when the dual variables μ, λ are close to their optimal values $\mu^\star, \lambda^\star$, supporting the idea of using directly optimized duals to penalize the primal network effectively.

6.2.1. Dual Completion Layer

Direct learning of the dual variables enables dual repair and completion layers. A dual completion mechanism has been introduced by Klamkin, Tanneau, and Van Hentenryck [29] that applies to primal problems in which the inequality constraints are box constraints that bound the decision variables from below and above. This completion layer computes a dual-feasible solution for the box constraints, μ, given the dual variables associated with the equality constraints λ. Later, the completion layer has been generalized to linear and nonlinear conic optimization problems by Tanneau and Van Hentenryck [55]. It has been tested on optimal power flow problems [29], as well as multidimensional knapsack and nonlinear production and inventory planning problems [55], but not yet on economic dispatch problems.

In the economic dispatch problem, the inequality constraints correspond to box constraints, and thus the completion layer by Klamkin, Tanneau, and Van Hentenryck [29] can be used in this setting. As a result, the dual network needs to predict only the equality-constraint dual variables λ, and feasibility is ensured by construction, enabling the advantages discussed in Chapter 5.

The details of applying the repair layer from the work by Klamkin, Tanneau, and Van Hentenryck [29] to the economic dispatch problem are shown below.

Since the inequality constraints are box constraints, where entries of the constraint matrix are –1 for lower bounds, and 1 for upper bounds, the inequality constraint matrix $\mathbf{G}$ can be restructured as

$$\mathbf{G}^\top \boldsymbol{\mu} = \boldsymbol{\mu}_u - \boldsymbol{\mu}_l, \tag{6.2}$$

where $\boldsymbol{\mu}_u$ and $\boldsymbol{\mu}_l$ respectively are the dual variables corresponding to upper and lower bounds. Similarly, the right-hand side of the inequality constraints $\mathbf{b}$ can be restructured as

$$\mathbf{b}^\top \boldsymbol{\mu} = \mathbf{b}_l^\top \boldsymbol{\mu}_l + \mathbf{b}_u^\top \boldsymbol{\mu}_u, \tag{6.3}$$

where $\mathbf{b}_l$ and $\mathbf{b}_u$ respectively are the parts of the right-hand side of the inequality constraints corresponding to lower and upper bounds.

The dual problem formulation 2.16 can then be restructured as

$$\begin{aligned}
\underset{\boldsymbol{\mu}, \boldsymbol{\lambda}}{\text{maximize}} \quad & -\mathbf{b}_l^\top \boldsymbol{\mu}_l - \mathbf{b}_u^\top \boldsymbol{\mu}_u - \mathbf{d}^\top \boldsymbol{\lambda} && \text{(6.4a)} \\
\text{subject to} \quad & \boldsymbol{\mu}_u - \boldsymbol{\mu}_l + \mathbf{H}^\top \boldsymbol{\lambda} = -\mathbf{c}, && \text{(6.4b)} \\
& \boldsymbol{\mu}_l, \boldsymbol{\mu}_u \geq 0. && \text{(6.4c)}
\end{aligned}$$

When the duals corresponding to the equality constraints $\boldsymbol{\lambda}$ are given, they can be removed from the objective value, resulting in:

$$\begin{aligned}
\underset{\boldsymbol{\mu}}{\text{maximize}} \quad & -\mathbf{b}_l^\top \boldsymbol{\mu}_l - \mathbf{b}_u^\top \boldsymbol{\mu}_u && \text{(6.5a)} \\
\text{subject to} \quad & \boldsymbol{\mu}_u - \boldsymbol{\mu}_l = -\mathbf{c} - \mathbf{H}^\top \boldsymbol{\lambda}, && \text{(6.5b)} \\
& \boldsymbol{\mu}_l, \boldsymbol{\mu}_u \geq 0. && \text{(6.5c)}
\end{aligned}$$

Because the difference between the positive and negative parts of a function is equal to the original function, that is, $f = f^+ - f^-$, a closed-form solution is available for $\boldsymbol{\mu}_l$ and $\boldsymbol{\mu}_u$:

$$\boldsymbol{\mu}_l^\star = (\mathbf{c} + \mathbf{H}^\top \boldsymbol{\lambda})^+, \qquad \text{and} \qquad \boldsymbol{\mu}_u^\star = (\mathbf{c} + \mathbf{H}^\top \boldsymbol{\lambda})^-. \tag{6.6}$$

The architecture of the dual network when integrating this completion layer is presented in Figure 6.1.

6.2.2. Loss Function

Since the dual completion layer ensures feasibility, dual constraint violations do not need to be penalized, and the dual loss can be defined directly as the negated dual objective:

$$\ell_D(\boldsymbol{\mu}, \boldsymbol{\lambda}) = -f_D(\boldsymbol{\mu}, \boldsymbol{\lambda}). \tag{6.7}$$

The negation appears because the dual objective is maximized, whereas the loss function is minimized during training.

In cases where such a completion layer cannot be applied to ensure feasibility, the dual loss could be extended to the dual augmented Lagrangian. In this formulation, the primal decision variables act as the Lagrange multipliers, penalizing dual constraint violations. However, this case is not considered in this thesis, as the dual completion layer introduced in this section applies to a broad class of convex optimization problems that can be expressed in conic form, including both linear and nonlinear cases. Thus, dual constraint violations can be eliminated by construction for such problems.

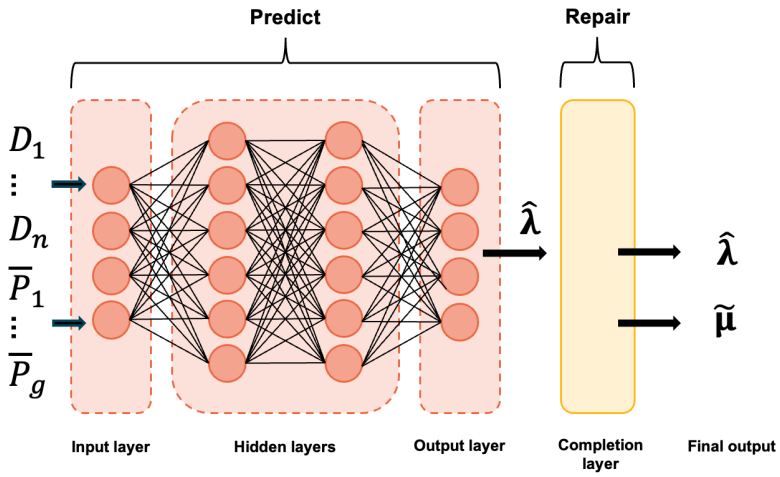

Figure 6.1: Overview of the dual network with the completion layer for the economic dispatch problem. The inputs remain the same as for the primal networks. Outputs of the neural network are the predicted dual equality decision variables $\hat{\boldsymbol{\lambda}}$. Using a dual completion layer, the dual inequality decision variables $\tilde{\boldsymbol{\mu}}$ are computed, resulting in a feasible dual solution.

6.3. Dual Learning with the Classification Layer

In this section, we propose a novel extension to dual learning for the economic dispatch problem, leveraging prior knowledge on the dual variables through a classification layer.

The dual variables are known to correspond to *shadow prices*[1] [10]. Thus, the optimal dual variable corresponding to the node balancing constraint is equal to the marginal cost of supplying one additional unit of demand. In the economic dispatch variant considered in this thesis, this means that the marginal cost will be either the cost of the cheapest available generator or, if all generators are at maximum capacity, the cost of unmet demand. Therefore, the optimal dual variable must be any of the generator costs or the cost of unmet demand, and any values in between are known to be suboptimal. Embedding this prior information into the learning process can lead to a substantial improvement.

Using the dual completion layer, the dual network output consists of the dual variables corresponding to the node balancing constraint $\boldsymbol{\lambda}$, and the other dual variables $\boldsymbol{\mu}$ are completed from them. Since $\boldsymbol{\lambda}$ can be categorized into predefined classes, the dual network is reformulated as a *classification*[2] model rather than a *regression*[3] model. In this formulation, the network outputs *logits*[4] that represent the probabilities of the dual variable belonging to a specific category, in this case, a specific shadow price. These logits are then transformed into actual probabilities using a *softmax*[5] layer:

$$\text{softmax}\left(D_\psi(\mathbf{z})\right) = \hat{\boldsymbol{\pi}}^{\boldsymbol{\lambda}}, \tag{6.8}$$

where $\hat{\boldsymbol{\pi}}^{\boldsymbol{\lambda}} \in \mathbb{R}^{M \times K}$ is the predicted categorical probability distribution over K classes for the O dual variables corresponding to the equality constraints.

The expected values of these dual variables are then computed, effectively computing a mix of classes, as

$$\mathbb{E}[\hat{\boldsymbol{\lambda}}] = \sum_{k=1}^{K} \hat{\pi}_k^{\boldsymbol{\lambda}} \cdot v_k, \tag{6.9}$$

[1] Shadow prices represent the marginal gain in the objective from relaxing a constraint by one unit.
[2] Classification refers to predicting discrete categories
[3] Regression refers to predicting continuous values
[4] Logits are the unnormalized outputs of a neural network layer.
[5] Softmax is a function that converts logits into a probability distribution over predefined categories.

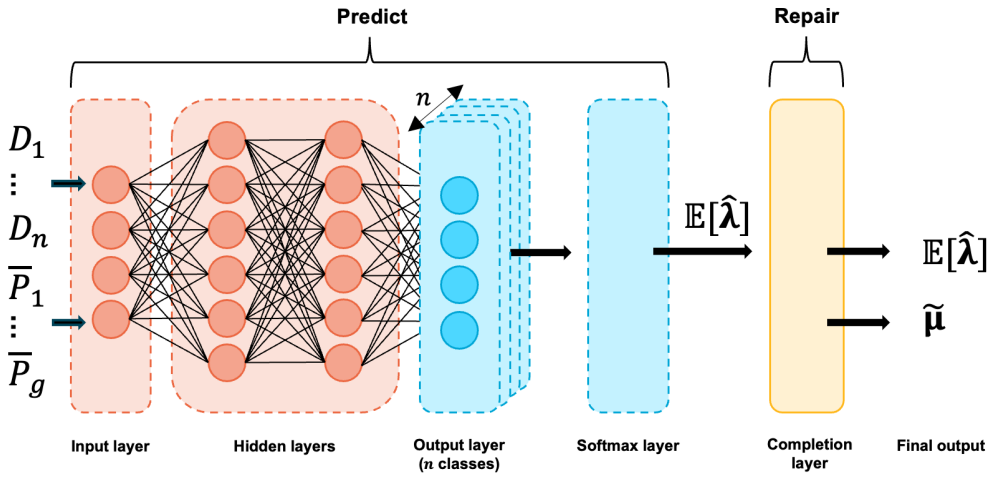

Figure 6.2: Overview of the dual network with a classification layer for the economic dispatch problem. The inputs are identical to those of the primal network. The final layer outputs logits for each of the n classes, which are converted to probabilities using a softmax. The expected value of the dual equality variable $\mathbb{E}[\hat{\lambda}]$ is then computed over these probabilities. The dual inequality decision variables $\tilde{\mu}$ are then completed from this expected value. During inference, the class with the highest probability is used instead of the expected value.

where v_k denotes the dual value associated with class k. These class values are the unique variable costs of generation and the cost of unmet demand. The expected value of the duals of the inequality constraints is then used to complete the dual variables corresponding to the inequality constraints. During inference, however, the *argmax*[6] is applied to select the dual value with the highest predicted probability, rather than computing the expectation. This cannot be done during training, as the argmax operation is not differentiable and would prevent gradient-based optimization.

The architecture of the dual network when incorporating classification is shown in Figure 6.2.

6.4. Experimental Setup

The two dual learning setups introduced in the previous sections were trained and evaluated on the same dataset of economic dispatch instances introduced in Chapter 5. As the predicted dual solutions are ensured to be feasible through the completion layer, there are no constraint violations that require penalization. Therefore, dual learning can be decoupled from primal learning, as the primal network does not influence dual learning. As a result, only a dual network is trained in the experiments presented in Section 6.5.

Training hyperparameter and model architecture tuning are not performed, for the reasons stated in Section 5.1. The used hyperparameters and model architecture are detailed in Appendix C.4.

An overview of the codebase [26] and the link to the repository are provided in Appendix A.

6.5. Results

Table 6.2 compares the test set performance of dual learning using the completion layer against the setup that includes an additional classification layer. Due to the use of the completion layer, both dual networks yield feasible solutions; hence, constraint violations are not reported. The results are averaged over five independent runs.

The results in Table 6.2 indicate that the mean dual optimality gap is smaller for the setup incorporating the dual classification layer compared to the one using only the completion layer. However, the standard deviation for the classification setup is significantly larger.

Table 6.3 presents the dual optimality gap of each individual run. It can be observed that the dual optimality gap for the classification setup includes a large outlier in the fifth run. When this outlier

[6]The argmax function returns the index of the largest element in a list.

Table 6.2: Evaluation of the quality of dual predictions using the dual completion and dual classification setups on the economic dispatch problem test set. The predicted dual objective values and corresponding optimality gaps are reported. A negative optimality gap indicates that the predicted objective is lower than the known optimal value. Results are averaged over five repeats to reduce the effect of randomness; standard deviations are also reported.

Experiment	Predicted obj.	Optimality gap (%)
Dual Completion	28 373.8 ± 77.1	−56.3 ± 4.4
Dual Classification	28 058.7 ± 1520.5	−20.7 ± 29.8

is excluded, the mean dual optimality gap with the classification layer is approximately an order of magnitude smaller than without it.

Table 6.3: Detailed evaluation of the dual optimality gaps achieved by the dual network with the dual completion layer and the classification layer, reported per distinct run on the economic dispatch problem test set. Values closer to 0 are preferred.

Experiment	Run 1 (%)	Run 2 (%)	Run 3 (%)	Run 4 (%)	Run 5 (%)
Dual Completion	-56.212	-59.473	-52.627	-62.603	-50.557
Dual Classification	-3.896	-10.457	-3.858	-5.314	-80.085

6.5.1. Discussion of Results

The results show that the classification layer achieves significantly lower dual optimality gaps than the setup using only completion in most runs, demonstrating the potential of incorporating prior knowledge about the structure of the dual variables. However, the approach also shows instability; in some instances, the dual network converges to suboptimal solutions, becoming trapped in local minima. Improving the stability of the classification-based method, therefore, remains an important direction for future research.

This instability can likely be attributed, in part, to a mismatch between the training loss and the true optimality gap. As in primal learning, minimizing the mean absolute objective value does not directly correspond to minimizing the relative optimality gap. In the classification setup, an additional mismatch arises from the difference between training and inference procedures. During training, the loss is computed using the expected value of the dual variable, calculated as a weighted sum over class probabilities from the softmax. During inference, however, the class with the highest predicted probability is selected. As a result, a model may achieve a low training loss while still producing suboptimal predictions at inference time. This mismatch can be problematic in settings such as Benders decomposition, as training favors larger dual objective values, while the decomposition process requires convergence toward smaller ones.

While embedding prior knowledge has shown clear benefits on the economic dispatch problem considered in this thesis, its applicability depends on the problem at hand. For example, extending the economic dispatch problem to include ramping constraints or storage would significantly complicate the marginal price structure, limiting the effectiveness of a direct classification strategy. Nonetheless, these findings offer a promising proof of concept for incorporating prior knowledge in dual learning and motivate further work to address its limitations.

6.6. Discussion

The dual network trained alongside the primal network in plain PDL has been shown to produce poor and, at times, infeasible dual predictions: a limitation that has so far been overlooked. However, the ALM is known to converge to the optimal dual variables under certain conditions [39]. A likely explanation for this discrepancy is that the theoretical convergence guarantees of the ALM do not hold when dual target updates depend on the inexact outputs of neural networks. Moreover, the PDL framework is primarily designed to produce high-quality primal solutions. Consequently, hyperparameters, such as the penalty weight, that strongly influence the behavior of the dual variables are typically tuned to support primal learning rather than to produce accurate duals. Additionally, the original loss function for the dual network measures only how well it approximates its targets, rather than directly optimizing any metric indicative of dual quality.

The alternative loss function proposed in this chapter directly optimizes the dual objective, enabling the integration of a dual completion layer that enforces feasibility by design and significantly improves dual prediction quality compared to the original target-matching loss, whose performance is presented in Table 6.1. Additionally, a dual classification layer that incorporates prior information about the optimal duals is capable of further reducing the dual optimality gap to within 4%. These results demonstrate the potential of direct dual objective optimization as a promising direction for improving dual learning.

While the dual learning approach proposed in this chapter shows promising results, it does not address scenarios in which the dual variables cannot be repaired to feasibility. Given the poor performance observed even with the use of the dual completion layer, evaluating setups without completion would be challenging. As such, investigating the penalization of dual constraint violations via the primal decision variables remains an open research direction. Nonetheless, the dual completion layer employed in this thesis is applicable to a wide class of convex optimization problems that can be formulated in conic form, including both linear and nonlinear cases. Extending the work in this chapter to problems involving nonlinear constraints or discrete decision variables will require further research.

The proposed loss function in this work is based solely on the dual objective value. However, more sophisticated loss functions may improve performance. For example, Klamkin, Tanneau, and Van Hentenryck [29] utilize a regularization term that smooths the optimization landscape, offering a promising avenue for improving training stability and convergence. Regularization terms could also help reduce the mismatch between the loss and the optimality gap by encouraging solutions that generalize better across instances and align more closely with the true optimal dual solutions, rather than fitting to local variations in the objective landscape.

Dual learning may also benefit from incorporating information derived from primal predictions, using duality theory. For instance, the presence of unmet demand in a primal solution indicates that the corresponding shadow price, which is the dual variable associated with the node balance constraint, should be equal to the cost of unmet demand. However, leveraging primal predictions in this way requires care, as they are often inexact. For example, even if the true value of unmet demand is zero, the primal network will typically be unable to predict an unmet demand of exactly zero.

Furthermore, assessing the impact of the alternatively learned duals on the quality of primal predictions is an important direction for future work. A natural starting point is the quadratic programming benchmark, where the performance of standard PDL is already well established.

As observed in primal learning, line flows substantially complicate the economic dispatch problem. As they influence the marginal cost of generation, these effects are expected to carry over to dual learning. Addressing the challenges introduced by line flows is likely to improve the quality of learned dual solutions.

Finally, the dual network used in this research is a simple multi-layer perceptron. As such, the architectural considerations discussed in Chapter 5 for primal learning are also applicable to dual learning.

Despite the limitations mentioned in this section, the proposed dual learning approach offers a practical and extensible framework for learning dual variables in constrained optimization. The improvements observed using the alternative loss, completion, and classification layers indicate that meaningful progress can be made by tailoring the dual learning objective to the problem structure. This sets the stage for further research into integrating duality principles more deeply into learning-based optimization and expanding the approach to broader problem classes.

6.7. Conclusion

This chapter highlights a key limitation of the PDL framework: its limited ability to produce accurate dual predictions. Experiments on the convex quadratic programming benchmark demonstrate that even when the primal network performs well, the corresponding dual predictions remain poor. Moreover, when employing repair layers, shown to be essential for effective primal learning on the economic dispatch problem, learning duals using plain PDL becomes impossible due to the absence of constraint violations.

We address this limitation with an alternative approach to dual learning based on directly maximiz-

ing the dual objective function. When combined with a completion layer that guarantees feasibility, this approach still faces challenges in learning accurate dual variables. To improve performance, we propose a novel classification-based method that leverages prior knowledge about the structure of the dual variables. Although such prior information may not be available in all problem settings, the results provide a promising proof of concept for incorporating domain knowledge into dual learning. Additionally, we identify that the proposed classification layer can make the learning process less stable. Further research is needed to address this issue.

7

Integrating Primal-Dual Solution Pairs into Benders Decomposition

As stated in Chapter 1, traditional constrained optimization methods face computational limitations when applied to energy systems planning. Integrating learned primal–dual solution pairs into Benders decomposition presents a promising approach to addressing these limitations, due to the inherent synergy between machine learning and the many similar subproblems encountered iteratively. By leveraging primal and dual networks previously developed in Chapters 5 and 6, predictions for the economic dispatch subproblem introduced in Section 3.3 can be made, facilitating the computation of Benders cuts used to solve the generation expansion planning problem described in Section 3.2.

Several learning-enhanced approaches have been introduced in the related work (see Section 1.2.3 for more details). This chapter builds on related ideas but introduces several key differences.

First, the method is applied to a new domain, generation expansion planning, where the problem structure and feasibility constraints differ significantly. In this setting, feasibility cannot be ensured by the master problem alone and must be verified within the subproblem, which was not necessary in prior work. Ensuring feasibility is crucial for producing approximate solutions with bounds on suboptimality.

Second, whereas earlier work is able to generate near-optimal approximate solutions using Benders, valid primal-dual solution pairs and thus theoretical bounds on suboptimality are lacking. In contrast, the proposed approach uses learned feasible primal-dual subproblem solution pairs, enabling the generation of approximate Benders cuts that include suboptimality bounds derived from duality. This allows the approximate solutions to be both computationally efficient and theoretically grounded.

Third, the presented framework supports a hybrid strategy: learning-based methods can be used to quickly generate approximate solutions with suboptimality bounds, while exact solving remains available when the predicted quality is insufficient, such as in cases involving out-of-distribution inputs.

Section 7.1 first presents an overview of the proposed learning-enhanced Benders decomposition framework, including proven theoretical guarantees. Next, Section 7.2 details the experimental setup, and Section 7.3 presents and discusses the results of the proposed approach. A further investigation into the results is performed and discussed in Section 7.4. Section 7.5 provides a discussion of the findings throughout the chapter, and finally, Section 7.6 concludes the chapter.

7.1. Primal-Dual Learning-Enhanced Benders Decomposition

This section details how predicted primal-dual solution pairs are integrated into the Benders decomposition framework to solve the generation expansion planning problem introduced in Section 3.2. We first describe the structure of the master and subproblems, highlighting how decomposition in the time dimension enables efficient subproblem solving and the use of predicted dual variables in Benders cuts.

Subsequently, the theoretical foundation for the framework is established, including a proof of the validity of the approximated Benders cuts, which ensures valid duality gaps and guarantees convergence to the optimal solution after switching from predictions to an exact solver for the subproblems. Finally, pseudo-code detailing the proposed algorithm is presented.

7.1.1. Master and Subproblem of Generation Expansion Planning

Following the theory introduced in Section 2.7, Benders decomposition is applied to the generation expansion planning problem (3.5). In this context, the investment decision variables u_g are the complicating variables, while the production $p_{g,t}$, transmission flows $f_{l,t}$, and unmet demand e are the non-complicating variables. The problem formulations in this section are based on those presented in Chapter 3, but have been adapted to use vector notation, as the non-vector form would result in equations that are excessively large and difficult to read.

Although the notation in Section 2.7 assumes a single subproblem, in the generation expansion planning problem, each timestep is operationally independent. Therefore, the economic dispatch subproblem can be solved independently for each timestep. This property enables parallelization and simplifies integration with the learning models. The solutions to these per-timestep subproblems are then aggregated to form the Benders cut in the master problem.

The master problem can thus be formulated as follows, following (2.39):

$$\underset{\mathbf{u},\,\alpha}{\text{minimize}} \quad (\mathbf{C}^{\text{inv}} \circ \mathbf{P}^{\max})^\top \mathbf{u} + W \cdot \alpha \tag{7.1a}$$

$$\text{subject to} \quad \sum_{t \in \mathscr{T}} \left[\left(\boldsymbol{\mu}_t^{(k)}\right)^\top (\mathbf{b}_t - \mathbf{G}_t^{\text{u}} \mathbf{u}) + \left(\boldsymbol{\lambda}_t^{(k)}\right)^\top \mathbf{d}_t \right] \leq \alpha \quad k = 1, \dots, \nu, \tag{7.1b}$$

$$\mathbf{u} \in \mathbb{N}, \tag{7.1c}$$

where $\mathbf{v}_1 \circ \mathbf{v}_2$ denotes the element-wise multiplication of some vectors $\mathbf{v}_1$ and $\mathbf{v}_2$. The objective consists of the investment costs plus the auxiliary variable α representing the objective value of the subproblems weighted by W. The ν cuts (7.1b) consist of the dual variables $\boldsymbol{\mu}_t^{(k)}, \boldsymbol{\lambda}_t^{(k)}$, predicted for a specific investment decision $\mathbf{u}^{(k)}$, the contribution of the investment decisions to the inequality constraints $\mathbf{G}_t^{\text{u}}\mathbf{u}$, and the right-hand-sides of the inequality and equality constraints, respectively $\mathbf{b}_t$ and $\mathbf{d}_t$. The term $\mathbf{H}^{\text{u}}\mathbf{u}$, which would represent the contribution of the investment decisions to the equality constraints, is omitted from the cuts because it evaluates to zero in this case due to independence of equality constraints on the variables $\mathbf{u}$.

The subproblem corresponds to the economic dispatch problem, and is solved independently for each timestep $t \in \mathscr{T}$:

$$\underset{\mathbf{x} = [\mathbf{p}^\top, \mathbf{f}^\top, \mathbf{e}^\top]^\top}{\text{minimize}} \quad (\mathbf{C}^{\text{var}})^\top \mathbf{p} + (\mathbf{C}^{\text{e}})^\top \mathbf{e} \tag{7.2a}$$

$$\text{subject to} \quad \mathbf{G}^{\text{x}}\mathbf{x} \leq \mathbf{b}_t - \mathbf{G}_t^{\text{u}}\mathbf{u} \quad | \quad \boldsymbol{\mu}_t, \tag{7.2b}$$

$$\mathbf{H}^{\text{x}}\mathbf{x} = \mathbf{d}_t \quad | \quad \boldsymbol{\lambda}_t. \tag{7.2c}$$

Here, only the elements that vary over time are indexed by timestep t. Additionally, $\mathbf{G}^{\text{x}}$ is the left-hand side, while $\mathbf{b}_t - \mathbf{G}_t^{\text{u}}\mathbf{u}$ is the right-hand side of Constraints (3.6b)–(3.6g). Similarly, $\mathbf{H}^{\text{x}}$ is the left-hand side and $\mathbf{d}_t$ is the right-hand side of Constraint (3.6h). Again, the equality constraint term involving the investment decisions $\mathbf{H}^{\text{u}}\mathbf{u}$ has been omitted from the right-hand side, as these constraints do not depend on the investment variable $\mathbf{u}$. Each timestep results in distinct dual variables $\boldsymbol{\mu}_t, \boldsymbol{\lambda}_t$.

7.1.2. Theoretical Guarantees

Integrating learned primal-dual solution pairs into Benders decomposition changes its theoretical foundation. Below are identified the properties of classical Benders decomposition that are preserved, those that are no longer guaranteed, and under what conditions these changes occur.

Validity of Predicted Cuts

In the context of Benders decomposition with approximate subproblem solutions, the validity of the generated cuts is essential to ensure that the master problem yields sound bounds on the true optimal value. A cut is considered valid if it serves as a global underestimator of the true value function $\theta(\mathbf{u})$. Invalid cuts, which overestimate the value function at some point, can result in unsound bounds on the approximate solution to the master problem. Additionally, such cuts can incorrectly eliminate regions containing the true optimum and prevent convergence when falling back on exact optimization.

Infeasible dual solutions do not satisfy the conditions of weak duality (Theorem 1). As a result, their associated dual objective values may exceed the true optimal value, potentially producing invalid cuts.

In contrast, feasible dual solutions satisfy weak duality and thus cannot overestimate the value function locally at the point of evaluation. However, an affine function that underestimates the value function at a single point may still violate global validity if its slope causes it to intersect the value function elsewhere. The following theorem establishes that such a slope, and thus, such a violation cannot occur for cuts derived from feasible dual solutions:

Theorem 3 (Validity of Feasible Cuts)**.** *Given a Benders decomposed optimization problem where weak duality holds in the subproblem, and a feasible dual solution to the subproblem* $(\boldsymbol{\mu}^{(k)}, \boldsymbol{\lambda}^{(k)})$ *at a given instance of the complicating variables* $\mathbf{u}^{(k)}$*. Then, for all* $\mathbf{u} \in \mathbb{R}^n$*, the corresponding Benders cut is a valid underestimator of the true subproblem value function* θ*, that is:*

$$\theta(\mathbf{u}) \geq \left(\boldsymbol{\mu}^{(k)}\right)^\top (\mathbf{b} - \mathbf{G}^u \mathbf{u}) + \left(\boldsymbol{\lambda}^{(k)}\right)^\top (\mathbf{d} - \mathbf{H}^u \mathbf{u}) \,. \tag{7.3}$$

Proof outline. Let the subproblem be parameterized by the complicating variables $\mathbf{u}$. Let $(\boldsymbol{\mu}^{(k)}, \boldsymbol{\lambda}^{(k)})$ be a feasible dual solution at the point $\mathbf{u}^{(k)}$. By weak duality, the dual objective value associated with any feasible dual solution provides a lower bound on the primal optimal value. In Benders decomposition, the complicating variables $\mathbf{u}$ appear only in the right-hand side of the primal constraints. As a result, the dual constraints are independent of $\mathbf{u}$. Consequently, the same dual solution remains feasible for all $\mathbf{u} \in \mathbb{R}^n$. Substituting $\mathbf{u}$ into the dual objective yields an affine function that globally underestimates $\theta(\mathbf{u})$. Therefore, the resulting Benders cut is valid globally. □

Validity of the Duality Gap

By Theorem 3, a set of valid Benders cuts forms a global underestimator of the true subproblem value function θ. From weak duality (Theorem 1), the subproblem objective value evaluated at a feasible primal solution $\mathbf{x}$ for a fixed complication variable vector $\mathbf{u}^{(k)}$ serves as a valid overestimator of the true value function θ at that point. Together, these bounds imply that a value function approximation constructed from feasible dual cuts, in combination with a feasible primal solution, defines a valid duality gap around the true subproblem objective value, and by extension, around the true optimal value of the original Benders-decomposed problem.

This establishes that feasible primal-dual solution pairs can be consistently integrated into Benders decomposition to produce valid bounds on optimality.

Convergence under Approximations

Each cut in the set of cuts forming the predicted value function $\hat{\theta}$ is valid. However, some cuts may be dominated, lying entirely below or coinciding with the current value function approximation. If in a given iteration the cut computed at investment decision $\mathbf{u}^{(\nu)}$ in that iteration is dominated by the predicted value function $\hat{\theta}$, adding that cut does not update the predicted value function. Therefore, the same investment decision $\mathbf{u}^{(\nu)}$ is selected in the next iteration, causing the algorithm to stall indefinitely. Thus, the Benders algorithm is not guaranteed to converge when using solely predicted primal-dual solution pairs.

Convergence under Exact Refinement

A set of valid Benders cuts forms a global underestimator of the true subproblem value function θ by Theorem 3. Therefore, the algorithm may, at any iteration, transition from using predicted feasible dual solutions to computing exact duals—a process we refer to as *exact refinement*. Since the predicted cuts

do not overestimate the value function, all new exact cuts will never underestimate the predicted cuts, and all classical convergence guarantees of Benders decomposition remain intact upon switching.

7.1.3. Algorithm and Convergence Criteria

During Benders iterations, the dual network predicts the feasible dual solutions to the economic dispatch subproblems, using the modifications described in Chapter 6. These dual predictions, $(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}})$, are used to construct valid Benders cuts, as shown in (7.1). The primal predictions are used to compute the upper bound, as it is tracked throughout classical Benders iterations. However, since both the lower and upper bounds are derived from approximations, it is no longer guaranteed that they will coincide in the limit. Consequently, a different convergence criterion is required.

Rather than monitoring the gap between the upper and lower bounds, we propose determining convergence based on the stagnation of the lower bound, specifically when it fails to improve by more than a pre-specified threshold ϵ. As mentioned earlier, the dual network may generate dominated cuts, which result in repeated selection of the same investment decision. Therefore, the algorithm may stagnate before the value function has been approximated with sufficient accuracy. In principle, this could be mitigated by actively selecting a new investment decision to promote further exploration. To that extent, a patience parameter p can be introduced, allowing the algorithm to continue for p additional iterations after the last improvement before termination. However, identifying new investment decisions that are likely to improve the bound is non-trivial and, as such, left for future work.

Once convergence is detected, the algorithm proceeds in one of two ways: it either terminates, returning the latest feasible primal-dual solution pair—thereby forming an upper and lower bound on the optimal objective value—or it transitions to an exact refinement phase to solve the problem to optimality.

The pseudocode for the proposed algorithm is presented in Algorithm 2.

Algorithm 2 Inexact Benders Decomposition with Learned Duals

Initialize iteration index $k \leftarrow 0$
Initialize cut set $\mathscr{C} \leftarrow \varnothing$
Initialize guess $\mathbf{u}^{(0)}$
Initialize approximation value $\alpha^{(0)} \leftarrow -\infty$
Initialize tolerance $\epsilon \geq 0$
repeat
    Predict feasible duals $(\hat{\boldsymbol{\mu}}^{(k)}, \hat{\boldsymbol{\lambda}}^{(k)})$ using dual network
    Add Benders cut to $\mathscr{C}$:
        $(\hat{\boldsymbol{\mu}}^{(k)})^\top(\mathbf{b} - \mathbf{G}^u\mathbf{u}) + (\hat{\boldsymbol{\lambda}}^{(k)})^\top(\mathbf{d} - \mathbf{H}^u\mathbf{u}) \leq \alpha$
    Solve master problem with cuts $\mathscr{C}$ to obtain:
        New decision $\mathbf{u}^{(k+1)}$
        Updated approximation value $\alpha^{(k+1)}$
    Predict feasible primal solution $(\hat{p}, \hat{f}, \hat{u})^{(k+1)}$ using primal network P_ϕ
    Evaluate subproblem objective at $(\hat{p}, \hat{f}, \hat{u})^{(k+1)}$ to compute $\theta(\mathbf{u}^{(k+1)})$
    if $\alpha^{(k+1)} - \alpha^{(k)} \leq \epsilon$ **then**
        if Exact refinement is enabled **then**
            Switch to exact Benders (Algorithm 1), carrying over $\mathscr{C}$
        else
            break
        end if
    end if
    $k \leftarrow k + 1$
until converged
return $\mathbf{u}^{(k)}, \alpha^{(k)}, \theta(\mathbf{u}^{(k)})$

Based on Theorem 3, Algorithm 2 produces valid Benders cuts using the feasible dual predictions from the dual network, guiding the Benders iterations. Upon convergence, the algorithm falls back on classical Benders if exact refinement is enabled, or otherwise outputs a feasible primal-dual solution pair.

7.2. Experimental Setup

This section evaluates the effectiveness of learning-enhanced Benders decomposition for solving the generation expansion planning problem introduced in Chapter 3. Three variants are compared: *exact Benders*, *inexact Benders*, and *inexact Benders with exact refinement*.

- *Exact Benders* is the classical Benders decomposition without any learning-based enhancements.
- *Inexact Benders* uses predicted primal-dual subproblem solution pairs obtained solely from the trained primal and dual neural networks.
- *Inexact Benders with exact refinement* begins as inexact Benders but switches to exact subproblem solutions once progress stagnates, thereby ensuring convergence to the optimal solution.

Benders decomposition is integrated with the primal network, incorporating bound repair, completion, and generation-prioritized layers, as detailed in Chapter 5. Additionally, the dual network incorporates completion and classification layers described in Chapter 6.

To ensure valid primal objective values during Benders decomposition, the objective function uses absolute values for the generation and unmet demand terms, as defined in (5.3). This design guarantees that the predicted primal objective value will never underestimate the true optimal value, even on challenging samples that might otherwise yield negative unmet demand values.

Since the dual network produces feasible predictions and any unmet demand is penalized in the primal objective, the training of the primal and dual networks is decoupled; constraint violations do not require additional penalization.

7.2.1. Implementation Details

Data Generation

For the generation expansion planning data, we use a one-year case study (8,760 hours) and divide it into 73 data samples, each covering a 120-hour horizon. The specific input data and parameter values are available in the accompanying codebase shared in Appendix A.

To train both the primal and dual networks, we use the dataset of economic dispatch samples introduced in Chapter 5. The reader is referred to that chapter for a detailed explanation of the data generation process and its underlying motivation.

Exact Solver

Gurobi [21] is used as the exact solver throughout the experiments in this chapter.

Neural Network Architecture and Hyperparameters

To tune the neural network architecture and hyperparameters, we use Optuna [2]. The tuning inputs, along with the selected hyperparameters and resulting architecture, are detailed in Appendix C.5.

Hardware Configurations

The specific hardware configurations used for the experiments in this chapter are detailed in Appendix E.

7.3. Performance of Exact Benders, Inexact Benders, and Inexact Benders with Exact Refinement

The performance of the trained primal and dual networks on the test subset of the training data is presented in Table 7.1. The networks produce feasible solutions, achieving a duality gap of approximately 4%. It should be noted that the duality gap is not equal to the sum of the primal and dual optimality gaps, as the latter are computed relative to the true objective value. In contrast, the duality gap is computed relative to the predicted primal objective value, as explained in Section 3.4.

Table 7.2 presents the results of three Benders decomposition variants: *exact Benders*, where all subproblems are solved exactly; *inexact Benders*, where all subproblems are solved using the primal and dual networks; and *inexact Benders with exact refinement*, in which the method initially relies on the neural networks, but switches to an exact solver once progress stagnates, subsequently solving the problem

Table 7.1: Performance of the trained primal and dual networks on the economic dispatch test set. The known mean optimal objective, mean predicted objectives, and the corresponding optimality gaps are reported. Additionally, the duality gap between the primal and dual predictions is presented. Training times for both networks are also included.

Experiment	Opt. obj.	Pred. obj.	Opt. gap (%)	Duality gap (%)	Train time (s)
Primal	29293.887	29311.776	0.769	4.009	148.831
Dual	29293.887	29177.740	-3.802		143.155

Table 7.2: Performance comparison of exact Benders, inexact Benders, and inexact Benders with exact refinement. For each method, the resulting upper and lower bounds and corresponding duality gaps are reported. The duality gap is zero for exact Benders and inexact Benders with exact refinement, as both solve the problem to optimality. The number of Benders iterations is shown, with a breakdown of exact and inexact subproblem solves. Total runtime is reported, along with time spent on the master problem and on subproblems (exact or inexact). Average time per iteration is also provided. Results (mean ± standard deviation) × magnitude are averaged over 73 samples with 120 timesteps, each repeated five times to reduce randomness; standard deviations of zero and unit-scale factors are omitted for clarity.

Metric	Exact Benders	Inexact Benders	In. Benders + exact refinement
Upper bound	$(22.632 \pm 5.895) \times 10^6$	$(84.755 \pm 15.898) \times 10^7$	$(22.632 \pm 5.895) \times 10^6$
Lower bound	$(22.632 \pm 5.895) \times 10^6$	$(57.170 \pm 53.434) \times 10^5$	$(22.632 \pm 5.895) \times 10^6$
Duality gap (%)	0.000	99.249 ± 0.919	0.000
Total Iterations	74.233 ± 22.265	9.795 ± 3.476	76.863 ± 22.723
Exact Iterations	74.233 ± 22.265	0.000	67.068 ± 24.185
Inexact Iterations	0.000	9.795 ± 3.476	9.795 ± 3.476
Time master problem (s)	0.773 ± 0.420	0.022 ± 0.009	0.822 ± 0.453
Time subproblem (exact) (s)	0.645 ± 0.201	0.000	0.603 ± 0.224
Time subproblem (inexact) (s)	0.000	0.009 ± 0.003	0.009 ± 0.003
Total time (s)	1.418 ± 0.610	0.032 ± 0.012	1.434 ± 0.666
Time per iteration (s)	0.018 ± 0.003	0.003	0.018 ± 0.004

to optimality. The results are averaged across all 73 generation expansion planning instances with 120 timesteps each, over 5 repeats.

The resulting upper and lower bounds are reported, along with the corresponding duality gap. The total number of Benders iterations and a breakdown into exact and inexact iterations are provided. Additionally, the total solution time is recorded and divided into time spent solving the master problem exactly, solving subproblems exactly, and solving subproblems inexactly, using the neural networks. The average time per iteration is also reported.

The results demonstrate that while inexact Benders converges to a suboptimal solution, incorporating an exact refinement phase enables convergence to the optimal solution. Notably, although inexact Benders with exact refinement incurs slightly more total computational time than exact Benders, it often requires fewer exact iterations, highlighting its potential efficiency in settings where subproblem solving dominates runtime. Particularly, on these generation expansion planning instances, inexact Benders with exact refinement requires the most computational time, followed closely by exact Benders. Inexact Benders produces significantly faster predictions, with approximately two orders of magnitude reduction in total runtime. This improvement is attributed to a smaller number of iterations and a per-iteration time roughly half that of the other approaches. However, inexact Benders, which depends solely on PDL, does not yield a meaningful solution, producing a primal and dual pair with an average duality gap of around 99%.

Figure 7.1 shows a runtime comparison between inexact Benders with exact refinement and exact Benders. Blue dots indicate instances where a speedup was achieved by using inexact Benders with exact refinement, while orange dots denote slower instances. Additionally, the figure shows another comparison between the speedup ratio and the duality gap after the inexact iterations, revealing a trend wherein a smaller duality gap corresponds to a higher speedup ratio.

Furthermore, Figure 7.2 presents the optimality gap of the primal and dual predictions across the iterations of inexact Benders, on a single generation expansion planning instance. Both the primal and dual predictions diverge progressively away from the optimum as iterations advance, until progress stagnates.

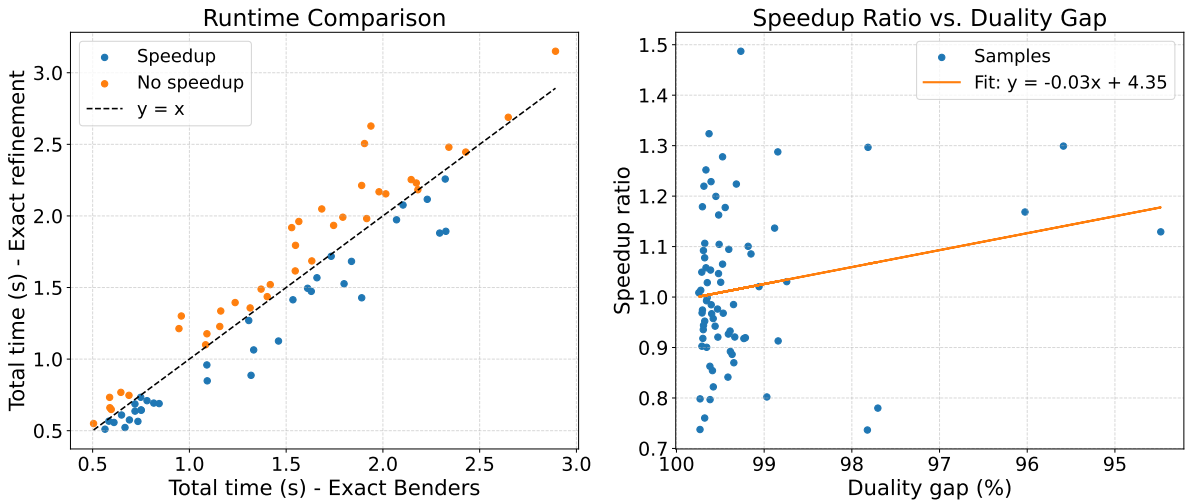

Figure 7.1: Runtime comparison between exact Benders and inexact Benders with exact refinement, evaluated on 73 economic dispatch instances, averaged over five repeats. The left plot shows a per-instance comparison, where each dot represents one instance: blue indicates a speedup using inexact Benders with exact refinement, and orange indicates slower performance. The right plot shows the relationship between the speedup ratio and the duality gap after the inexact phase, revealing a trend in which smaller duality gaps correspond to higher speedups.

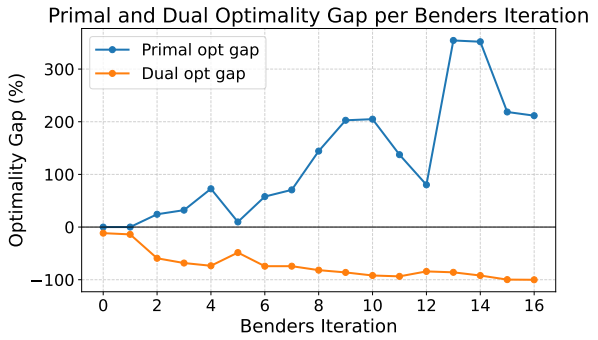

Figure 7.2: Predicted primal and dual optimality gaps over Benders iterations for the first generation expansion planning sample, evaluated using the inexact Benders approach. Each point represents the optimality gap at a given iteration: the blue curve shows the evolution of the primal optimality gap, while the orange curve shows the dual optimality gap.

Similarly, Figure 7.3 reports the optimality gaps of the primal and dual predictions, but now following the trajectory of exact Benders, where investment decisions at each iteration are obtained from the exact method, using the same problem instance as Figure 7.2. In this case, the optimality of the primal and dual predictions fluctuates, at times deviating from the known optimum and at other times approaching it closely.

7.3.1. Discussion of Results

The results show a promising proof of concept, where all of the theoretical guarantees provided in Section 7.1.2 are confirmed empirically through the experiments.

However, the duality gap between the primal and dual pair produced by inexact Benders, as shown in Table 7.2, is substantially larger than the duality gap observed on the test set of the training data, as reported in Table 7.1. This indicates a mismatch between the distribution of the training data, generated according to the scheme described in Chapter 5, and the data encountered during Benders iterations. This observation is further supported by Figures 7.2 and 7.3, where the primal and dual optimality gaps frequently deviate significantly from zero. Section 7.4 explores this in more detail.

Another contributing factor may be the misalignment between the loss function used during training, which is based on the average objective value, and the optimality gap, which is a relative measure. This issue has been discussed previously in Chapters 5 and 6. Specifically, instances with large objective

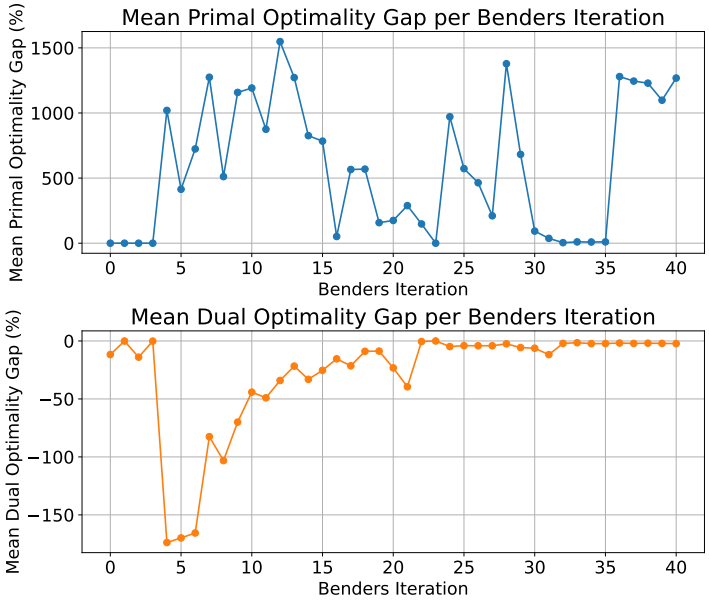

Figure 7.3: Mean predicted primal and dual optimality gaps computed when following the trajectory of exact Benders on the first generation expansion planning sample. The top plot shows the evolution of the mean primal optimality gap at each iteration, while the bottom plot displays the mean dual optimality gap.

values tend to dominate the training process when an average-based loss is used, potentially resulting in poor generalization on instances with smaller objective values. The precise cause of the degraded performance is investigated in Section 7.4.

The observed fluctuations in the optimality gap over iterations also suggest that the distribution of investment decisions changes dynamically throughout Benders decomposition. Consequently, the sampling strategy used for training should be adapted to reflect this evolving distribution.

Although Benders decomposition based solely on approximate, predicted solutions does not produce meaningful solutions, inexact Benders with exact refinement demonstrates promising potential. In the generation expansion planning instances, inexact Benders with exact refinement required, on average, over nine fewer exact iterations despite the limited quality of the predictions. It is reasonable to expect that improved predictive performance would result in a further reduction in the number of exact iterations. This intuition is further enforced by the trendline that can be seen in Figure 7.1. Moreover, as problem complexity increases, the time required to solve subproblems exactly is expected to grow significantly faster than the time required for inference using neural networks. Therefore, the results provide a compelling proof of concept.

While the reported speedups exclude training time, this cost is amortized across multiple problem instances in many practical settings. The benefits of the inexact Benders with exact refinement approach must be weighed against the cost of model training and online updates. Particularly in large-scale applications, it is expected that the benefits may outweigh the costs; however, this trade-off should be carefully evaluated for each use case.

Furthermore, the total number of iterations increases when using inexact Benders with exact refinement compared to exact Benders. Since the master problem must be solved at each iteration, the overall computational gain depends on the relative complexity of the master problem and the subproblems. This highlights a tradeoff between inexact and exact Benders, determined by the structural and computational properties of the problem being solved.

In summary, this section presents promising results on the integration of learning-based primal-dual solution pairs within Benders decomposition, providing a compelling proof of concept for future research. However, the conclusions are based on a relatively small instance and on models with limited predictive performance. It remains unclear how these results will extend to larger-scale or more heterogeneous problem settings. Nonetheless, the demonstrated gap reductions and runtime savings on

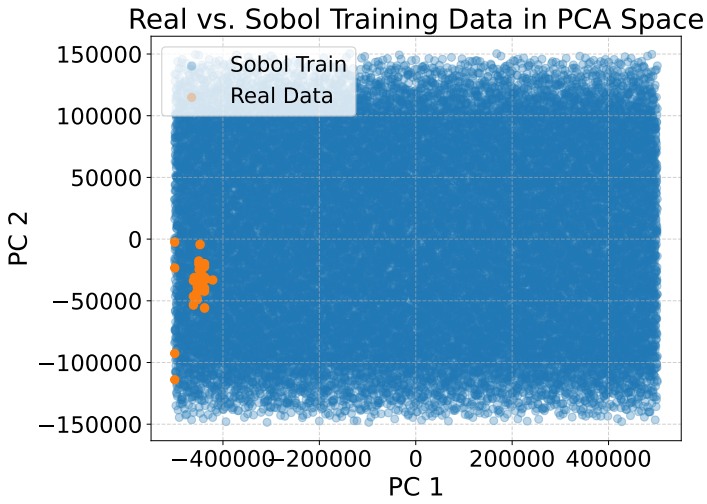

Figure 7.4: Comparison of real data generated during exact Benders decomposition on the first generation expansion planning sample, and the Sobol sequence training points, visualized in the space of the first two principal components after applying PCA. Blue points are the training data, and orange points are the real data.

moderately sized instances suggest that improved predictive models and broader training distributions could significantly extend the applicability and impact of this approach. Addressing the mismatch between the training data and the data encountered during Benders iterations remains a key direction for future work.

7.4. Comparison of Training Data and Benders-Generated Data

This section investigates the mismatch between the synthetic training dataset and the data encountered during Benders decomposition. Understanding this discrepancy is crucial for diagnosing limitations in model generalization and improving performance during deployment. Specifically, the training dataset is compared with the data collected during the iterations of exact Benders on the generation expansion planning instance.

Figure 7.4 visualizes the training points generated using the Sobol sequence alongside the real data generated during Benders decomposition, after reducing the data to two dimensions using principal component analysis (PCA)[1]. The figure shows that the training data is distributed nearly uniformly across the space, reflecting the motivation for using the Sobol sequence. In contrast, the real data points are highly clustered, occupying only a small region of the overall space.

Directly assessing the distribution of the training and real samples is challenging, as each data point is characterized by many distinct properties that could serve as axes for defining a distribution. It is not immediately clear which of these properties is most relevant. For example, a change in the investment decision may lead to a problem instance with or without missed demand, or with or without the need for flows, depending on the structure of the problem. These variations are often non-trivial to quantify in a meaningful or consistent way.

However, the same structure that is exploited by the dual network, as described in Section 6.3 which arises from prior knowledge of the optimal dual decision variables, can be used to assess the distribution by examining the frequency of the predicted classes in both the training data and the data encountered during Benders decomposition. Figure 7.5 presents the class counts for the training set and for the data generated during the iterations of exact Benders. A clear discrepancy can be observed between the two distributions, with many classes that are frequently encountered during Benders being significantly underrepresented in the training dataset.

Another important source of mismatch arises from the structure of the loss functions themselves. Since the losses are weighted by the magnitude of the objective value, instances with larger objectives exert

[1]PCA is a method for reducing dimensionality by projecting the data onto directions of maximum variance.

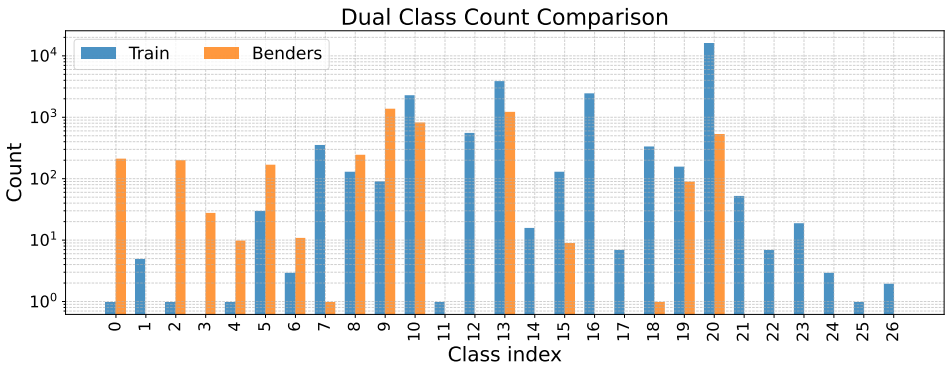

Figure 7.5: Comparison of dual class counts between the Sobol training data and the data generated during Benders decomposition on the first generation expansion planning sample. Each bar represents the number of samples for a given class index, where the class index is an arbitrary label assigned to distinguish different dual solution classes. Blue bars correspond to the training data and orange bars to the Benders data. The plot highlights differences in class distribution between the synthetic training set and the real data encountered during optimization. Note that the count axis is shown on a logarithmic scale.

more influence on training. This introduces a bias that can distort generalization, especially when performance is measured using the optimality gap rather than the raw objective prediction error. As a result, instances with larger objective values receive greater importance during training, which increases the emphasis placed on accurate prediction by the networks. To examine the effect of this weighting, the distribution with respect to the magnitude of the objective value is analyzed. Figure 7.6 presents scatter plots of the primal and dual optimality gaps against the objective values for both the training set and the data obtained from exact Benders. The black line represents the kernel density estimation with respect to the objective values, while the red dashed line indicates the kernel density estimation weighted by the objective values. It can be observed that the unweighted kernel density of the training data aligns well with the overall distribution of objective values. In contrast, the weighted kernel density is heavily concentrated around higher objective values, indicating that the primal and dual loss functions disproportionately emphasize these instances during training.

7.4.1. Discussion of Results

The figures clearly demonstrate a discrepancy between the distribution of the training data and the data generated during the iterations of Benders decomposition. This discrepancy is likely a key factor contributing to the reduced performance of the networks during deployment within Benders.

Figure 7.4 illustrates that the Sobol sequence samples the space as intended. In the space defined by the principal components, the real data appears to lie within the boundaries of the sampled training data. However, the sampled space spans a broader region than what is actually required, placing emphasis on areas that are rarely visited during Benders iterations. This suggests that near-uniform coverage of the input space is not sufficient. Figure 7.5 further supports this conclusion, showing that aligning the distribution of data encountered during Benders is more complex than simply covering the input space. The mismatch between the dual classes observed in the training data and those encountered during Benders indicates that the structure of the dual solutions is not captured adequately by the initial sampling procedure.

Figure 7.6 confirms the mismatch between the self-supervised losses and the optimality gap metric. A consistent pattern is observed, particularly in the dataset generated by Benders decomposition: optimality gaps tend to be large for instances with smaller objective values and decrease as the objective values increase. In regions where the weighted kernel density estimation is concentrated, the optimality gaps are relatively small, indicating that the networks have learned to make accurate predictions in areas where the loss function assigns the most weight. It is important to emphasize that the networks are not behaving incorrectly; they are minimizing the optimality gaps in the regions prioritized by the loss function. This behavior highlights a fundamental misalignment between the training objective and the evaluation metric. These findings suggest that effective learning in this context may require not only better sampling strategies but also redesigned loss functions or evaluation frameworks that are more tightly aligned with optimization performance.

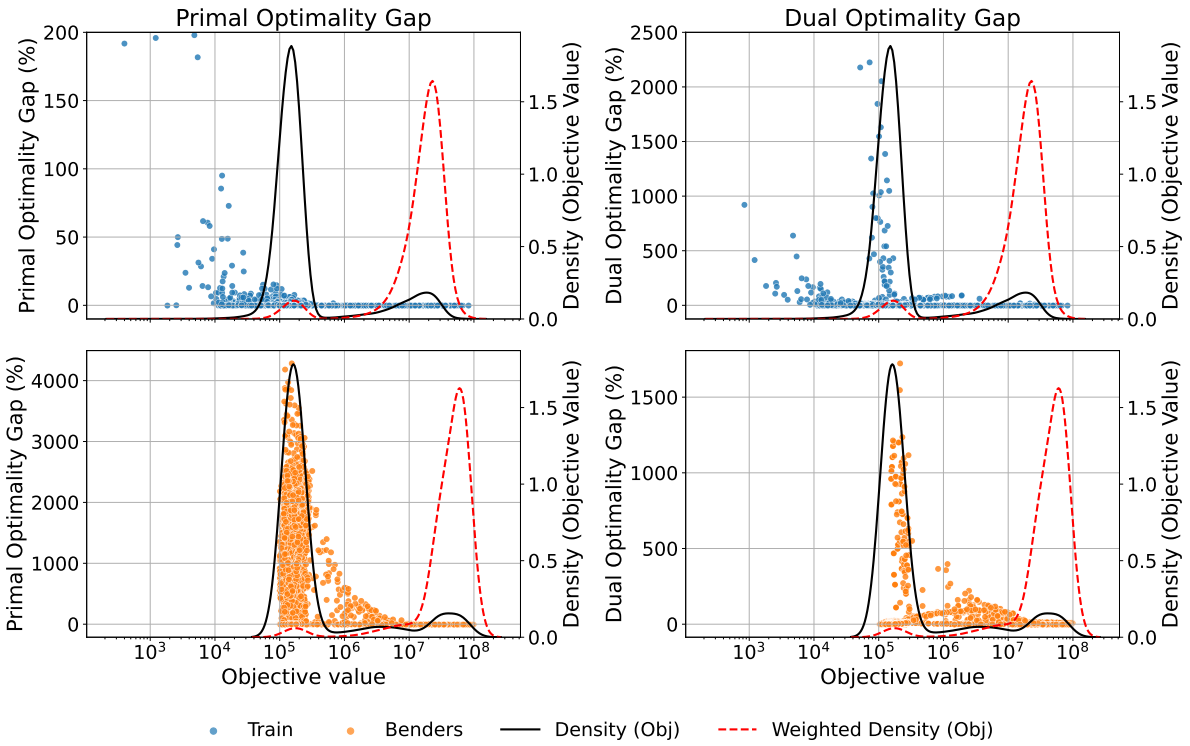

Figure 7.6: Scatter plots of primal and dual optimality gaps versus objective values for both the training set (top row) and the data obtained from exact Benders decomposition (bottom row). The black line shows the kernel density estimation of the objective values, while the red dashed line represents the kernel density estimation weighted by the objective values. The unweighted density of the training data closely matches the overall distribution of objective values, whereas the weighted density is concentrated around higher objective values. This indicates that the primal and dual loss functions place greater emphasis on instances with larger objective values during training.

7.5. Discussion

The results in this chapter demonstrate a promising proof of concept for integrating learning-based methods into Benders decomposition by predicting primal and dual solution pairs. This integration is empirically validated on a generation expansion planning problem; however, it preserves theoretical guarantees for all convex subproblems that can be formulated conically, including all linear programs, while potentially improving computational efficiency. In such problems, feasibility can be readily enforced on the dual predictions using the dual completion layer introduced in Section 6.2. Consequently, only valid cuts are produced during inexact iterations of learning-enhanced Benders decomposition, and the optimal solution can be recovered through exact refinement. However, when approximate solutions with bounds on suboptimality are desired, feasibility must also be enforced on the primal predictions.

Moreover, this approach opens the door to applying learning-assisted Benders decomposition in more complex settings, including problems with varying constraint matrices (as discussed in Chapter 4). Since the Benders-decomposed subproblems, by construction, will never have a varying constraint matrix, this reduces the complexity for learning-based predictors.

Similarly, frameworks based on the same principles may be developed for different decomposition techniques, such as Dantzig—Wolfe decomposition. In this case, instead of relying on dual feasible predictions during inexact iterations, the algorithm would depend on primal feasible predictions for the subproblems.

These results underscore the broad applicability of the proposed framework. Future improvements may focus on enhancing the training procedure and extending the inexact Benders algorithm, as described in the following subsections.

7.5.1. Training Procedure

There are multiple avenues for improving the training procedure of the primal and dual networks for use during Benders decomposition.

First, the research in this thesis demonstrates a proof of concept using offline learning. However, an online learning approach, where the training data is sampled during Benders decomposition, could mitigate several of the challenges observed in this chapter by generating training samples during Benders iterations. In such an online setting, the mismatch in training distribution is likely reduced because the model is trained directly on the data it encounters during inference, ensuring closer alignment between training and deployment distributions. It is worth considering, however, that the distribution of data evolves throughout the iterations of Benders decomposition. Distribution shift is a well-known challenge in online learning, and incorporating established methods from related fields, such as *deep reinforcement learning*[2], would be a valuable direction for future work.

Second, imbalanced datasets are often addressed in classical supervised classification by over-sampling underrepresented classes, under-sampling overrepresented classes, or assigning class-dependent weights. In supervised regression, target values are known, allowing the loss function to be rescaled so that all samples contribute equally, regardless of their magnitude. In contrast, the used learning method is self-supervised, which is advantageous because it eliminates the need to precompute computationally expensive targets. However, this also makes rebalancing more difficult, since the optimality gap cannot be computed without ground truth values. Nevertheless, the primal and dual networks offer a promising rebalancing mechanism through duality. The duality gap between a primal-dual pair provides a meaningful measure of optimality and can therefore be used to guide rebalancing efforts without explicit targets. Several directions using duality can be explored in future work. One possibility is to define a shared loss function for the primal and dual networks that directly minimizes the duality gap. Another approach is to investigate sampling strategies that prioritize training points with a large duality gap, thereby focusing learning on regions of the solution space that are currently underrepresented or poorly understood.

7.5.2. Inexact Benders

Several clear directions remain for extending inexact Benders using PDL or other learning-based methods.

First, the current setup predicts full primal solutions using the primal network. However, in cases where the decision-maker is only interested in the investment decisions and not in the full set of operational variables, this approach may introduce unnecessary complexity. In such cases, it may be sufficient to predict only the primal objective value, which could reduce model complexity and memory requirements—an advantage in large-scale settings. However, when predicting the objective value without access to the underlying primal decision variables, ensuring feasibility remains a key challenge in this case.

Second, the current implementation halts the inexact iterations once a generated cut no longer changes the investment decision produced by the master problem. This can lead to early termination following a single poor prediction from the dual network, even though subsequent predictions may still be informative. Future work could explore strategies that better utilize the dual network by restarting the algorithm from different investment decisions while retaining all previously generated valid cuts. Selecting an effective restarting point is not straightforward and requires further investigation.

In addition, methods that alternate between exact and inexact solving can be considered. For example, the algorithm could switch to exact solving when the predicted primal and dual pair exhibits a large duality gap, and return to inexact solving after a few iterations or based on a heuristic rule. Such adaptive strategies may improve overall performance and robustness by balancing predictive efficiency with solution accuracy.

Finally, the inexact predictions can also be leveraged during exact solving by providing the predicted primal and dual solutions as a warm start to the solver. For example, the search space could be restricted

[2]Deep reinforcement learning is a machine learning approach where an agent learns to make decisions by interacting with an environment. Distribution shift occurs because the agent's behavior changes during learning, affecting the data it collects.

based on the lower bound, potentially reducing the time required to reach optimality. An example of this approach is given by Václavík et al. [56].

All in all, the results demonstrate the feasibility and potential of integrating learning-based techniques capable of providing feasible solution predictions into Benders decomposition. While this chapter has focused on proof-of-concept experiments, the approaches outlined here provide a strong foundation, and the avenues for further improvement and extension are clear.

7.6. Conclusion

The results presented in this chapter demonstrate a promising proof of concept for integrating primal-dual learning into Benders decomposition. By learning to approximate primal and dual solutions, the approach enables the generation of valid Benders cuts with bounded suboptimality, offering the potential for reduced computational cost while preserving theoretical soundness under exact refinement.

Experiments demonstrate that the hybrid method—combining inexact and exact solving—achieves promising performance with fewer exact iterations than classical Benders. However, the effectiveness of purely inexact predictions remains limited due to the distributional mismatch of the training data, as well as a misalignment between the training loss and the optimality gap. A detailed analysis of the training and Benders-generated data confirms that the networks are biased toward regions with high objective value and often face out-of-distribution samples during deployment.

The findings underscore the need for improved sampling strategies, loss formulations, and adaptive solving procedures. Several directions for future work have been identified, including online learning, duality-aware loss functions, and warm-starting techniques. Together, these developments may enable more scalable and effective learning-enhanced decomposition algorithms.

8

Conclusion

This chapter concludes the thesis by reflecting on the role of self-supervised primal-dual learning (PDL), a method that jointly trains primal and dual neural networks, in providing solution quality guarantees for energy systems optimization. It also reflects on the integration of such learning-based predictions into Benders decomposition, aiming to accelerate constrained optimization while preserving theoretical guarantees.

Section 8.1 summarizes the main findings of this thesis. Next, Section 8.2 answers the research question posed at the outset. This is followed by Section 8.3, which discusses the research contributions and places them in a broader context. Finally, Section 8.4 outlines the limitations of this work and suggests directions for future research.

8.1. Summary of Contributions

To begin with, additional research extending prior work on PDL [47, 46] was conducted by varying different structural components of the benchmark problems. While prior work varied the right-hand side and linear term of the objective function, this work tested variations in the left-hand side and the quadratic objective term, which were found to be more challenging, offering insight into which problem structures may pose greater difficulty for PDL and why.

Building on this, PDL was applied to the economic dispatch domain for the first time, where initial primal predictions showed significantly worse performance than benchmark problems, with large optimality gaps and constraint violations. The introduction of differentiable repair and completion layers was found to be crucial for improving learning. A node-balancing repair layer, adapted from a different setting and extended to prioritize generation over unmet demand, further improved performance, reducing optimality gaps to within 2%. Learning transmission line flows, which are commonly present in energy systems planning problems, emerged as a new and significant challenge for PDL.

Next, the quality of dual predictions produced by the dual network trained alongside PDL was investigated and found to be insufficient, with the method effectively using the duals only to support primal learning. Moreover, a key limitation was identified: the use of repair layers, which are crucial for effective primal learning, eliminates constraint violations. This disrupts the dual target updates, rendering learning dual solutions impossible within the standard PDL framework using repair layers.

In response, a different dual loss function was proposed that directly optimizes the dual objective. This approach, combined with a dual completion layer from existing literature, was tested on the economic dispatch problem. A novel classification layer was introduced, incorporating prior knowledge of the dual structure. It significantly outperformed the completion layer on this problem, although it showed instability during training and occasional premature convergence.

Subsequently, a framework was developed for integrating learning methods with Benders decomposition. Theoretical guarantees on the duality gap and convergence to optimality under exact refinement

were proven to hold, provided that the dual predictions for the subproblem are feasible. This condition can be readily satisfied for convex subproblems admitting a conic formulation, including all linear programs, by utilizing an existing dual completion layer. Consequently, the guarantees apply to any mixed-integer linear program where integer variables are limited to the master problem of Benders decomposition.

Finally, a case study was conducted on Benders decomposition using the proposed primal and dual network architectures in a generation expansion planning problem. The results demonstrated a promising proof of concept, showing that the number of exact iterations can be reduced by incorporating PDL on the generation expansion planning instance. Moreover, the study illustrates how Benders can enable the use of PDL in problems where learning is complicated by the presence of investment variables in the left-hand side of the constraints.

8.2. Answer to the Research Question

The findings presented throughout the thesis collectively support the answer to the main research question posed in Chapter 1:

> *Can self-supervised primal-dual learning efficiently generate high-quality primal and dual solutions with verifiable optimality guarantees, and can these be integrated into Benders decomposition to accelerate energy systems planning?*

PDL has been shown to be incapable of producing high-quality dual solutions without modifying the dual loss function. However, when trained separately and with an objective-value-based dual loss, both the primal and dual networks are capable of generating high-quality solutions on the economic dispatch problem using differentiable repair and completion layers. Due to the limited performance of PDL without enforcing feasibility through repair layers, it was not possible to evaluate the effect of the dual loss function within the full PDL setup, where dual predictions are used to penalize constraint violations, and thus, results remain inconclusive whether PDL is able to generate high-quality primal-dual pairs in this setting. Nevertheless, a proof of concept on Benders decomposition demonstrates that integrating learned, approximate solutions while preserving theoretical performance guarantees is feasible. Although the results are not yet conclusive due to the relatively small size of the generation expansion planning problem, they highlight the potential of PDL and other learning-based methods to enable computational speedups with theoretical guarantees and bounds on suboptimality for energy systems planning.

8.3. Discussion

One of the key motivations for predicting primal-dual solution pairs, rather than only primal solutions, is the potential to address common concerns about the use of machine learning in optimization, particularly the perception of learned models as black boxes, as discussed in Section 1.1. However, a feasible primal-dual pair yields a provable duality gap, which can be used as a measure of prediction quality, regardless of the underlying predictor. This supports interpretability and builds trust, enhancing its practicality in applications such as energy systems planning. The contributions of this work provide insight into the strengths and limitations of learning such pairs in constrained optimization problems and offer a foundation for future research in this domain.

Integrating learning-based methods into classical optimization frameworks like Benders decomposition further enhances the approach's credibility. As demonstrated in this thesis, even when learned solutions are approximate, convergence to optimality can still be guaranteed through exact refinement. This hybrid setup provides a promising middle ground between computational efficiency and theoretical rigor, particularly in cases where solving all subproblems exactly is a limiting factor.

However, the suitability of PDL for predicting primal-dual pairs remains debatable. The findings in this thesis reveal that while primal predictions can be made accurate through repair and completion layers, the dual predictions produced by the standard PDL setup are of low quality. More critically, enforcing primal feasibility through repair layers removes the constraint violations that dual learning relies on. This raises a fundamental issue: although PDL is designed to learn both the primal and dual jointly, the benefits of doing so may be lost if the duals are learned through a separate mechanism, where primal

learning might suffer. Although PDL presents a compelling framework for joint learning of primal and dual solutions, its limitations in dual learning reduce its practical usefulness for this specific goal.

Furthermore, whether a provable primal-dual gap is necessary is ultimately context-dependent. In some applications, generating a feasible and near-optimal primal solution may be sufficient, which can be achieved through existing primal methods. In other cases, particularly when solution quality must be guaranteed, methods providing a primal-dual solution pair are preferred. However, this increases the overall complexity and cost of training. It is therefore essential to weigh the value of the duality gap against the computational effort required to learn it.

Finally, the broader applicability of PDL and other learning-based methods should be considered carefully. In scenarios where a problem instance is solved only once or infrequently, the time and resources required to train a model may not be justified. However, when similar problems are solved repeatedly, such as during Benders decomposition, training costs can be spread out over many instances. In such cases, especially with the use of online or adaptive training, learning-based methods become more attractive. Still, an inherent tradeoff remains between training time, inference speed, solver speed, and the resulting accuracy, which must be assessed on a case-by-case basis.

Overall, the results of this thesis highlight both the potential and the complexity of applying self-supervised learning methods, such as PDL, to structured optimization. They suggest that while predictive learning can be useful and provide theoretical guarantees, it must be tailored to the structure and requirements of each specific problem.

8.4. Limitations and Future Work

This section provides an overview of the limitations of this thesis and proposes future research directions to address these limitations.

8.4.1. Limitations

The work presented in this thesis has several limitations that constrain the generality and applicability of its findings. First, the proposed learning methods rely heavily on differentiable repair and completion layers to enforce feasibility. These layers require problem-specific knowledge and may not be available or easily designed for all problem types. Moreover, the proposed integration of domain information into the neural network architectures reduces generality.

Furthermore, the research is limited to linear programs and mixed-integer linear programs via Benders decomposition. All experiments were conducted on small-scale problem instances, which limits conclusions about scalability or performance in real-world systems. Additionally, the training data was generated independently of the data encountered during Benders iterations, resulting in a distribution mismatch that negatively affects prediction performance.

Learning transmission line flows emerged as a particular challenge, as these variables were more difficult to approximate accurately. Dual learning also exhibited instability and poor performance under the standard PDL setup, especially when constraint violations were removed by repair layers. While theoretical guarantees were established for problems admitting a conic convex formulation, this assumption excludes broader classes of optimization problems.

Finally, the problem formulations considered in this thesis omit several inter-period constraints commonly found in energy systems, such as ramping and storage dynamics. Including these components could significantly increase the difficulty of learning and may alter the effectiveness of the proposed methods.

8.4.2. Future Work

Several promising directions remain for extending and improving the work presented in this thesis. First, addressing the distribution mismatch between training samples and the data generated during Benders iterations is necessary. More realistic or adaptive sampling strategies, potentially by leveraging information on the duality gap, will likely significantly improve prediction performance in deployment. A promising starting point is integrating online learning into Benders decomposition.

Another key step is scaling up the problem instances to better reflect the complexity of real-world en-

ergy systems. Larger models would enable a more robust evaluation of computational speedups and highlight new challenges in learning under increased dimensionality. Including ramping and storage is a clear next step towards realistic problems.

Improving the prediction of transmission line flows remains an open challenge. Architectural changes or dedicated modules may be required to better capture their behavior. Related to this, more expressive neural network architectures, particularly graph neural networks, offer promising directions. Graph neural networks can leverage two complementary approaches to exploit the underlying problem structure. First, they can embed the physical topology of the power network, enabling the model to more effectively learn interactions involving multiple nodes, such as power flow across transmission lines. Second, graph neural networks can represent a constrained optimization problem's structure by modeling variables and constraints as nodes in a graph, connected when they appear in the same expression. This formulation allows learning the interactions between decision variables and constraints directly. A notable example of this approach is provided by Chen et al. [11]. An additional benefit of graph neural networks is their ability to generalize across problem instances where not only parameters, but also the underlying structure, such as constraints or topologies, vary. This capability may reduce the need for retraining, and thus, strengthens the case for learning-based methods.

Finally, further research into the interaction between primal and dual learning is needed. One open question is how learning dual variables using an objective-value-based loss affects the quality of the learned primal solutions in the context of primal-dual learning. In addition, jointly learning primal-dual pairs creates opportunities for stronger synergies than fully decoupled approaches. Leveraging the structure of duality during training is likely to improve both feasibility and optimality. Promising directions include rebalancing the training distribution based on the magnitude of the duality gap or incorporating the KKT conditions to enforce optimality constraints on the predicted primal-dual pair.

References

[1] Akshay Agrawal et al. "Differentiable Convex optimization layers". In: *arXiv* (*Cornell University*) 32 (Oct. 2019), pp. 9558–9570. URL: https://arxiv.org/pdf/1910.12430.pdf.

[2] Takuya Akiba et al. "Optuna: A next-generation hyperparameter optimization framework". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. July 2019, pp. 2623–2631. DOI: 10.1145/3292500.3330701.

[3] Brandon Amos and J. Zico Kolter. "OptNet: differentiable optimization as a layer in neural networks". In: Aug. 2017, pp. 136–145. URL: http://proceedings.mlr.press/v70/amos17a/amos17a.pdf.

[4] R. Andreani et al. "On Augmented Lagrangian Methods with General Lower-Level Constraints". In: *SIAM Journal on Optimization* 18.4 (Nov. 2007), pp. 1286–1309. DOI: 10.1137/060654797.

[5] Matthijs Arnoldus. "Exploiting Symmetry in the Generation Expansion Planning Problem to Accelerate Crossover". In: (2024). Available at https://resolver.tudelft.nl/uuid:e698df15-05ac-4b74-84da-bbe30c29adba.

[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization". In: *arXiv* (*Cornell University*) (Jan. 2016). DOI: 10.48550/arxiv.1607.06450.

[7] Olubayo M. Babatunde, Josiah L. Munda, and Yskandar Hamam. "A comprehensive state-of-the-art survey on power generation expansion planning with intermittent renewable energy source and energy storage". In: *International Journal of Energy Research* 43.12 (Mar. 2019), pp. 6078–6107. DOI: 10.1002/er.4388.

[8] Chiranjib Bhowmik et al. "Optimal green energy planning for sustainable development: A review". In: *Renewable and Sustainable Energy Reviews* 71 (Dec. 2016), pp. 796–813. DOI: 10.1016/j.rser.2016.12.105.

[9] Stefan Borozan et al. "Machine learning-enhanced benders decomposition approach for the multi-stage stochastic transmission expansion planning problem". In: *Electric Power Systems Research* 237 (2024), p. 110985.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Mar. 2004. DOI: 10.1017/cbo9780511804441.

[11] Ziang Chen et al. "On representing linear programs by graph neural networks". In: *arXiv* (*Cornell University*) (Jan. 2022). DOI: 10.48550/arxiv.2209.12288.

[12] Ld.S. Coelho and V.C. Mariani. "Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect". In: *IEEE Transactions on Power Systems* 21.2 (May 2006), pp. 989–996. DOI: 10.1109/tpwrs.2006.873410.

[13] Antonio J. Conejo et al. *Decomposition techniques in Mathematical Programming*. Jan. 2006. DOI: 10.1007/3-540-27686-6.

[14] G. Cote and M. Laughton. "Decomposition techniques in power system planning: the Benders partitioning method". In: *International Journal of Electrical Power & Energy Systems* 1.1 (Apr. 1979), pp. 57–64. DOI: 10.1016/0142-0615(79)90032-2.

[15] George B. Dantzig. *Linear programming and extensions*. Jan. 1963. DOI: 10.7249/r366.

[16] George B. Dantzig and Philip Wolfe. "Decomposition principle for linear programs". In: *Operations Research* 8.1 (Feb. 1960), pp. 101–111. DOI: 10.1287/opre.8.1.101.

[17] Priya L. Donti and J. Zico Kolter. "Machine learning for sustainable energy systems". In: *Annual Review of Environment and Resources* 46.1 (Aug. 2021), pp. 719–747. DOI: 10.1146/annurev-environ-020220-061831.

[18] Priya L. Donti, David Rolnick, and J Zico Kolter. *DC3: A learning method for optimization with hard constraints*. Jan. 2021. DOI: 10.48550/arxiv.2104.12225.

[19] K. R. Frisch. “The logarithmic potential method of convex programming”. In: *Memorandum, University Institute of Economics, Oslo* 5.6 (1955).

[20] A. M. Geoffrion. “Generalized Benders decomposition”. In: *Journal of Optimization Theory and Applications* 10.4 (Oct. 1972), pp. 237–260. DOI: 10.1007/bf00934810.

[21] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2024. URL: https://www.gurobi.com.

[22] Magnus R. Hestenes. “Multiplier and gradient methods”. In: *Journal of Optimization Theory and Applications* 4.5 (Nov. 1969), pp. 303–320. DOI: 10.1007/bf00927673.

[23] Sepp Hochreiter. “The vanishing gradient problem during learning recurrent neural nets and problem solutions”. In: *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems* 06.02 (Apr. 1998), pp. 107–116. DOI: 10.1142/s0218488598000094.

[24] Cara A. Horowitz. “Paris Agreement”. In: *International Legal Materials* 55.4 (Aug. 2016), pp. 740–755. DOI: 10.1017/s0020782900004253.

[25] International Renewable Energy Agency (IRENA). *World Energy Transitions Outlook 2023: 1.5°C Pathway*. 2023. URL: https://www.irena.org/Publications/2023/Jun/World-Energy-Transitions-Outlook-2023.

[26] Jacobs, Ben. *Self-Supervised Learning with Formal Guarantees for Energy Systems Optimization*. June 2025. DOI: 10.5281/zenodo.15656401.

[27] Huiwen Jia and Siqian Shen. “Benders cut classification via support vector machines for solving Two-Stage stochastic programs”. In: *INFORMS Journal on Optimization* 3.3 (Mar. 2021), pp. 278–297. DOI: 10.1287/ijoo.2019.0050.

[28] Diederik P. Kingma and Jimmy Lei Ba. “Adam: A method for stochastic optimization”. In: *arXiv* (*Cornell University*) (Jan. 2014). DOI: 10.48550/arxiv.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[29] Michael Klamkin, Mathieu Tanneau, and Pascal Van Hentenryck. “Dual Interior-Point Optimization Learning”. In: *arXiv* (*Cornell University*) (Feb. 2024). DOI: 10.48550/arxiv.2402.02596.

[30] Nikolaos E. Koltsaklis and Athanasios S. Dagoumas. “State-of-the-art generation expansion planning: A review”. In: *Applied Energy* 230 (Sept. 2018), pp. 563–589. DOI: 10.1016/j.apenergy.2018.08.087.

[31] James Kotary and Ferdinando Fioretto. “Learning Constrained Optimization with Deep Augmented Lagrangian Methods”. In: *arXiv* (*Cornell University*) (Mar. 2024). DOI: 10.48550/arxiv.2403.03454.

[32] James Kotary et al. “End-to-End Constrained Optimization Learning: A survey”. In: *arXiv* (*Cornell University*) (Jan. 2021). DOI: 10.48550/arxiv.2103.16378.

[33] Leander Kotzur et al. “A modeler’s guide to handle complexity in energy systems optimization”. In: *Advances in Applied Energy* 4 (Aug. 2021), p. 100063. DOI: 10.1016/j.adapen.2021.100063.

[34] Mengyuan Lee et al. “Accelerating generalized benders decomposition for wireless resource allocation”. In: *IEEE Transactions on Wireless Communications* 20.2 (Oct. 2020), pp. 1233–1247. DOI: 10.1109/twc.2020.3031920.

[35] C. E. Lemke. “The dual method of solving the linear programming problem”. In: *Naval Research Logistics Quarterly* 1.1 (Mar. 1954), pp. 36–47. DOI: 10.1002/nav.3800010107.

[36] Meiyi Li, Soheil Kolouri, and Javad Mohammadi. “Learning to Solve Optimization Problems with Hard Linear Constraints”. In: *arXiv* (*Cornell University*) (Jan. 2022). DOI: 10.48550/arxiv.2208.10611.

[37] Zhigang Li et al. “Adjustable robust Real-Time Power dispatch with Large-Scale wind Power integration”. In: *IEEE Transactions on Sustainable Energy* 6.2 (Jan. 2015), pp. 357–368. DOI: 10.1109/tste.2014.2377752.

[38] Dustin McLarty et al. “Dynamic economic dispatch using complementary quadratic programming”. In: *Energy* 166 (Oct. 2018), pp. 755–764. DOI: 10.1016/j.energy.2018.10.087.

[39] Bierlaire Michel. *Optimization: Principles and Algorithms*. Mar. 2015. DOI: 10.55430/6116v1mb.

[40] Ilias Mitrai and Prodromos Daoutidis. "Computationally efficient solution of mixed integer model predictive control problems via machine learning aided Benders Decomposition". In: *Journal of Process Control* 137 (Apr. 2024), p. 103207. DOI: 10.1016/j.jprocont.2024.103207.

[41] Ilias Mitrai and Prodromos Daoutidis. "Learning to initialize generalized benders decomposition via active learning". In: *FOCAPO/CPC, San Antonio, Texas* (2023). URL: https://skoge.folk.ntnu.no/prost/proceedings/focapo-cpc-2023/Oral%20Talks/9_Oral.pdf.

[42] Ilias Mitrai and Prodromos Daoutidis. "Machine Learning-Based initialization of generalized benders decomposition for mixed integer model predictive control". In: vol. 2. July 2024, pp. 4460–4465. DOI: 10.23919/acc60939.2024.10644418.

[43] Rahimeh Neamatian Monemi, Shahin Gelareh, and Nelson Maculan. "A machine learning based branch-cut-and-Benders for dock assignment and truck scheduling problem in cross-docks". In: *Transportation Research Part E Logistics and Transportation Review* 178 (Sept. 2023), p. 103263. DOI: 10.1016/j.tre.2023.103263.

[44] Mouad Morabit, Guy Desaulniers, and Andrea Lodi. "Machine-Learning–Based column selection for column generation". In: *Transportation Science* 55.4 (June 2021), pp. 815–831. DOI: 10.1287/trsc.2021.1045.

[45] Vishwamitra Oree, Sayed Z. Sayed Hassen, and Peter J. Fleming. "Generation expansion planning optimisation with renewable energy integration: A review". In: *Renewable and Sustainable Energy Reviews* 69 (Dec. 2016), pp. 790–803. DOI: 10.1016/j.rser.2016.11.120.

[46] Seonho Park and Pascal Van Hentenryck. *Self-Supervised learning for Large-Scale preventive security constrained DC optimal power flow*. Apr. 2024. DOI: 10.48550/arxiv.2311.18072.

[47] Seonho Park and Pascal Van Hentenryck. "Self-Supervised Primal-Dual learning for constrained optimization". In: *arXiv* (*Cornell University*) (Nov. 2022). DOI: 10.48550/arxiv.2208.09046. URL: https://arxiv.org/abs/2208.09046.

[48] M. J. D Powell. "A method for nonlinear constraints in minimization problems". In: *Optimization* (1969). Publisher: Academic Press, pp. 283–298. URL: https://cir.nii.ac.jp/crid/1574231876073595264.

[49] Guancheng Qiu, Mathieu Tanneau, and Pascal Van Hentenryck. "Dual Conic proxies for AC optimal power flow". In: *arXiv* (*Cornell University*) (Mar. 2024). DOI: 10.48550/arxiv.2310.02969.

[50] Ragheb Rahmaniani et al. "The Benders decomposition algorithm: A literature review". In: *European Journal of Operational Research* 259.3 (Dec. 2016), pp. 801–817. DOI: 10.1016/j.ejor.2016.12.005.

[51] Gerald Reid and Lawrence Hasdorff. "Economic dispatch using quadratic programming". In: *IEEE Transactions on Power Apparatus and Systems* PAS-92.6 (Nov. 2007), pp. 2015–2023. DOI: 10.1109/tpas.1973.293582.

[52] Elias Ridha, Lars Nolting, and Aaron Praktiknjo. "Complexity profiles: A large-scale review of energy system models in terms of complexity". In: *Energy Strategy Reviews* 30 (June 2020), p. 100515. DOI: 10.1016/j.esr.2020.100515.

[53] R. T. Rockafellar. "The multiplier method of Hestenes and Powell applied to convex programming". In: *Journal of Optimization Theory and Applications* 12.6 (Dec. 1973), pp. 555–562. DOI: 10.1007/bf00934777.

[54] I.M Sobol. "On the distribution of points in a cube and the approximate evaluation of integrals". In: *USSR Computational Mathematics and Mathematical Physics* 7.4 (Jan. 1967), pp. 86–112. DOI: 10.1016/0041-5553(67)90144-9.

[55] Mathieu Tanneau and Pascal Van Hentenryck. "Dual Lagrangian learning for Conic optimization". In: *arXiv* (*Cornell University*) (Feb. 2024). DOI: 10.48550/arxiv.2402.03086.

[56] Roman Václavík et al. "Accelerating the Branch-and-Price algorithm using machine learning". In: *European Journal of Operational Research* 271.3 (May 2018), pp. 1055–1069. DOI: 10.1016/j.ejor.2018.05.046.

[57] Pascal Van Hentenryck. "Optimization learning". In: *arXiv* (*Cornell University*) (Jan. 2025). DOI: 10.48550/arxiv.2501.03443.

[58] Tony J. Van Roy. "Cross decomposition for mixed integer programming". In: *Mathematical Programming* 25.1 (Jan. 1983), pp. 46–63. DOI: 10.1007/bf02591718.

[59] Johannes Varga et al. "Speeding Up Logic-Based Benders Decomposition by Strengthening Cuts with Graph Neural Networks". In: *Lecture notes in computer science*. Jan. 2024, pp. 24–38. URL: https://doi.org/10.1007/978-3-031-53969-5_3.

[60] Tianhe Yu et al. "Gradient surgery for Multi-Task learning". In: *arXiv* (*Cornell University*) (Jan. 2020). DOI: 10.48550/arxiv.2001.06782.

A

Codebase

The codebase developed and used in this thesis is publicly available at `https://github.com/benjacobs1999/masterthesis` and permanently archived on Zenodo at [26].

The repository includes the input data, experiment configurations, and resulting output data. In addition, the code used to generate the figures and tables presented in this thesis is also provided.

Furthermore, the repository contains configuration files, data generation scripts, problem definitions, and training code. Configuration files (`config.json`, `config.toml`) specify problem instances and training parameters. Dataset generation is handled by `create_gep_dataset.py` and `create_QP_dataset.py`, supported by problem definitions in `gep_problem.py`, `gep_problem_operational.py`, and `QP_problem.py`. Neural network architectures and training routines are defined in `networks.py`, `primal_dual.py`, and executed via `main.py`. Logging is handled by `logger.py`. Benders decomposition logic is implemented in `gep_benders.py`.

B

Reproduction of Primal-Dual Learning on quadratic programming benchmarks

In order to accurately investigate Primal-Dual Learning, a correct and validated implementation is first required. While the original publication [47] provides a comprehensive description of the algorithm, including pseudo-code and implementation details, no source code is provided. As a result, the implementation must be reproduced based solely on the information in the paper. Once the reproduced implementation closely matches the original by comparing results on the same benchmark problems, it serves as a validated basis for the analyses conducted throughout this thesis.

Park and Van Hentenryck [47] evaluate Primal-Dual Learning on four benchmark datasets: convex quadratic programming (QP), non-convex QP, quadratically constrained QP, and optimal power flow problems. In this thesis, the convex and non-convex QP benchmarks are considered. This selection is motivated, first, by the fact that reproducing similar results on two of the four benchmarks is deemed sufficient to validate the correctness of the implementation. Second, this thesis focuses on linear programming problems, which are convex. In contrast, quadratically constrained QP and optimal power flow problems are increasingly non-convex and therefore less relevant to the scope of this work.

The formulation of the convex QP benchmark is provided in Problem (3.1). For the non-convex QP benchmark, the linear term of the objective function is modified to take the sine of the decision variables. The objective function then becomes:

$$\frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{p}^\top \sin(\mathbf{x}). \tag{B.1}$$

For both problem types, a dataset $\mathscr{D}$ is constructed by generating multiple instances through variation of the right-hand side of the equality constraints $\mathbf{d}$, which then serve as input to both the primal and dual networks.

$$\mathscr{D} = \left\{\mathbf{z}^{(i)} = \mathbf{d}^{(i)}\right\}_{i=1}^{S} \tag{B.2}$$

The dataset contains 10000 instances, each with 100 decision variables, and 50 inequality and equality constraints: $Q = 100$, $O = M = 50$. The dataset is split into training data, validation data, and test data with respective sizes of 80%, 10%, and 10%. During training, the hyperparameters and neural network configuration are kept consistent with those in the first paper on PDL [47]. However, certain ambiguities remain regarding its implementation. The resolutions to these ambiguities and the exact hyperparameter and neural network configurations are provided in Appendix C.1.

While training, the neural network is saved when it performs better on the validation set than in previous steps. To decide this, a set of thresholds is used for measuring constraint violations. A threshold is considered satisfied if the maximum violation of both inequality and equality constraints is smaller

than that threshold. At each iteration, the network is saved if it has the lowest objective value so far among all networks that satisfy the same threshold. After all iterations are complete, the network with the lowest objective value that satisfies the lowest threshold is selected and evaluated on the test dataset.

The results of our implementation of PDL are compared to the previously reported results [47] in Table B.1. From these results, it can be concluded that PDL has been successfully reproduced on the convex and non-convex QP benchmark. Any minor deviations in the results are likely attributed to model selection.

Origin	Objective			Inequality		Equality	
	Opt.	Predicted	Opt. gap (%)	Max	Mean	Max	Mean
			Convex QP				
Original [47]	−15.047	−15.017 ± 0.009	0.176 ± 0.054	0.001	0.000	0.005 ± 0.001	0.002
Ours	−15.037	−15.022 ± 0.002	0.104 ± 0.016	0.002	0.000	0.006	0.002
			Non-convex QP				
Original [47]	−11.592	−11.552 ± 0.006	0.324 ± 0.051	0.001	0.000	0.004 ± 0.001	0.001
Ours	−11.583	−11.561 ± 0.001	0.190 ± 0.007	0.002	0.000	0.004 ± 0.001	0.001

Table B.1: Reproduction results of PDL on the convex and non-convex quadratic programming benchmarks. Reported metrics include optimal and predicted objective values, percentage gap (%), and maximum and mean constraint violations for both inequality and equality constraints. Results, averaged over 5 repeats, are presented as mean ± standard deviation; standard deviations of zero are omitted for clarity.

C

Neural Network Architecture and Hyperparameter Configurations

In this appendix, the hyperparameters employed for training the neural networks used throughout the thesis are detailed.

C.1. Reproduction

For the reproduction of PDL presented in Appendix B, the hyperparameters and implementation have been kept consistent with the original paper. For training the models, the Adam optimizer [28] is employed. The learning rate is decayed using the PyTorch 'ReduceLROnPlateau' learning rate scheduler. Furthermore, for the dual net, we use a single output layer where the dual variables are concatenated. For both the learning rate scheduler and the dual net output layer, it is unclear whether our implementation aligns with the one used in the original paper by Park and Van Hentenryck [47].

An overview of the specific hyperparameters used is given in Table C.1.

Table C.1: Hyperparameter configuration used for training the primal and dual networks in the reproduction experiments.

Parameter	Value
τ	0.8
ρ	0.5
$\rho_{\max}$	5000
η	10
Outer iterations	10
Inner iterations	500
Batch size	200
Primal learning rate	10^{-4}
Dual learning rate	10^{-4}
Learning rate decay	0.99
Learning rate scheduler patience	10

The neural network architecture for both the primal and the dual net consists of a multi-layer perceptron with two hidden layers, each containing 500 neurons, following the work by Park and Van Hentenryck [47].

C.2. Chapter 4

The hyperparameters shown in Table C.1 are the *original* hyperparameters used for the experiments in Chapter 4.

For the trials with *random* hyperparameters, only the PDL-specific parameters (τ, ρ, $\rho_{\max}$, η) were sampled uniformly from the ranges given in Table C.2, following the grid-search values of Park and Van Hentenryck [47]. All other training hyperparameters, such as number of iterations, batch size, and learning rate, were held constant at the values in Table C.1, since they were not explicitly tuned in Park and Van Hentenryck [47]. Therefore, they are assumed to remain appropriate across the various problem datasets.

Table C.2: Candidate values for randomly sampled PDL-specific hyperparameters used in the experiments in Chapter 4.

Parameter	Values
τ	{0.5, 0.6, 0.7, 0.8, 0.9}
ρ	{0.1, 0.5, 1, 10}
$\rho_{\max}$	{1000, 5000, 10000, 50000}
η	{1, 1.5, 2, 5, 10}

The values of the randomly sampled PDL-specific hyperparameters for each repeat are presented in Table C.3.

Table C.3: Specific hyperparameter configurations for each of the five runs with *random* hyperparameters presented in Chapter 4.

Parameter	Repeat 1	Repeat 2	Repeat 3	Repeat 4	Repeat 5
τ	0.9	0.9	0.7	0.5	0.9
ρ	10	0.1	0.1	1.0	10.0
$\rho_{\max}$	50000	5000	50000	10000	50000
η	1.5	1.0	2.0	10.0	10.0

The neural network architecture of the primal and dual networks is a multi-layer perceptron with two hidden layers, each containing 500 neurons, following the work by Park and Van Hentenryck [47].

C.3. Chapter 5

The hyperparameters used for training the primal and dual networks in Chapter 5 remain largely consistent with those presented in Table C.1. However, the batch size is increased to 2000 to reduce computational time.

The neural network architectures continue to be multi-layer perceptrons. Following the updated methodology presented by Park and Van Hentenryck [46], which addresses security-constrained optimal power flow, a problem more closely related to economic dispatch than the original convex quadratic programming benchmarks, the networks' depth is increased to four layers. Additionally, the hidden layer size is adjusted to scale with the input dimension. Although Park and Van Hentenryck [46] utilized a scaling factor of 1.5, this was found insufficient for the relatively small problem instances considered throughout this thesis. Consequently, the factor was increased to 20.

C.4. Chapter 6

The hyperparameters used to train the dual networks in Chapter 6 are the same as those described in Section C.3.

C.5. Chapter 7

The hyperparameters, as well as the neural network architecture used for training the primal network in Chapter 7 are tuned using Optuna [2]. Optuna is a hyperparameter optimization framework that efficiently searches for well-performing configurations using probabilistic, model-based optimization methods. The PDL-specific hyperparameters are not tuned, as the primal network outputs feasible solutions, and therefore does not require constraint violation penalization using PDL.

The Optuna study consists of 50 trials. Additionally, underperforming trials are pruned based on the median, excluding the initial five startup trials and the first 2000 warmup steps of each trial. Pruning evaluations are conducted at 10-step intervals.

The input ranges per hyperparameter for Optuna are specified in Table C.4. The best hyperparameter configurations found by Optuna are detailed in Table C.5.

The hyperparameters for the dual network are not tuned because of the instability in the performance of the dual network using the classification layer reported in Chapter 6. Therefore, tuning cannot be done properly. Instead, a well-performing model trained using the same hyperparameters and architecture as those in Chapter 6 is used.

Table C.4: Hyperparameter search space used for Optuna tuning. For continuous and integer types, the bounds indicate the range within which values are sampled. For categorical types, the listed values are directly selected.

Parameter	Type	Search space
Primal learning rate	continuous	$[10^{-5}, 10^{-3}]$
Dual learning rate	continuous	$[10^{-5}, 10^{-3}]$
Hidden size factor	integer	[2, 50]
Number of layers	integer	[2, 8]
Decay	continuous	[0.98, 0.999]
Batch size	categorical	{2048, 4096}

Table C.5: Best hyperparameter configuration found by Optuna. The configuration is used to train the primal network used throughout Chapter 7.

Parameter	**Value**
Primal learning rate	6.79×10^{-4}
Hidden size factor	28
Number of layers	2
Decay	0.999
Batch size	2048

D

Dual derivation

The primal formulation of the convex quadratic programming benchmark is:

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^Q}{\text{minimize}} \quad & \tfrac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{r}^\top \mathbf{x} \\ \text{subject to} \quad & \mathbf{G}\mathbf{x} \leq \mathbf{b}, \\ & \mathbf{H}\mathbf{x} = \mathbf{d}. \end{aligned} \tag{D.1}$$

Its Lagrangian is defined as:

$$\mathscr{L}_{\text{QP}}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) \coloneqq \tfrac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{r}^\top \mathbf{x} + \boldsymbol{\mu}^\top (\mathbf{G}\mathbf{x} - \mathbf{b}) + \boldsymbol{\lambda}^\top (\mathbf{H}\mathbf{x} - \mathbf{d}). \tag{D.2}$$

The stationarity condition results in:

$$\nabla_{\mathbf{x}} \mathscr{L}_{\text{QP}}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \mathbf{Q}\mathbf{x} + r + \mathbf{G}^\top \boldsymbol{\mu} + \mathbf{H}^\top \boldsymbol{\lambda} = 0 \tag{D.3}$$

Solving for $\mathbf{x}$, we get:

$$\mathbf{x} = -\mathbf{Q}^{-1}(r + \mathbf{G}^\top \boldsymbol{\mu} + \mathbf{H}^\top \boldsymbol{\lambda}) \tag{D.4}$$

For the same reasons described in Section 2.4, $\boldsymbol{\mu} \geq 0$. When substituting (D.4) in the Lagrangian (D.2), the dependence on $\mathbf{x}$ vanishes, resulting in the dual formulation:

$$\underset{\boldsymbol{\mu}, \boldsymbol{\lambda}}{\text{maximize}} \quad -\tfrac{1}{2}(\mathbf{r} + \mathbf{G}^\top \boldsymbol{\mu} + \mathbf{H}^\top \boldsymbol{\lambda})^\top \mathbf{Q}^{-1}(\mathbf{r} + \mathbf{G}^\top \boldsymbol{\mu} + \mathbf{H}^\top \boldsymbol{\lambda}) - \boldsymbol{\mu}^\top \mathbf{b} - \boldsymbol{\lambda}^\top \mathbf{d} \tag{D.5a}$$

$$\text{subject to} \quad -\boldsymbol{\mu} \leq 0. \tag{D.5b}$$

E

Hardware Specifications

Table E.1 summarises the hardware configuration of the MacBook Pro used for the experiments presented throughout this thesis.

Table E.1: Hardware configuration.

Component	Specification
Model identifier	Mac14,9
Chip	Apple M2 Pro
CPU Cores	10 (6 performance + 4 efficiency)
Memory	32 GB
MacOS version	Ventura 13.2