ADVANCES IN LARGE-SCALE SECURITY-CONSTRAINED ECONOMIC DISPATCH: LOSS MODELING, STOCHASTIC DISPATCH, AND PROXY VERIFICATION

A Dissertation
Presented to
The Academic Faculty

By

Haoruo Zhao

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Engineering
H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology

May 2025

ADVANCES IN LARGE-SCALE SECURITY-CONSTRAINED ECONOMIC DISPATCH: LOSS MODELING, STOCHASTIC DISPATCH, AND PROXY VERIFICATION

Thesis committee:

Dr. Pascal Van Hentenryck, advisor
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Daniel Molzahn
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Constance Crozier
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Hassan Hijazi
Senior R&D Scientist
Gurobi

Dr. Mathieu Dahan
H. Milton Stewart School of Industrial and Systems Engineering
Georgia Institute of Technology

Date approved: April 24, 2025

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Pascal Van Hentenryck, for his guidance throughout my PhD journey, for the opportunities he has given me to tackle interesting problems in optimization and machine learning for energy systems, for his support in fostering a dynamic research environment, and for the many qualities he has taught me. I am grateful to Dr. Hassan Hijazi for his mentorship in neural network verification. His creativity and support have opened new horizons in my research. I also would like to thank my committee members, Dr. Constance Crozier, Dr. Daniel Molzahn, and Dr. Mathieu Dahan for their feedback and suggestions on the work.

I am fortunate to have worked with labmates who have deeply influenced both my research and me as a person. Special thanks to Mathieu for his patient mentorship and high standards that helped shape my approach to research. Thanks to Wenbo for his active energy and creative mind, to Dahye, whose warmth and wisdom have inspired me, and to Minas for his research help and career advice.

I am also grateful to my research group members and friends - Ali, Amira, Andrew, Anthony, Arnaud, Asal, Brad, Cami, Chungjae, Connor, Enpeng, Faeze, Guancheng, Hanyu, Hongzhao, Jiawei, Tinghan, Jorge, Kevin D., Kevin W., Keyu, Meen-Sung, Mehdi, Miao, Michael B., Michael K., Rabab, Reza, Rosemarie, Rosemary, Seonho, Seyma, Shuaicheng, Sikai, Stefan, Tejas, Terrence, Thomas, Vahid, and Yunji - for creating such a welcoming and collaborative environment. They have made my PhD journey both rewarding and enjoyable. I am also thankful to my friends Hanyu, Dahye, Jie, and Miao who took care of my cat Maozai when I was away.

Finally, I am deeply grateful to my family: my beloved grandma Guofang, who has been my greatest inspiration and motivation; my always-supportive mother Liang; and my late grandpas whose memory I will always cherish. I also thank my husband Chen for his support throughout this journey, sharing both the challenges and celebrations with love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ACOPF AC optimal power flow

DCOPF DC optimal power flow

IBP Interval bound propagation

ISO Independent System Operators

LAD look-ahead dispatch

MILP mixed-integer linear programming

MIP mixed-integer programming

MISO Midcontinent Independent System Operator

OBBT optimization-based bound tightening

ReLU Rectified Linear Unit

SCED Security-constrained economic dispatch

SLAD stochastic look-ahead dispatch

SUMMARY

Security-constrained economic dispatch (SCED) optimizes generation resources to meet demand at minimal cost while maintaining grid reliability. This thesis presents novel optimization techniques for large-scale SCED problems that improve computational tractability and solution quality for modern power systems. Chapter 1 introduces a novel linear model for line loss outer approximation (LLOA) in DC optimal power flow (DCOPF), a critical component of SCED that models network constraints, providing an efficient and practical approach that balances accuracy and computational tractability for large-scale power systems. Chapter 2 develops a stochastic look-ahead dispatch (SLAD) framework for real-time energy markets, demonstrating that stochastic optimization is now computationally viable and offers greater savings compared to traditional deterministic formulations with flexible ramping products. Chapter 3 presents a new approach to neural network verification, a critical challenge in machine learning that seeks formal guarantees on neural network outputs to ensure robustness against adversarial examples and enable deployment in safety-critical applications. Chapter 4 introduces a compact formulation for optimality verification of optimization proxies for DCOPF and knapsack problems.

Chapter 1 proposes a novel linear model to capture line losses for use in linearized DC models, such as DCOPF and SCED. The *Line Loss Outer Approximation* (LLOA) model implements an outer approximation of line losses lazily and typically terminates in a small number of iterations. Experiments on large-scale power systems demonstrate the accuracy and computational efficiency of LLOA and contrast it with classical line loss approaches. The results indicate that LLOA is a practical and useful model for real-world applications, providing an effective tradeoff between accuracy, computational efficiency, and implementation simplicity. In particular, the LLOA method offers significant advantages compared to the traditional loss factor formulation for multi-period, stochastic optimization problems where good reference points may not be available. The chapter also provides a comprehen-

sive overview and evaluation of line loss approximation methods.

Chapter 2 examines how the increasing integration of intermittent renewable energy sources (RES) into power systems has created significant challenges for real-time market operations. In the U.S., real-time electricity markets are cleared every five minutes by solving a SCED, co-optimizing energy dispatch and reserve to minimize costs while meeting physical and reliability constraints. Current SCED formulations in the U.S. are deterministic and mostly consider a single time period, limiting their effectiveness in managing real-time operational uncertainty from RES intermittency. To address this limitation, U.S. markets have introduced multiple short-term ramping products to bridge the gap between deterministic and stochastic SCED formulations. Although stochastic formulations offer a unified approach to uncertainty management, their adoption has been hindered by computational costs and the availability of probabilistic forecasts. This chapter demonstrates that stochastic economic dispatch has become computationally viable for real-time market clearing. The proposed SLAD formulation, solved using an accelerated Benders' decomposition, achieves high computational efficiency on real, industry-sized transmission grids, with solutions obtained in under 5 minutes. Experimental results show that SLAD delivers more than 50% additional savings compared to flexiramp products while demonstrating greater robustness to forecasting methodology.

Chapter 3 presents a new approach to neural network verification, a critical challenge in machine learning that seeks formal guarantees on neural network outputs for ensuring robustness against adversarial examples and enabling deployment in safety-critical applications. The chapter introduces a novel mixed-integer programming (MIP) rolling-horizon decomposition method. The algorithm leverages the layered structure of neural networks by employing optimization-based bound tightening (OBBT) on smaller sub-graphs of the original network in a rolling-horizon fashion and tightening the bounds in parallel. This strategy strikes a balance between achieving tighter bounds and ensuring the tractability of the underlying MIP problems. Extensive numerical experiments, conducted on instances

from the VNN-COMP benchmark library, demonstrate that the proposed approach yields significantly improved bounds compared to existing efficient bound propagation methods. Notably, the proposed method proves effective in solving open verification problems.

Chapter 4 focuses on formal verification of optimization proxies, which are machine learning models that approximate the input-output mapping of parametric optimization problems to generate near-optimal feasible solutions. Building on recent interest in this field, the chapter addresses the critical challenge of optimality verification - determining the worst-case optimality gap over the instance distribution. To solve this challenge, the chapter develops a compact formulation for optimality verification along with a gradient-based primal heuristic, bringing substantial computational benefits compared to existing approaches. The proposed formulation extends beyond previous work by applying to non-convex optimization problems, with its benefits demonstrated through extensive testing on large-scale DCOPF and knapsack problems.

CHAPTER 1

A LINEAR OUTER APPROXIMATION OF LINE LOSSES FOR DC-BASED OPTIMAL POWER FLOW PROBLEMS

Nomenclature

Parameters

$\mathcal{N}$	Set of buses $\mathcal{N} = \{1, ..., N\}$
$\mathcal{E}$	Set of branches $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$, with $E := \|\mathcal{E}\|$
$\mathcal{E}^-$	Set of reverse branches $\mathcal{E}^- = \{(j,i) \| (i,j) \in \mathcal{E}\}$
A^{br}	Branches Incidence matrix, size $E \times N$
B	Branch susceptance matrix, size $E \times E$
r_{ij}	Resistance of branch $(i,j) \in \mathcal{E}$
$\text{T}_{ij}^{\text{max}}$	Thermal limit of branch $(i,j) \in \mathcal{E}$
P_i^{min}	Minimum active power injection at bus $i \in \mathcal{N}$
P_i^{max}	Maximum active power injection at bus $i \in \mathcal{N}$
Φ	Branch-bus PTDF matrix, size $E \times N$
$c(\cdot)$	Cost function for economic dispatch

Variables

ℓ^{tot}	Total real power losses
ℓ_{ij}	Line losses on branch $(i,j) \in \mathcal{E}$
$\overrightarrow{p_{ij}}$	Power flow from bus i to bus j on branch $(i,j) \in \mathcal{E}$

$\overleftarrow{p_{ij}}$ Power flow from bus j to bus i on branch $(i, j) \in \mathcal{E}$

p_i Active power injection at bus $i \in \mathcal{N}$

θ_i Voltage angle at bus $i \in \mathcal{N}$

v_i Voltage magnitude at bus $i \in \mathcal{N}$

1.1 Introduction

Optimal power flow optimization problems are ubiquitous in power system planning and operations, due to their wide applications in power blackouts prevention [1], power restoration [2], and power dispatch problems [3] to name only a few. The well-known AC optimal power flow (ACOPF) is often regarded as the ground truth, but its nonconvexity pose significant difficulties in large-scale deployments. The linearized DC model is often considered a reasonable approximation of ACOPF, and is especially appealing given its formulation simplicity, computational efficiency, and robustness in many situations [4]. SCED optimizations are widely used by independent system operators in real-time markets to produce active generator dispatches quickly. These models typically go beyond the pure linearized DC model and capture line losses to avoid significant errors [5, 6].

Since line losses are present during operations, it is important to study systematically how they can be captured efficiently and with high fidelity. Many existing methods require AC-feasible prior points as warm starts [7, 8, 9], which may be challenging in some settings. Line losses can also be modeled elegantly with the linear-programming method proposed in [10], but it was only examined on small test cases and its efficiency has not been satisfactorily evaluated.

This chapter systematically evaluates a number of approaches to model line losses in DC models. In addition, it presents a novel and intuitive method called *Line Loss Outer Approximation* (LLOA) in which quadratic line losses are iteratively approximated via linear constraints. Iterative approximations of line losses have been discussed before (e.g., [7, 10,

9]), but the LLOA method admits a simpler and clearer formulation that should be easier to deploy. The chapter evaluates the LLOA method within DCOPF and SCED optimizations; it can also be easily integrated within many market-clearing optimizations, including those based on MIPs. The LLOA method exhibits competitive performance on many large benchmarks, both in terms of accuracy and solving times. Being an outer approximation method, it should be ideal for multi-period, stochastic optimization problems, as well as market-clearing optimizations featuring discrete variables. This contrasts with the traditional Loss Factor formulation where a reference point is needed to seed the method. It is also interesting to mention that the LLOA method seems to bring significant improvements for more congested test cases.

This chapter is organized as follows. Section 1.2 discusses the motivation and describes the LLOA method, including the algorithms for its application to DCOPF and SCED. Section 1.3 evaluates and compares LLOA with traditional line loss approaches on a wide variety of test cases, in terms of accuracy and computational efficiency. Section 1.4 studies the recovery of AC-feasible solutions from a DC formulation. Section 1.5 concludes the chapter.

1.2 Methods For Modeling Line Losses

This section reviews the various methods to approximate line losses in linearized DC models. These linear approximations assume that all voltage magnitudes are close to 1 and that the phase angle differences are small [11, 12]. As a result, $\cos(\theta_i - \theta_j) \approx 1 - \frac{(\theta_i - \theta_j)^2}{2}$ and the losses on branch $(i, j) \in \mathcal{E}$ can be approximated by

$$\ell_{ij} = r_{ij} \times \overrightarrow{p_{ij}}^2, \tag{1.1}$$

where r_{ij} is the line resistance value and $\overrightarrow{p_{ij}}$ is the active power flow on the branch. The three methods described next differ in how Eq. (1.1) is approximated.

Throughout this section, the following conventions are used. For simplicity, and without loss of generality, the net power injection p_i at bus i captures both generation and fixed load, if any. The vector of active power injection at each bus is denoted by $\mathbf{p}$, whose i^{th} coordinate is p_i. Similarly, the vector of power flow (resp., reverse power flow) on each branch, with components $\overrightarrow{p_{ij}}$ (resp., $\overleftarrow{p_{ij}}$), is denoted by $\overrightarrow{\mathbf{p}}$ (resp. $\overleftarrow{\mathbf{p}}$). Finally, $\odot$ denotes the element-wise product between two vectors.

1.2.1 The DC-based Line Loss Loss Factor

The classical *DC-based Line Loss Loss factor* (LLLF) approach builds on a PTDF-based DC formulation. Line losses are modeled via a linearization of Eq. (1.1) around a base point, which may be the latest state estimation or a previous DC solution. Thereby, total line losses are estimated and added to the global power balance constraint [13, 14, 15], while loss factors are introduced to distribute the losses and ensure that flows are computed accurately. The resulting DC model is presented in Model 1.

Loss factors LF, loss offset ℓ^0, and loss distribution factors D are obtained as follows. First, given the reference active power injection/withdrawal p_i^{ref} at each bus i, the corresponding line flows are computed using the PTDF matrix Φ, i.e.,

$$\overrightarrow{\mathbf{p}}^{\text{ref}} = \Phi \times \mathbf{p}^{\text{ref}}. \tag{1.2}$$

Then, Eq. (1.1) is differentiated with respect to line flows, and its derivative evaluated at $\overrightarrow{\mathbf{p}}^{\text{ref}}$, yielding the nodal loss factors

$$\text{LF}^\top = -2(R \odot \overrightarrow{\mathbf{p}}^{\text{ref}})^\top \Phi. \tag{1.3}$$

The loss offset ℓ^0 is then defined as the scalar

$$\ell^0 = -\text{LF}^\top \mathbf{p}^{\text{ref}} + \sum_{(i,j)\in\mathcal{E}} r_{ij}(\overrightarrow{p_{ij}}^{\text{ref}})^2. \tag{1.4}$$

The vector D in the model distributes the losses among the buses, setting each element proportional to the losses on adjacent lines, i.e.,

$$D_i = \frac{1}{2} \times \sum_{j|(i,j)\in\mathcal{E}\cup\mathcal{E}^-} \frac{\ell_{ij}^{\text{ref}}}{\mathbf{1}^\top \ell^{\text{ref}}}. \tag{1.5}$$

Model 1 The LLLF Model.

Inputs:

ℓ^0	Loss offset
LF	Loss factors
D	Loss Distribution factors

Variables:

$\ell^{\text{tot}} \in \mathbb{R}$	Total real power losses
$\mathbf{p} \in \mathbb{R}^N$	Vector of active power injection at each bus

Model:

$$\begin{aligned}
\min \quad & c(\mathbf{p}) && \text{(1.6a)}\\
\text{s.t.} \quad & \mathbf{1}^\top \mathbf{p} = \ell^{\text{tot}} && \text{(1.6b)}\\
& \ell^{\text{tot}} = \ell^0 + \text{LF}^\top \mathbf{p} && \text{(1.6c)}\\
& -\text{T}^{\text{max}} \le \Phi\left(\mathbf{p} - \ell^{\text{tot}} D\right) \le \text{T}^{\text{max}} && \text{(1.6d)}\\
& \text{P}^{\text{min}} \le \mathbf{p} \le \text{P}^{\text{max}} && \text{(1.6e)}
\end{aligned}$$

1.2.2 Line Loss Quadratic Convex Program

Model 2 The LLQCP Model.

Inputs:

b_{ij}	Susceptance of branch $(i,j) \in \mathcal{E}$
r_{ij}	Resistance of branch $(i,j) \in \mathcal{E}$
s	Slack bus index

Variables:

θ_i	Voltage phase angle at bus $i \in \mathcal{N}$
p_i	Active power injection at bus $i \in \mathcal{N}$
$\overrightarrow{p_{ij}}$	Power flow from bus i to bus j on branch $(i,j) \in \mathcal{E}$
$\overleftarrow{p_{ij}}$	Power flow from bus j to bus i on branch $(i,j) \in \mathcal{E}$

Model:

$$\min \quad c(\mathbf{p}) \tag{1.7a}$$

$$\text{s.t.} \quad \overrightarrow{p_{ij}} = b_{ij}(\theta_j - \theta_i) \qquad \forall (i,j) \in \mathcal{E} \tag{1.7b}$$

$$p_i = \sum_{j|(i,j)\in\mathcal{E}^-} \overleftarrow{p_{ij}} + \sum_{j|(i,j)\in\mathcal{E}} \overrightarrow{p_{ij}} \qquad \forall i \in \mathcal{N} \tag{1.7c}$$

$$\overrightarrow{p_{ij}} + \overleftarrow{p_{ij}} \geq r_{ij}\overrightarrow{p_{ij}}^2 \qquad \forall (i,j) \in \mathcal{E} \tag{1.7d}$$

$$-\mathrm{T}_{ij}^{\max} \leq \overrightarrow{p_{ij}} \leq \mathrm{T}_{ij}^{\max} \qquad \forall (i,j) \in \mathcal{E} \tag{1.7e}$$

$$-\mathrm{T}_{ij}^{\max} \leq \overleftarrow{p_{ij}} \leq \mathrm{T}_{ij}^{\max} \qquad \forall (i,j) \in \mathcal{E} \tag{1.7f}$$

$$\mathrm{P}_i^{\min} \leq \mathbf{p} \leq \mathrm{P}_i^{\max} \qquad \forall i \in \mathcal{N} \tag{1.7g}$$

$$\theta_s = 0 \tag{1.7h}$$

The *Line Loss Quadratic Convex Program* (LLQCP) [16] is a DC formulation based on

phase angles. It captures the flow direction on each branch $(i, j) \in \mathcal{E}$ through two variables $\overrightarrow{p_{ij}}$ and $\overleftarrow{p_{ij}}$ and adds the following quadratic constraint on each branch:

$$\overrightarrow{p_{ij}} + \overleftarrow{p_{ij}} \geq \ell_{ij} = r_{ij}(\overrightarrow{p_{ij}})^2. \tag{1.8}$$

The formulation is specified in Model 2. Constraint (1.7c) is the nodal power balance at bus $i \in \mathcal{N}$, which states an equivalence between nodal net active power injection p_i and the sum of active power flowing in ($\overleftarrow{p_{ij}}$) and out ($\overrightarrow{p_{ij}}$) of the bus. Line losses are modeled via constraint (1.7d), and thermal limits are enforced by constraints (1.7e)-(1.7f). Other constraints are identical to the vanilla DC formulation. Figure 1.1 compares the real and estimated line losses on the French system and it highlights the fact that the LLQCP model captures the line losses with high fidelity.

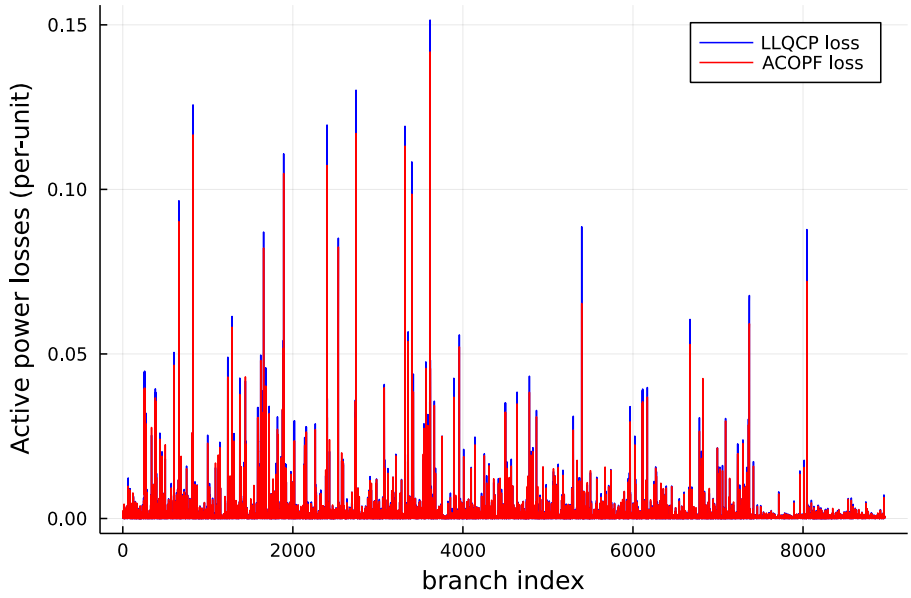


Figure 1.1: Real vs. Estimated DCOPF Line Losses on the France RTE system using LLQCP.

The main drawback of the LLQCP formulation is the nonlinearity of constraint (1.7d), which leads to increased computing times and can make the model challenging when discrete decisions are included. Thus, in the original presentation [16], the authors consider a

static linearization as illustrated in Figure 1.2. Nevertheless, computational and accuracy issues were reported on large test cases.

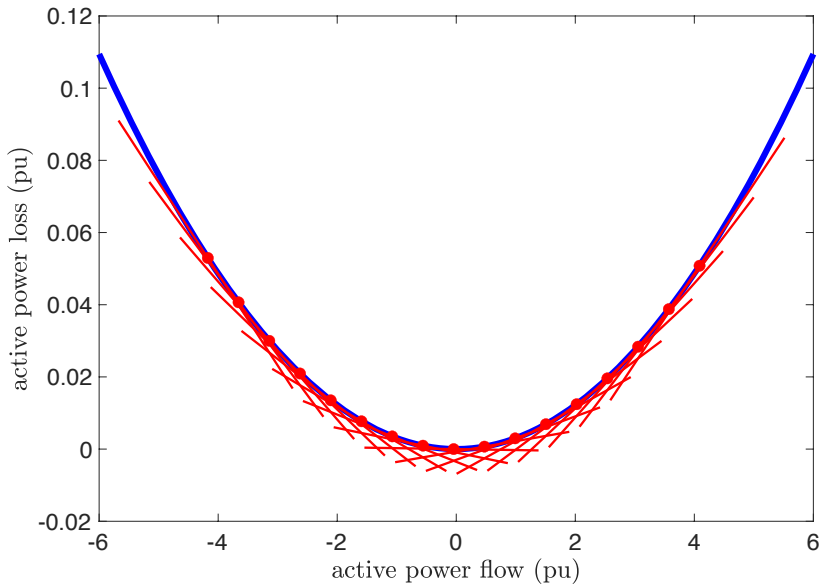


Figure 1.2: The LLQCP Linearization considered in [16]. The quadratic relation of Eq. (1.1) is depicted in **blue**, and the static linearization in **red**.

1.2.3 The Line Loss Outer Approximation

To reduce the computational burden and maintain the accuracy of the LLQCP model, the LLOA approach replaces the static linearization of the quadratic terms with a lazy outer approximation guided by a candidate solution to the model. Given a good approximation $\overrightarrow{p_{ij}}^{\text{ref}}$ of the active power flow on branch (i, j), constraint (1.7d) can be outer-approximated with the *linear* constraint

$$\overrightarrow{p_{ij}} + \overleftarrow{p_{ij}} \geq r_{ij} \left[-(\overrightarrow{p_{ij}}^{\text{ref}})^2 + 2 \times \overrightarrow{p_{ij}}^{\text{ref}} \times \overrightarrow{p_{ij}} \right]. \tag{1.9}$$

More generally, given a set of H linearization points $\mathcal{F} = \{\overrightarrow{\mathbf{p}}^1, ..., \overrightarrow{\mathbf{p}}^H\}$, the LLOA formulation includes, for each branch, H constraints of the form (1.9). The corresponding

formulation is given in Model 3: it is identical to Model 2, except for constraint (1.7d) which is replaced with outer-approximation constraints of the form (1.9).

Model 3 The LLOA Model.

Inputs:

b_{ij}	Susceptance of branch $(i,j) \in \mathcal{E}$
r_{ij}	Resistance of branch $(i,j) \in \mathcal{E}$
s	Slack bus index
$\mathcal{F}$	Set of linearization points

Variables:

θ_i	Voltage phase angle at bus $i \in \mathcal{N}$
p_i	Active power injection at bus $i \in \mathcal{N}$
$\overrightarrow{p_{ij}}$	Power flow from bus i to bus j on branch $(i,j) \in \mathcal{E}$
$\overleftarrow{p_{ij}}$	Power flow from bus j to bus i on branch $(i,j) \in \mathcal{E}$

Model:

$$
\begin{aligned}
\min \quad & c(\mathbf{p}) \\
\text{s.t.} \quad & \text{(1.7b) – (1.7c)} \\
& \text{(1.9)} && \forall (i,j) \in \mathcal{E}, \forall \overrightarrow{\mathbf{p}}^{\text{ref}} \in \mathcal{F} \\
& \text{(1.7e) – (1.7h)}
\end{aligned}
$$

The general LLOA iterative algorithm is described in Algorithm 4, in the context of DCOPF. Besides the power network data, which is omitted for brevity, the only input is a convergence tolerance $\epsilon > 0$: unlike LLLF, no reference solution is needed for correctness. At each iteration, the current approximation is solved and, if the convergence

criterion is not met, the outer approximation is refined using the obtained solution. This process is equivalent to applying Kelley's cutting-plane algorithm to the LLQCP formulation, thereby yielding, in the limit, the same accuracy as solving LLQCP. Furthermore, the LLOA model remains linear throughout the procedure, it has a smaller size compared to the static linearization used in [16], and the algorithm can be terminated at any point to produce an appropriate trade-off between efficiency and accuracy. Finally, note that, although a starting point is not required for correctness, the algorithm can be warm-started using a reference solution, e.g., a previous operating point, or the solution of the vanilla DCOPF.

Algorithm 4 The LLOA iterative algorithm.

Input: Convergence tolerance ϵ
Set $\mathcal{F} = \emptyset, \Delta^0 = +\infty, Z^0 = +\infty, k = 0$
while $|\Delta^k| > \epsilon$ **do**
    Solve current approximation using Model 3
    Retrieve corresponding dispatch $\mathbf{p}^k$ and flow $\overrightarrow{\mathbf{p}}^k$
    $Z^k = c\left(\mathbf{p}^k\right)$
    $\Delta^k = \dfrac{|Z^k - Z^{k-1}|}{|Z^{k-1}|}$
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{\overrightarrow{\mathbf{p}}^k\}$
    $k \leftarrow k + 1$
end while
Return: Optimal active power dispatch $\mathbf{p}^*$ and optimal objective value $Z^* = c(\mathbf{p}^*)$

1.2.4 Discussion

While the three methods LLLF, LLQCP, and LLOA all adjust the vanilla DC formulation to approximate line losses based on Eq. (1.1), they exhibit some fundamental differences, whose benefits and limitations are discussed next.

First, on the one hand, the LLLF method is a top-down approach: total losses are estimated using a reference point, then distributed onto the lines and buses in order to account for thermal limits and power balance. In particular, the LLLF method only differs

from a vanilla PTDF-based DC formulation by the addition the total line loss variable ℓ^{tot}, and the modification of PTDF coefficients using loss distribution factors. This approach is best-suited to low-congestion settings where few, albeit dense, PTDF constraints are needed. On the other hand, LLQCP and LLOA are bottom-up approaches, wherein losses are estimated on each branch through the convexification of Eq. (1.1), without the need for a reference point. These methods rely on sparser phase-angle formulations, at the price of a higher number of variables.

Second, the LLOA formulation can be dynamically refined to improve its accuracy as needed, which exploits the warm-starting capabilities of simplex-based LP solvers. In contrast, because the loss factor coefficients are embedded in the problem's constraint matrix, the LLLF formulation cannot be refined without re-building the problem, effectively forcing a cold-start subsequent solve. While doing so may be tractable for single-period, deterministic problems, this is no longer the case in settings where re-optimizing a model from scratch is costly, for instance, when considering multi-period and/or stochastic problems, or when discrete decisions are involved, e.g., in unit-commitment problems.

Third, in contrast to LLLF, LLQCP and LLOA do not require an initial solution for correctness. While a reasonably good reference point is typically available for solving real-time, single-period OPF problem, this is no longer the case when considering multiple time periods spanning several hours in the future, e.g., in day-ahead markets. This limitation is made worse by the fact that longer-horizon problems are also more expensive to re-solve if the loss factors in LLLF were to be updated.

Finally, while presented in the context of OPF problems, all methods can obviously be applied to the SCED [17, 18]; note that the SCED formulation used by Midcontinent Independent System Operator (MISO) in its real-time market is based on LLLF. The inclusion of LLQCP or LLOA formulations only impacts the representation of line flows, namely, using a phase angle-based formulation instead of the PTDF approach used by MISO. Once this formulation is available, the line-loss constraints are similar to those of the DC model

just presented, and the rest of the model is unchanged.

1.3 Computational Experiments

This section compares the above formulations on a collection of large-scale test cases. All models are formulated in Julia 1.6 using JuMP [19]; all DC-based formulations are solved using Gurobi 9.1.0 [20] with default parameters, and all AC models are solved with Ipopt. Experiments are carried out on Linux machines with dual Intel Xeon 6226@2.7GHz CPUs on the PACE Phoenix cluster [21]. Section 1.3.1 presents the test cases. Section 1.3.2 studies the trade-offs between LLOA and LLQCP, and Section 1.3.3 compares the accuracy of the considered DC-based approaches against an AC formulation. Computing times are reported in Section 1.3.4

1.3.1 Test cases

Experiments are carried out on a collection of large-scale instances from PGLib [22], as well as three instances provided by the French transmission system operator RTE. The size of the test cases are given in Table 1.1, which reports the number of buses, branches, and generators in each system. The three test cases from RTE are the following: `MSR` is a snapshot of a south-east region in France known to suffer from voltage issues, `LYON` is the French high-voltage system plus low- and medium-voltage grid components in the Lyon area, and `France` is the full French transmission grid.

For each system, both a vanilla OPF and MISO's real-time SCED formulations are considered; the latter is described in [17, 18]. The SCED formulation mainly differs from OPF in the inclusion of reserve requirements and contingency constraints. In the current formulation used by MISO, line losses are modeled via loss factors, as in LLLF, and reserve requirements, power balance, and transmission constraints are soft, i.e., they can be violated at a (high) penalty. The AC-SCED formulation considers the same modeling of reserve requirements, but uses explicit AC power flow equations. Consequently, in order to keep

the AC-SCED formulation tractable, only a single generator contingency is considered, corresponding to the loss of the largest generator. Finally, all models are expressed in per-unit, with a base scaling factor of 100MW (resp. 100MVAR) for active (resp. reactive) power.

All DC-based formulations are compared to the AC formulation using the following metrics: the percent objective gap with respect to the AC solution found by Ipopt, the total estimated line losses, and the Mean Absolute Error (MAE) of active power dispatch. Note that, because DC-based formulations are *approximations*, the objective gap may be positive (higher than AC) or negative (lower than AC). Letting $\mathbf{pg}^*$ and $\mathbf{pg}^*_{AC}$ denote the vectors of active power dispatch for the considered DC-based and AC formulation, respectively, the MAE is given by

$$\text{MAE} = \frac{1}{G} \left\| \mathbf{pg}^* - \mathbf{pg}^*_{AC} \right\|_1 ,$$

where G is the number of generators.

Table 1.1: Test Cases from PGLib and RTE

Source	System	Buses	Branches	Generators
PGLib	`1354_pegase`	1354	1991	260
	`2869_pegase`	2869	4582	510
	`4661_sdet`	4661	5997	1176
	`6468_rte`	6468	9000	1295
	`9241_pegase`	9241	16049	1445
	`13659_pegase`	13659	20467	4092
	`19402_goc`	19402	34704	971
RTE	`MSR`	403	550	115
	`LYON`	3411	4499	771
	`France`	6705	8962	1708

1.3.2 Tradeoff between LLOA and LLQCP

The convergence of Kelley's cutting-plane algorithm in LLOA is illustrated in Figure 1.3 for the `2869_pegase` system, with a strict convergence tolerance $\epsilon = 10^{-6}$. Values obtained for the LLQCP solution are also displayed as a reference. The behavior observed in Figure 1.3 is representative of LLOA's behavior across the overall testset. Indeed, across all considered instances, LLOA always reaches the default $\epsilon = 10^{-3}$ tolerance in no more than 3 iterations. As can be seen in Figure 1.3, further iterations do not improve the approximation significantly.

Note that, if no initial reference point is provided, then the first iteration (iteration 0 in Figure 1.3) is equivalent to solving a vanilla DC formulation, with no line losses. To enable a fair comparison to LLLF, which requires a reference solution, in all that follows, LLOA is seeded with the same initial solution as LLLF, obtained by solving the DC formulation with no losses. Accordingly, the number of iterations required by LLOA to converge is typically 1 and never more than 2. Therefore, subsequent experiments use a single LLOA iteration, and the corresponding method is denoted by LLOA_1.

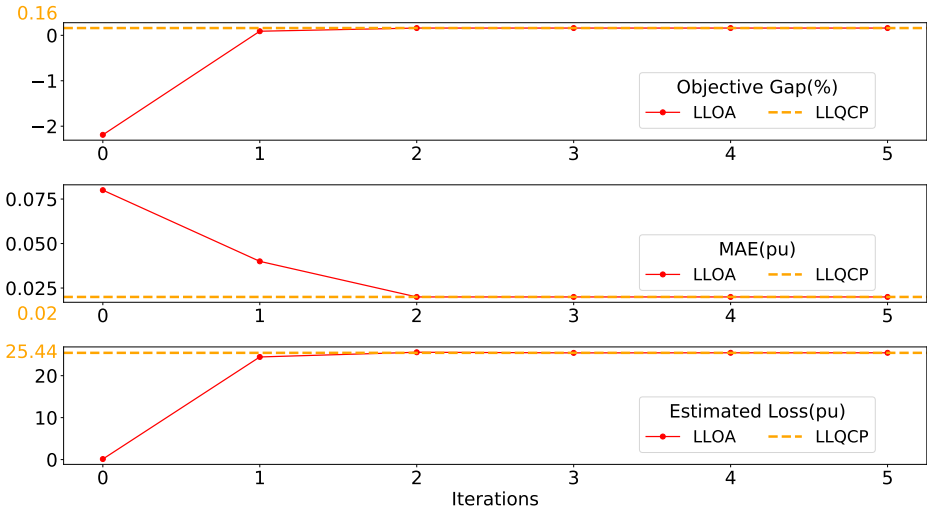


Figure 1.3: Convergence of LLOA on the `2869_pegase` system.

1.3.3 Accuracy Comparison

Table 1.2: Comparison of DC-based formulations against AC solution (OPF): objective gap, Mean Absolute Error (MAE) of active power dispatch, and total estimated line losses. Best performing method for each metric and system is shown in bold.

	Obj Gap (%)				MAE (pu)				Estimated Loss (pu)			
System	DC	$LLOA_1$	LLQCP	LLLF	DC	$LLOA_1$	LLQCP	LLLF	$LLOA_1$	LLQCP	LLLF	AC
1354_pegase	-3.23	+0.26	+0.30	**+0.09**	0.08	0.02	**0.02**	0.04	15.99	16.19	**15.46**	15.03
2869_pegase	-3.11	-0.32	**+0.24**	-0.50	0.13	0.19	**0.02**	0.16	19.82	**29.67**	18.61	27.02
4661_sdet	-1.56	+0.05	+0.09	**-0.04**	0.04	0.04	**0.03**	0.04	11.70	**12.29**	11.34	12.20
6468_rte	-4.20	-0.46	**-0.45**	-0.82	0.10	0.05	**0.05**	0.06	20.42	20.55	**18.60**	19.45
9241_pegase	-3.46	**-0.21**	+0.21	-0.42	0.12	0.14	**0.02**	0.13	55.87	**77.06**	53.88	72.26
13659_pegase	-2.07	**-0.07**	+0.08	+0.13	0.04	0.03	**0.01**	0.04	73.88	**90.73**	80.93	88.71
19402_goc	-4.04	**-0.02**	+0.03	+0.06	0.04	0.00	**0.00**	0.00	35.68	**36.29**	36.34	36.18
MSR	-0.14	+0.01	+0.01	**+0.01**	0.00	0.00	**0.00**	0.00	0.50	0.50	**0.50**	0.47
LYON	-1.79	**+0.85**	+0.90	+1.04	0.04	**0.01**	0.01	0.01	12.73	12.89	**10.50**	10.10
France	-3.64	-0.18	**-0.15**	-0.21	0.01	0.01	**0.00**	0.01	**12.77**	13.67	12.60	13.14

Table 1.2 reports, for each method and test system, the percent objective gap, the MAE of active power dispatch, and the total estimated line losses, for the OPF problems. The same statistics are reported for SCED problems in Table 1.3. In each table, the best-performing method is indicated in bold. Because the vanilla DC formulation (denoted by DC in the tables) does not model line losses, it is removed from Tables 1.2 and 1.3 which, instead, report total line losses in the AC solution. The results are consistent across OPF and SCED problems, and motivate the following comments.

Table 1.3: Comparison of DC-based formulations against AC solution (SCED): objective gap, Mean Absolute Error (MAE) of active power dispatch, and total estimated line losses. Best performing method for each metric and system is shown in bold.

	Obj Gap (%)				MAE (pu)				Estimated Loss (pu)			
System	DC	$LLOA_1$	LLQCP	LLLF	DC	$LLOA_1$	LLQCP	LLLF	$LLOA_1$	LLQCP	LLLF	AC
1354_pegase	-2.31	+0.28	+0.29	**+0.15**	0.07	0.03	0.03	**0.02**	15.47	15.53	**14.78**	14.51
2869_pegase	-2.18	**+0.02**	+0.16	-0.13	0.09	0.07	**0.02**	0.06	**22.77**	25.44	21.04	23.69
4661_sdet	-1.50	+0.07	+0.09	**-0.01**	0.03	0.03	**0.03**	0.03	11.65	**12.06**	11.06	11.98
6468_rte	-3.77	-0.16	**-0.12**	-0.49	0.12	0.08	**0.07**	0.08	**19.43**	19.89	17.66	18.92
9241_pegase	-3.05	-0.18	**+0.12**	-0.26	0.13	0.11	**0.05**	0.11	55.90	**74.39**	54.14	71.54
13659_pegase	-2.03	-0.14	**+0.06**	+0.07	0.04	0.04	**0.01**	0.05	68.11	**89.58**	74.39	87.77
19402_goc	-3.91	-0.09	-0.04	**-0.02**	0.04	0.01	**0.01**	0.01	36.66	**37.41**	37.28	37.38
MSR	-4.58	+0.21	+0.25	**+0.16**	0.01	0.00	**0.00**	0.00	0.46	0.46	**0.45**	0.44
LYON	-1.80	**+0.81**	+0.83	+1.05	0.02	0.02	**0.01**	0.03	13.79	13.90	**10.10**	10.46
France	-3.55	-0.19	**-0.15**	-0.24	0.01	0.01	**0.00**	0.01	12.29	**12.76**	12.15	12.59

First, LLLF, LLQCP, and $LLOA_1$ all outperform the vanilla DC formulation in terms of objective gap, MAE, and total estimated losses. Indeed, because it ignores line losses, the vanilla DC formulation is found to systematically under-estimate the total objective cost, by as much as 4.20% on OPF and 4.58% on SCED problems. Furthermore, ignoring line losses results in lower total active power generation which, in unit-commitment problems, may lead to committing too few units, thereby causing price spikes and/or reliability issues later in the day. Since all electricity markets in the US are based on DC formulations, this observation reinforces the fact that line loss information should be included in TSO market specifications.

Second, among LLLF, LLQCP, and $LLOA_1$, no individual method dominates the others across all considered instances. Nevertheless, the LLQCP method is found to be the most accurate overall, with typically smaller objective gaps and MAE, as well as better line loss estimation. This demonstrates the benefits of exploiting nonlinear information in the formulation although, as will be shown next, this comes at the price of higher computing times. In addition, it establishes the accuracy that LLOA can reach since, in the limit, it

solves the same formulation as LLQCP.

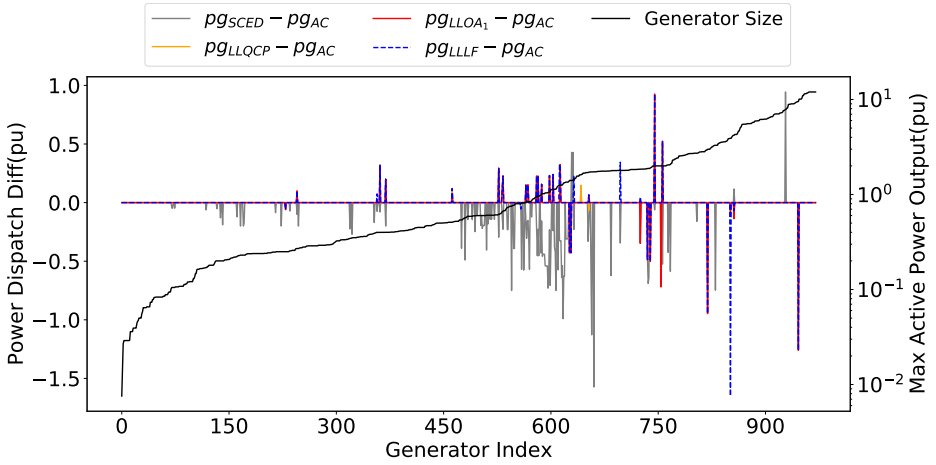


Figure 1.4: Differences in active power dispatch (left axis, in p.u.) between DC-based and AC SCED formulations, on the `19402_goc` system. Generators are ordered from left to right by increasing maximum capacity (right axis, in p.u.).

In light of the above observations, it is interesting to look at each method's behavior at the individual generator level. To that end, Figure 1.4 plots the difference in active power dispatch between each DC-based method, and the reference AC solution, for the SCED problem on the `19402_goc` system. Individual differences are depicted (left axis), together with the maximum capacity of each generator (right axis, in log-scale).

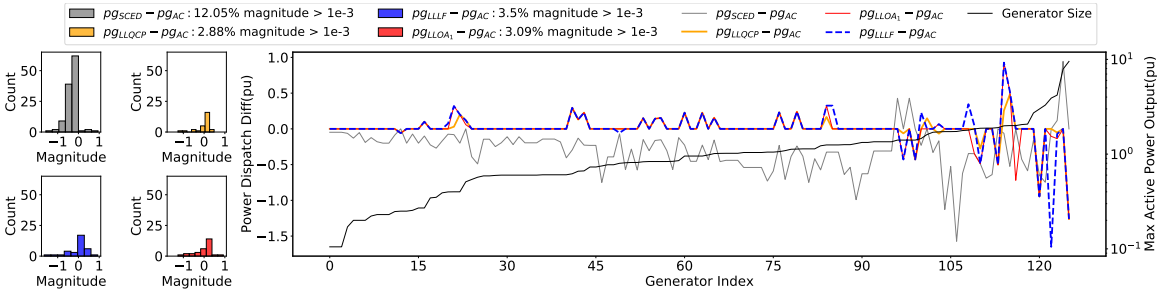


Figure 1.5: Histogram (left) and individual differences (right) of active power dispatch (left axis, in p.u.) between DC-based and AC SCED formulations, on the `19402_goc` system. Only generators for which at least one method yields an error $>10^{-3}$p.u. are displayed, in increasing order of maximum capacity (right axis, in p.u.).

Figure 1.5 presents the same data after filtering out generators for with small (<0.1MW) differences, as well as the corresponding histograms of active power dispatch differences,

for each DC-based method. Unsurprisingly, the generator dispatch in the vanilla DC solution are usually lower than in the AC solution. Moreover, with about 12% of generators with non-zero difference, the vanilla DC solution differs from the AC solution significantly more than LLLF, LLQCP, and LLOA_1, which display both fewer and smaller differences. Remarkably, the LLLF, LLQCP, and LLOA_1 solutions are very similar, with about 97% of generators having an almost identical dispatch to the AC solution. This latter behavior is highly desirable, since it means that operators need only adjust a handful of generators to recover an AC-feasible operating point; this question is further investigated in Section 1.4.

1.3.4 Computational Efficiency

Tables 1.4 and 1.5 report the computing time of each method on OPF and SCED problems.

Table 1.4: Computing times, in seconds (OPF).

Solving time (secs)	DC	LLOA_1	LLQCP	LLLF	AC
1354_pegase	0.03	0.31	0.29	0.03	25.71
2869_pegase	0.14	1.01	0.83	0.05	41.48
4661_sdet	1.75	2.06	1.04	1.04	47.28
6468_rte	0.02	3.41	2.75	0.02	161.91
9241_pegase	0.60	5.10	5.42	0.34	106.56
13659_pegase	2.24	6.83	6.74	0.99	141.20
19402_goc	0.04	4.34	20.49	0.03	346.67
MSR	0.00	0.03	0.10	0.00	20.08
LYON	0.03	1.03	0.88	0.02	40.55
France	0.09	3.85	2.47	0.04	185.13

The reported times are in seconds, and do not include model setup time, which is negligible, nor the computation of PTDF coefficients, which can be performed offline as it only depends on the network topology. Finally, computing times for LLLF and LLOA_1

do not include the time needed to obtain the initial reference solution since, in real-time operations, the previous solution or current state estimation is readily available.

Table 1.5: Computing times, in seconds (SCED).

Solving time (secs)	DC	$LLOA_1$	LLQCP	LLLF	AC
1354_pegase	0.02	0.43	1.38	0.03	55.24
2869_pegase	0.09	1.24	5.35	0.08	121.96
4661_sdet	0.85	7.85	79.56	0.85	124.43
6468_rte	0.06	3.24	40.07	0.07	540.92
9241_pegase	0.54	6.70	52.98	0.53	675.15
13659_pegase	3.70	15.04	467.52	2.26	782.49
19402_goc	0.19	26.98	272.10	0.24	1247.80
MSR	0.00	0.07	0.13	0.01	31.83
LYON	0.12	1.94	7.13	0.11	137.24
France	0.20	4.89	18.51	0.50	996.21

First, computing times for SCED are consistently higher than for OPF. This is due to (i) the SCED formulation comprising more variables and constraints than OPF, and (ii) the presence of post-contingency constraints that are added lazily, thereby requiring several re-optimizations. The latter is especially detrimental to LLQCP: the presence of nonlinear constraints forces the use of Gurobi's barrier algorithm, which cannot be warm-started. Indeed, although LLQCP is surprisingly fast on the simpler OPF problems, its computing time for SCED increases by as much as 70x on the 13659_pegase system. In contrast, linear formulations like LLLF and $LLOA_1$ make use of the simplex algorithm's warm-starting capabilities, and therefore see a lesser performance degradation between OPF and SCED.

Second, for both OPF and SCED problems, LLLF is the fastest method. Namely, on SCED problems, LLLF outperforms LLQCP, by up to two orders of magnitude, and $LLOA_1$

by about one order of magnitude on average. Indeed, as mentioned in Section 1.2.4, thermal constraints in LLLF are added lazily, whereas LLQCP and LLOA require the full set of transmission flow variables and constraints. Because, in practice, only a subset of thermal limits are binding, the LLLF formulation remains compact and requires few iterations to converge. Nevertheless, all SCED instances are solved by LLOA_1 within 30 seconds, fast enough to meet the requirements of real-time operations.

Third, the performance of LLLF is not always consistent across the test cases. For instance, LLLF takes 0.85s to solve the `4661_sdet` system, but is 12x faster to solve the larger `6468_rte` system. In contrast, computing time for LLOA scales roughly linearly with the system size. A deeper analysis of this behavior reveals that the performance of LLLF is directly impacted by how congested a system is. Again, this is explained by the PTDF-based formulation used by LLLF: congested systems have more binding limits, which leads to denser formulations and slows down convergence. On the other hand, LLOA and LLQCP include all transmission constraints by design, and are therefore less impacted by congestion.

1.3.5 Sensitivity analysis

Motivated by the above remarks, further experiments are conducted to assess the robustness of each method to changes in total load. Namely, additional instances are generated by perturbing each load with a multiplicative noise ξ, which follows a log-normal distribution with mean α and standard deviation 0.05. In the present setting, the parameter α, referred to as *load scaling factor*, takes values between 0.9 and 1.1 in 0.02 increments, corresponding to mean variations in total load that range between -10% and $+10\%$ of the reference load. For each value of α, noise is sampled using ten different random seeds, which yields an extended test set of 1100 OPF and 1100 SCED instances.

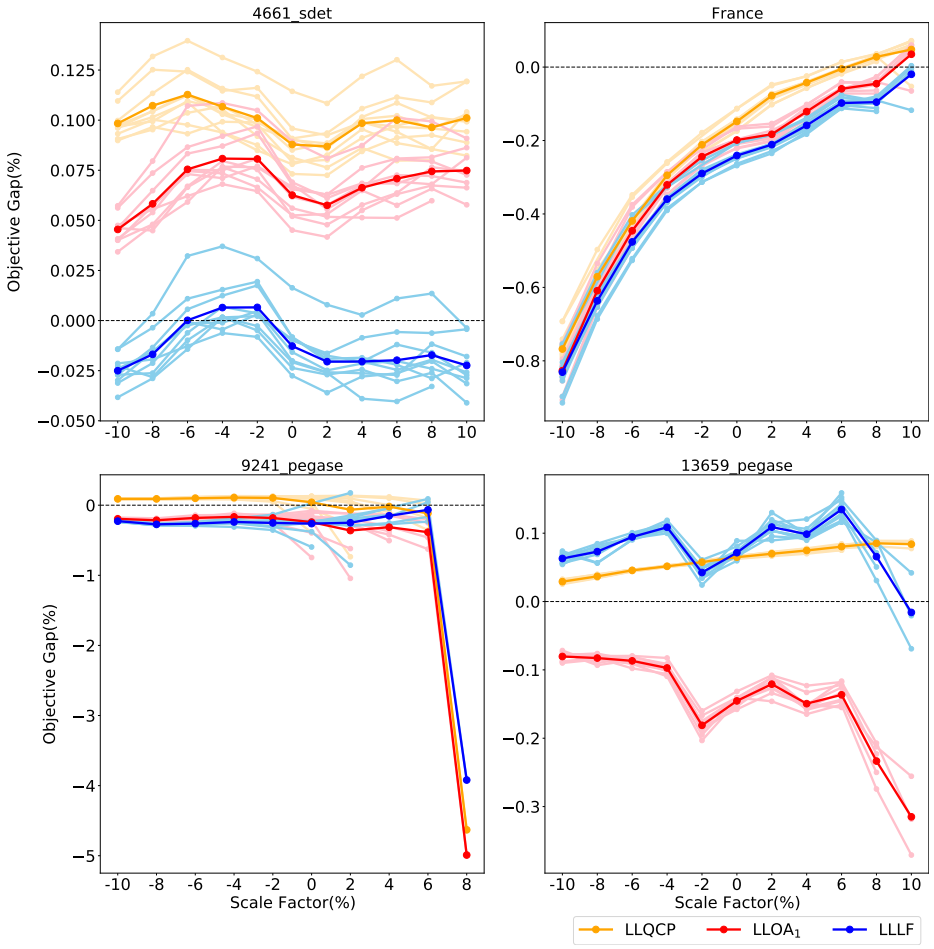


Figure 1.6: Behavior of the percent objective gap with respect to variations in load (SCED).

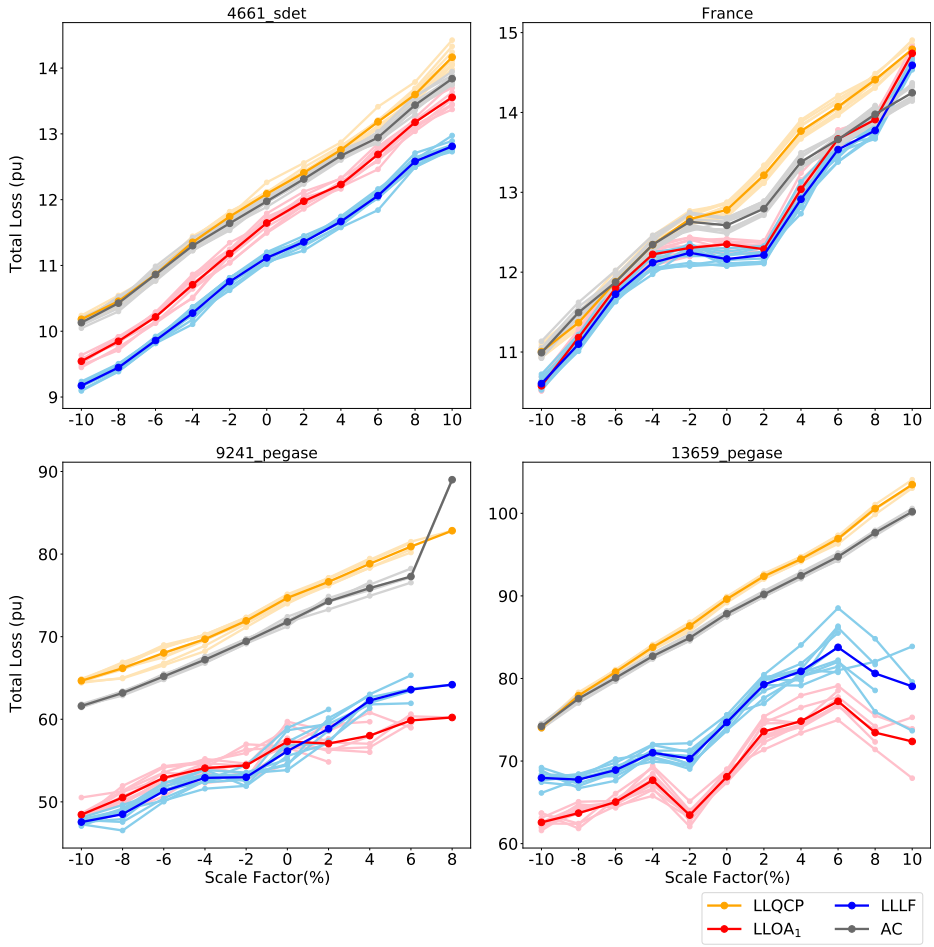


Figure 1.7: Behavior of the total estimated losses with respect to variations in load (SCED)

Numerical results for SCED are illustrated in Figures 1.6-1.9, which depict the behavior of LLLF, LLQCP, and LLOA in terms of objective gap, total losses, MAE and computing time, respectively.

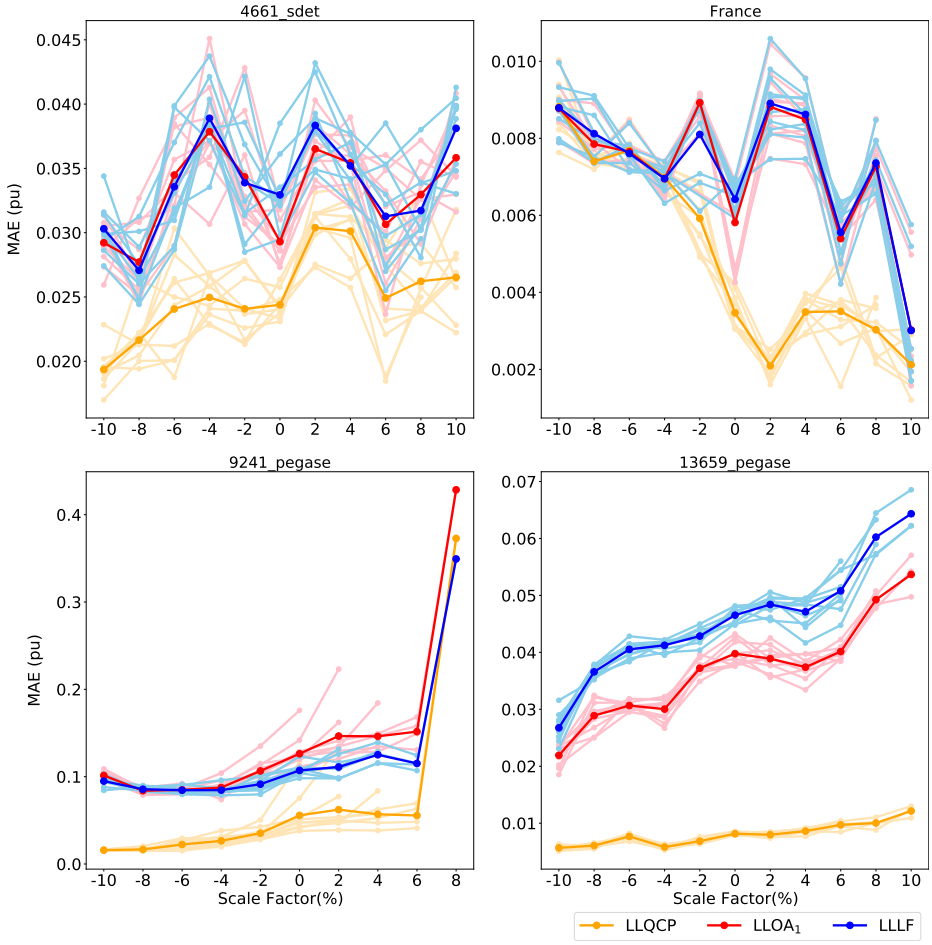


Figure 1.8: Behavior of the MAE with respect to variations in load (SCED).

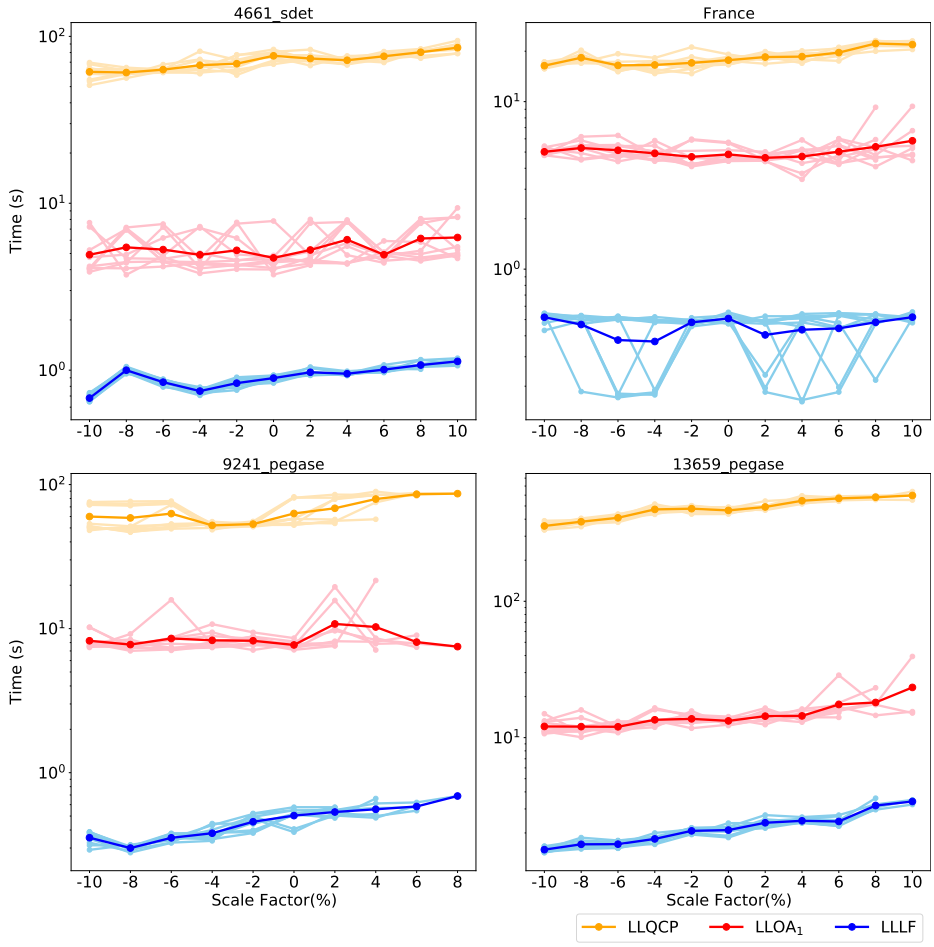


Figure 1.9: Behavior of the computing time with respect to variations in load (SCED).

Each figure depicts results for individual seeds (in lighter tone), as well as averages across the 10 seeds. Instances found to be DC-infeasible, or for which Ipopt failed to find any AC-feasible solution have been factored out of the plots. For brevity, four representative test systems are selected: `4661_sdet`, `France`, `9241_pegase` and `13659_pegase`; complete results for OPF and SCED are reported in the appendix.

Overall, the results are consistent with those of Section 1.3.3. Namely, no individual method is uniformly better than the other two in terms of objective gap, LLQCP is typically more accurate in terms of MAE and total estimated losses, and LLLF displays lower computing times than LLOA, which itself outperforms LLQCP by about an order of magnitude. Note that the sudden changes in objective gap and MAE observed on `9241_pegase`, as

total load is increased by 8%, are caused by a phase-transition behavior of the AC formulation, which was studied in [23].

Figures 1.6-1.9 also highlight different behaviors as the load changes, depending on both the method and the test case. First, the behavior of LLQCP is generally smoother than LLOA and LLLF, especially on the `13659_pegase` system. This can be attributed to LLQCP being a nonlinear formulation, which is typically less sensitive to small changes in input data than linear formulations like LLOA and LLLF. Second, although LLLF is the fastest method, its computational efficiency is more severely affected by increases in total load than LLOA and LLQCP. Indeed, on SCED instances, compared to the nominal setting, LLLF is roughly 20% faster when the load is decreased by 10%, and 20% slower when the load is increased by 10%; the worst slowdown is observed on `4661_sdet`, wherein instances with the highest load are on average 2.10x slower than in the nominal case. In contrast performance variations for LLOA are, on average, twice smaller than LLLF. As discussed in Section 1.2.4, because LLLF is based on PTDF, higher congestion has an adverse impact on its performance.

Finally, LLOA and LLLF often behave similarly to each other in terms of objective gap, MAE and total estimated losses, most likely because they use the same reference point for estimating line losses. This observation holds for most of the considered test cases, OPF and SCED test cases. Nevertheless, a few instances show the two methods diverging as total load is increased. To analyze this behavior in more detail, Figure 1.10 reports the objective gap for each method on the `LYON` SCED; results from only 3 seeds are depicted for readability. While LLOA and LLQCP are almost indistinguishable, with objective gap that remains low, the objective gap of LLLF increases to over 12%, meaning the objective value of LLLF is significantly higher than LLOA and LLQCP. The observed behavior remains even when using the AC solution as a reference point for computing the loss factors. This behaviour is not an exception on the `LYON` system, we observed similar pattern on `500_goc` as well. A deeper analysis reveals that the change in objective

value is caused by several thermal violations in LLLF, which trigger higher reserve levels and transmission violation penalties; the same set transmission lines is at fault across the different seeds. Furthermore, although the thermal limits are not violated in either LLQCP, LLOA, or AC formulations, taking the corresponding solutions and evaluating the flows as per Eq. (1.6d) systematically yields violations on those branches. This demonstrates the limitation of LLLF in estimating line flows accurately, and that such inaccuracies can result in significantly higher operating costs.

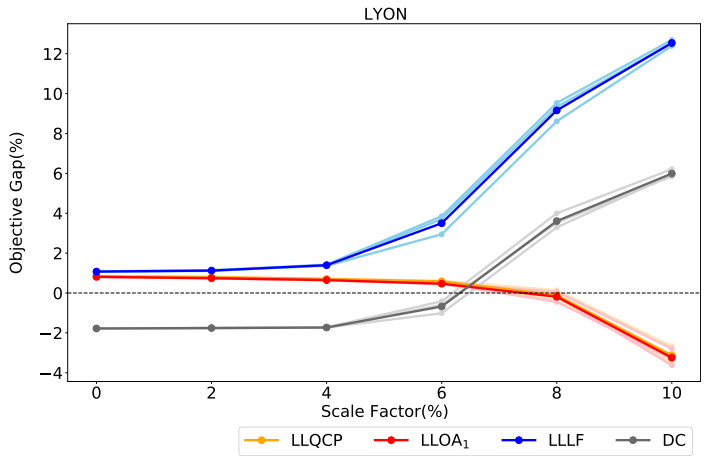


Figure 1.10: Objective Gap after Load Scaling for `LYON`, with scale factors ranging from 0% to 10% and three different seeds.

1.4 AC Feasibility Restoration

While electricity markets are based on DC formulations, operators must ensure that final dispatch instructions are AC-feasible. To that end, the market solution seeds a power flow analysis to determine voltage levels and active/reactive power dispatch for each generator. Whenever the power flow engine is unable to recover a feasible AC operating point, operators must manually override some of the market decisions, which typically leads to increased production costs.

This process is replicated by seeding a power flow engine with the solution of DC, LLQCP, LLOA, and LLLF methods. Although the power flow engine does not enforce operating and thermal limits, violations in reactive power capacity are alleviated –but not

always eliminated– via PV-PQ bus switching. Recall that reserve dispatch (in SCED) and total operating costs only depend on active power dispatch which, by definition, the power flow engine does not modify except for the generator located at the slack bus. Therefore, the objective value will not be changed much by the power flow and, accordingly, the rest of the analysis will focus on AC feasibility.

Table 1.6: Constraint violation statistics for active and reactive capacity (OPF). Violations are in per-unit.

		Active Power		Reactive Power	
Test case	Method	#viol.	Max	#viol.	Max
1354_pegase	DC	1	19.21	0	0.00
	LLQCP	1	1.62	0	0.00
	$LLOA_1$	1	1.80	0	0.00
	LLLF	1	2.42	0	0.00
2869_pegase	DC	1	39.69	0	0.00
	LLQCP	1	3.06	0	0.00
	$LLOA_1$	1	13.56	0	0.00
	LLLF	1	14.46	0	0.00
4661_sdet	DC	0	0.00	0	0.00
	LLQCP	0	0.00	0	0.00
	$LLOA_1$	0	0.00	0	0.00
	LLLF	0	0.00	0	0.00
MSR	DC	0	0.00	2	1.86
	LLQCP	0	0.00	1	1.87
	$LLOA_1$	0	0.00	1	1.87
	LLLF	0	0.00	1	1.87
LYON	DC	1	12.14	6	1.13
	LLQCP	0	0.00	6	0.83
	$LLOA_1$	0	0.00	6	0.83
	LLLF	1	0.65	6	0.83
France	DC	1	14.95	5	1.56
	LLQCP	1	0.72	4	1.56
	$LLOA_1$	1	1.00	5	1.56
	LLLF	1	1.19	5	1.55

Table 1.7: Constraint violation statistics for voltage magnitude and thermal limits (OPF). Violations are in per-unit.

		Voltage magnitude		Thermal	
Test case	Method	#viol.	Max	#viol.	Max
1354_pegase	DC	1	0.01	23	6.05
	LLQCP	0	0.00	21	4.06
	$LLOA_1$	0	0.00	21	4.05
	LLLF	0	0.00	22	4.35
2869_pegase	LLQCP	90	0.04	33	5.70
	$LLOA_1$	49	0.03	32	5.64
	LLLF	115	0.05	34	6.52
4661_sdet	DC	9	0.02	196	5.30
	LLQCP	8	0.02	180	1.83
	$LLOA_1$	8	0.02	177	2.13
	LLLF	8	0.02	190	2.24
MSR	DC	36	0.02	0	0.00
	LLQCP	35	0.02	0	0.00
	$LLOA_1$	35	0.02	0	0.00
	LLLF	35	0.02	0	0.00
LYON	DC	542	0.06	7	13.07
	LLQCP	538	0.06	1	0.11
	$LLOA_1$	539	0.06	1	0.11
	LLLF	538	0.06	3	0.12
France	DC	413	0.05	10	18.37
	LLQCP	409	0.05	9	0.18
	$LLOA_1$	403	0.05	9	0.18
	LLLF	404	0.05	9	0.21

To that end, Tables 1.6-1.7 report, for the OPF instances, the number (#viol) and largest magnitude (Max) of constraint violations for the power-power flow solutions, namely, ac-

tive/reactive power limits, voltage limits, and thermal limits; results for SCED instances are in Tables 1.8-1.9. All results are obtained in the nominal setting, i.e., without any load perturbation, and test cases for which the power flow did not converge are factored out of the tables.

Table 1.8: Active and reactive capacity violation statistics (SCED). Constraint violations are in per-unit.

		Active Power		Reactive Power	
Test case	Method	#viol.	Max	#viol.	Max
1354_pegase	DC	1	18.15	0	0.00
	LLQCP	1	1.63	0	0.00
	$LLOA_1$	1	1.68	0	0.00
	LLLF	1	2.29	0	0.00
2869_pegase	DC	1	35.79	0	0.00
	LLQCP	1	2.75	0	0.00
	$LLOA_1$	1	5.13	0	0.00
	LLLF	1	6.89	0	0.00
4661_sdet	DC	0	0.00	0	0.00
	LLQCP	0	0.00	0	0.00
	$LLOA_1$	0	0.00	0	0.00
	LLLF	0	0.00	0	0.00
MSR	DC	0	0.00	7	1.43
	LLQCP	0	0.00	7	1.45
	$LLOA_1$	0	0.00	7	1.45
	LLLF	0	0.00	7	1.45
LYON	DC	1	11.94	19	1.42
	LLQCP	0	0.00	17	1.42
	$LLOA_1$	0	0.00	17	1.42
	LLLF	1	0.75	19	1.44
France	DC	1	14.28	19	1.28
	LLQCP	1	0.35	20	1.28
	$LLOA_1$	1	0.60	21	1.28
	LLLF	1	0.77	21	1.28

Table 1.9: Voltage magnitude and thermal violation statistics (SCED). Constraint violations are in per-unit.

		Voltage magnitude		Thermal	
Test case	Method	#viol.	Max	#viol.	Max
1354_pegase	DC	1	0.02	22	5.32
	LLQCP	1	0.01	19	3.41
	$LLOA_1$	1	0.01	19	3.41
	LLLF	1	0.01	18	3.39
2869_pegase	DC	73	0.09	53	9.72
	LLQCP	18	0.02	35	5.42
	$LLOA_1$	18	0.02	33	5.34
	LLLF	16	0.02	34	5.71
4661_sdet	DC	9	0.02	176	5.15
	LLQCP	8	0.02	166	1.22
	$LLOA_1$	8	0.02	170	1.22
	LLLF	8	0.02	177	1.42
MSR	DC	70	0.02	0	0.00
	LLQCP	70	0.02	0	0.00
	$LLOA_1$	70	0.02	0	0.00
	LLLF	70	0.02	0	0.00
LYON	DC	573	0.06	5	12.63
	LLQCP	572	0.06	4	0.20
	$LLOA_1$	572	0.06	4	0.20
	LLLF	566	0.06	3	0.14
France	DC	479	0.05	6	17.26
	LLQCP	467	0.05	6	0.17
	$LLOA_1$	467	0.05	6	0.17
	LLLF	467	0.05	7	0.20

Results are consistent across OPF and SCED instances. Vanilla DC leads to the highest

violations after power flow, especially regarding active power capacity and thermal limits. The former is due to the slack generator having to compensate for all losses, which are ignored in vanilla DC yet often exceed that generator's maximum output. Similarly, large thermal limit violations are found to systematically occur on lines adjacent to the slack bus. In contrast, LLLF, LLQCP, and LLOA exhibit similar behavior, with fewer and smaller violations than vanilla DC, although some violations still occur. In particular, active power capacity violations are directly correlated to the accuracy of total estimated losses: the more accurate the estimated losses, the smaller the violation. The use of a more robust power flow engine –which is beyond the scope of this chapter– would likely reduce overall violations in reactive power capacity. Nevertheless, the observed violations in reactive power and voltage magnitude highlight the limitations of DC-based formulations, which ignore reactive power and assume voltage magnitude to be 1 per-unit.

1.5 Conclusion

This chapter presents LLOA, a novel and simple linear model to capture line losses. The essences of LLOA formulation are that it not only provides an intuitive and clean outer approximation to the LLQCP, which can be efficiently solved using linear programming approaches, but it also can be easily integrated within different formulations, including multi-period and stochastic formulations as well as market-clearing optimizations with discrete decisions. Extensive numerical results comparing LLOA, LLQCP, and LLLF also demonstrate the competitive performance of LLOA against traditional methods such as LLLF, illustrating its practical usefulness.

Phase-angle formulations such as LLQCP and LLOA are less impacted by congestion than LLLF, whose performance deteriorates as the load increases. They are also less prone to inaccuracies in line flows which, when they occur on congested lines, can lead to large increases in operating costs. Finally, LLOA shows no drop in the ability to recover AC-feasible solutions through a power flow analysis, compared to LLQCP or LLLF. Further

numerical studies and performance benchmarks, in settings involving discrete decisions, multiple time periods or stochastic formulations, will be the topic of future work.

CHAPTER 2
ON THE VIABILITY OF STOCHASTIC ECONOMIC DISPATCH FOR REAL-TIME ENERGY MARKET CLEARING

2.1 Introduction

The SCED is a fundamental optimization problem in power systems operations. In the US, Independent System Operators (ISO) clear the real-time electricity market by solving a SCED every few minutes. The SCED outputs energy and reserve dispatch for each generator, as well as energy and reserve prices [24].

Over the past few years, the growing penetration of renewable generation and distributed energy resources (DERs) has caused an increase in operational uncertainty and short-term flexibility requirements. Inadequate flexibility leads to price spikes, expensive emergency actions, and, in the worst case, load shedding events. As a result, ISOs introduced flexible ramping products –also known as *flexiramp*– in electricity markets, with the goal of mitigating short-term uncertainty. For instance, MISO implemented a 10-minute up/down ramping product in 2016 [25], and introduced a 30-minute ramping product in December 2021 [26, 27]. Similarly, the California ISO (CAISO) market includes a 5-minute ramping product [28].

While it provides reliability benefits, *flexiramp* is outperformed by stochastic optimization models, as convincingly demonstrated in [29]. SLAD considers the uncertainty of future events and seeks to find the optimal dispatch decisions that maximize the system performance, such as minimizing costs and maximizing reliability. Nevertheless, no US ISO currently uses stochastic optimization in production, in a large part because of the increased computational cost [29]. Furthermore, to the best of the authors' knowledge, *there has been no study that quantifies the potential benefits of stochastic economic dispatch for-*

mulations on real, industrial systems compared to existing deterministic approaches with ramping products.

This chapter addresses these gaps by proposing SLAD as a stochastic extension of existing deterministic formulations for real-time electricity markets. The benefits of SLAD are demonstrated on a realistic, large-scale power system, through simulations that replicate the operations of an ISO. Moreover, the chapter presents a comprehensive comparison of several real-time dispatch approaches, ranging from simple myopic economic dispatch, to multi-period and stochastic dispatches.

Contributions and outline

Building upon the challenges identified by Hobbs, this study addresses the management of increasingly volatile net loads driven by the growth in renewable energy. Utilizing stochastic look-ahead dispatch SLAD, the chapter explores the benefits of multi-period formulations and ramping products within a practical, industrial context. The chapter's key contributions are the following:

1. The formulation of a SLAD model and an efficient solution methodology to solve it. The formulation integrates energy and reserve variables. The solution algorithm uses parallel computing and accelerated Benders decomposition for enhanced computational efficiency.

2. Simulations on a large-scale realistic network confirm SLAD's superior performance. Comprehensive comparisons demonstrate its advantages over traditional deterministic real-time dispatch models with ramping products, under realistic forecasts and scenarios.

3. A detailed case study and sensitivity analysis reveal why SLAD outperforms existing approaches and to show how different scenarios and look-ahead horizons affect its effectiveness, providing insights for system planning and operations.

The rest of the chapter is organized as follows. Section 2.2 surveys the relevant literature. Section 2.3 introduces the economic dispatch formulations considered in the chapter. Section 2.4 presents the proposed accelerated Benders decomposition algorithm for solving SLAD. Section 2.5 illustrates the different formulations on a small example. Section 2.6 presents the experiment setting on a real-life system, Section 2.7 presents numerical results, and Section 2.8 discusses the sensitivity of each formulation. Section 2.9 identifies future research directions and concludes the chapter.

2.2 Related Work

2.2.1 Deterministic Formulation

Security-Constrained Economic Dispatch The SCED is the mathematical formulation that underlies virtually all real-time electricity markets in the US [28, 30, 31, 32]. The SCED formulations used in industry are linear programming (LP) problems that decide each generator's energy and reserve dispatch so as to minimize total operating costs, subject to physical, engineering and market constraints [24]. In its simplest form, SCED is a single-period, deterministic LP problem with tens of thousands of variables and constraints. It is typically solved in a few seconds using state-of-the-art LP solvers.

SCED with Flexiramps The main limitation of a single-period SCED formulation is that it does not consider the uncertainty and volatility of the net load. Therefore, to manage operational uncertainty in real-time markets, US ISOs are increasingly using short-term ramping products, also known as Flexible Ramping Products (FRPs) or *flexiramp*. For instance, MISO's real-time market includes a 10-minute and a 30-minute ramping product [26, 31]. Similarly, CAISO considers a 5-minute ramping product in its real-time market [28]. Also, similar ideas are proposed for unit commitment problems [33]. Nevertheless, previous studies on the benefits of these FRPs have observed that single-period formulations may still fail to account for future ramping needs, ultimately resulting in sub-optimal

decisions [29]. While FRPs can improve system flexibility, their design and implementation present challenges such as inaccurate forecasting, uncertainty estimation, and lack of structured pricing mechanisms [34]. Therefore, the authors recommend a careful ramp product market design and advocate a shift towards stochastic dispatch and commitment models. However, due to differences and incompatibilities in market design, an alternative method is proposed for European markets to approximate stochastic dispatch by determining reserve requirements for zonal operating reserves [35].

Look-Ahead Dispatch To extend SCED from a single-period to a multi-period formulation, look-ahead dispatch (LAD) is proposed in [36]. The main motivation for LAD is that the longer horizon allows to take into account future operating conditions, thereby achieving better economic performance in the long run. While the use of a look-ahead window is well established in ISOs' intra-day reliability unit commitment process [37, 38, 39], the use of LAD formulations in real-time markets has received less attention.

A LAD formulation for the ERCOT nodal market was been proposed in [40]. Building on this work, a study on the ERCOT system was conducted wherein a LAD with a one-hour look-ahead horizon is considered [41]. The chapter raises the question of "*how far should LAD look ahead?*", and highlights that forecast errors and unforeseen events can create misleading price signals, thereby resulting in sub-optimal decisions. The observation that LAD is sensitive to forecast errors is echoed in [42], which evaluates the sensitivity of LAD to changes in its input data. In particular, the authors report that LAD may exhibit worse performance than a single-period SCED when errors in the input data are too large. More recently, a reduced formulation for a LAD model that considers the N-k contingency criterion has been proposed, with results reported on systems with up to 1354 buses [43]. However, the reported computing times exceed 10 minutes for the larger case, which is too high for real-time market clearing (which occurs every 5 minutes).

In the US, both CAISO and NYISO use a LAD formulation in their real-time markets

[28, 30]. Namely, CAISO considers a one-hour look-ahead window with 5-minute time intervals, and NYISO considers a one-hour look-ahead window with a mixed 5-minute and 15-minute time intervals. Dispatch decisions in the first interval of LAD are binding, and dispatch decisions from subsequent intervals are advisory.

2.2.2 Stochastic Formulation

The reader is referred to [44] for a detailed overview of power systems optimization under uncertainty, which includes stochastic, robust, distributionally robust and chance-constrained optimization. The latter two approaches are computationally heavy, and have not been demonstrated to scale to industry-size instances. While several studies have used robust optimization (RO) for economic dispatch [45, 46, 47, 48], most consider small systems with only a few hundred buses, which is two orders of magnitude smaller than real-life instances. Furthermore, the tractability of RO heavily depends on the choice of uncertainty set: the resulting RO problem may be, e.g., a linear, second-order cone, mixed-integer, or semi-definite positive optimization problem. The design of data-driven, computationally tractable uncertainty sets in high-dimensional settings is an active area of research.

In contrast, stochastic optimization takes as input a probabilistic forecast in the form of scenarios, which can be generated using a variety of models [49, 50, 51]. Importantly, the structure of stochastic optimization models, and the corresponding solution algorithms, is agnostic to the nature of the probabilistic forecasting model. Therefore, the chapter focuses on stochastic optimization approaches, and assumes that a probabilistic forecasting model is available. The training of probabilistic forecast models is beyond the chapter's scope; the reader is referred to [49] for a recent overview of available tools.

Stochastic look-ahead dispatch (SLAD) (see Section 2.3.5) is a stochastic version of LAD to hedge against forecast uncertainty. Previous work has shown that, compared to deterministic ED formulations, SLAD can improve the reliability and efficiency. In [52], SLAD is evaluated on a 51-bus system, demonstrating more efficient dispatch decisions

compared to SCED. A multistage SLAD model with renewable production uncertainty and storage is proposed in [53], and solved using Stochastic Dual Dynamic Programming (SDDP). The approach is evaluated on a 228-bus system representing the German grid, with results showing around 1% savings compared to SCED . Similarly, a study on SLAD with a flexible ramping product demonstrates its effectiveness and resilience in handling worst-case scenarios on a 5-bus system [54]. Further comparisons of SLAD with SCED and LAD on a 5889-bus system in [55] indicate that SLAD provides greater cost savings, especially during intervals with high economic risks. However, this formulation does not account for reserve variables, which are crucial in U.S. real-time markets [55].

One of the primary challenges in stochastic optimization is accurately representing the uncertainty. To address this challenge, the sample average approximation method is often used. However, this method typically requires a large number of samples to estimate the expected cost accurately. As a result, the problem size increases substantially, making it challenging to solve the optimization problem efficiently. Benders decomposition is a well-suited method for solving stochastic optimization problems; it leverages the inherent structure of these formulations [56]. SDDP is a method used for solving multi-stage stochastic problems [57, 58] by combining cutting-plane methods with a Monte Carlo simulation. Progressive hedging [59] is another technique used to address two-stage stochastic problems. This method involves decomposing the problem by considering an augmented Lagrangian of the original problem.

Finally, the chapter focuses on addressing the scalability challenges of SLAD for industry-size problems, and evaluates the potential operational benefits of SLAD on a real industrial system. The design of SLAD-compatible pricing schemes, while important for the adoption of SLAD in real-time markets, is beyond the scope of the chapter. The reader is referred to [60, 61, 62, 63, 64] for recent studies on such pricing schemes.

2.3 Problem Formulation

This section presents the single-period (SCED), single-period with ramping products (SCED+RP), deterministic multi-period (LAD), and stochastic multi-period (SLAD) economic dispatch formulations considered in the chapter. These formulations are based on MISO's real-time market formulation [31]. Section 2.3 presents a nomenclature of the notations used in the chapter. Table 2.1 outlines the number of time steps, the number of scenarios, and whether flexible ramping products (FRPs) are included in each problem formulation.

For brevity, this chapter presents a simplified version of the economic dispatch formulation. The complete mathematical model, including all constraints, is detailed in Appendix A.

Nomenclature

Sets and Indices

$i \in \mathcal{N}$	Buses
$e \in \mathcal{E}$	Branches
$g \in \mathcal{G}$	Generators
$t \in \mathcal{T}$	Time steps
$\mathcal{G}_i$	Generators at bus i
$s \in \mathcal{S}$	Scenarios

Variables

$\mathbf{p}_{i,t,s}$	Net power injection at bus i at time t in scenario s (MW)
$\mathbf{pg}_{g,t,s}$	Energy dispatch of generator g at time t in scenario s (MW)
$\mathbf{res}_{g,t,s}$	Reserve of generator g at time t in scenario s (MW)

$\mathbf{rc}^{\uparrow}_{g,t,s}, \mathbf{rc}^{\downarrow}_{g,t,s}$ Ramping-up and -down capability of generator g at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{rc}^{\uparrow}_{t,s}, \boldsymbol{\delta}\mathbf{rc}^{\downarrow}_{t,s}$ Ramping-up and -down capability shortage at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{f}_{e,t,s}$ Thermal limit violation on branch e at time t in scenario s (MW)

Parameters

Δ_t Duration of a time period (min)

Γ_t Duration of the ramping product (min)

p_s Probability of scenario s

$\mathbf{pd}_{i,t,s}$ Power demand at bus i at time t in scenario s (MW)

$\sigma_{e,i}$ PTDF coefficient for branch e and bus i

$\underline{\mathbf{pg}}_{g,t,s}, \bar{\mathbf{pg}}_{g,t,s}$ Min and max output of generator g at time t in scenario s (MW)

$\mathbf{r}^{\uparrow}_{g,t}, \mathbf{r}^{\downarrow}_{g,t}$ Ramping-up and -down rate of generator g at time t (MW/min)

$\mathbf{rr}^{\mathbf{res}}_{t}$ Reserve requirement (MW)

$\mathbf{rr}^{\mathbf{rc}^{\uparrow}}_{t}, \mathbf{rr}^{\mathbf{rc}^{\downarrow}}_{t}$ Ramping-up and -down capability requirement at time t (MW)

$\underline{\mathbf{pf}}_{e,t}, \bar{\mathbf{pf}}_{e,t}$ Lower and upper thermal limits on branch e at time t (MW)

$\mathbf{c}^{\mathbf{pg}}_{g,t}$ Energy dispatch cost for generator g at time t (\$/MW)

$\mathbf{c}^{\mathbf{res}}_{g,t}$ Reserve cost for generator g at time t (\$/MW)

$\pi^{\mathbf{rc}^{\uparrow}}, \pi^{\mathbf{rc}^{\downarrow}}$ Ramping-up and -down capability shortage penalty price (\$/MW)

$\pi^{\mathbf{f}}{}_{e,t}$ Thermal limit violation penalty price for branch e at time t (\$/MW)

2.3.1 Variables

The economic dispatch models comprise several key decision variables. The net power injection at bus i for a given time t and scenario s is denoted by $\mathbf{p}_{i,t,s}$ and measured in megawatts (MW). Generator outputs are represented by $\mathbf{pg}_{g,t,s}$, indicating the energy dispatched by generator g at time t in scenario s. For simplicity, different types of reserve capacities, such as regulating, operating, spinning and non-spinning reserves, are aggregated together and represented as $\mathbf{res}_{g,t,s}$ for each generator. The model also accounts for the ramping capabilities of generators, which are specified by $\mathbf{rc}^{\uparrow}_{g,t,s}$ for ramping up and $\mathbf{rc}^{\downarrow}_{g,t,s}$ for ramping down. Shortages in these ramping capabilities are captured by $\boldsymbol{\delta}\mathbf{rc}^{\uparrow}_{t,s}$ and $\boldsymbol{\delta}\mathbf{rc}^{\downarrow}_{t,s}$, respectively. Lastly, $\boldsymbol{\delta}\mathbf{f}_{e,t,s}$ quantifies the thermal limit violations on branch e at time t in scenario s.

2.3.2 Constraints

Power balance constraints Equation (2.1a) ensures nodal power balance by equating the net power at each node i (the difference between generated power $\mathbf{pg}$ and demanded power pd) for every time step t and scenario s. Equation (2.1b) enforces that the total net power across all nodes $\mathcal{N}$ sums to zero, maintaining the overall system balance for each time and scenario.

$$\mathbf{p}_{i,t,s} = \sum_{g \in \mathcal{G}_i} \mathbf{pg}_{g,t,s} - \mathrm{pd}_{i,t,s}, \qquad \forall\, i, t, s \tag{2.1a}$$

$$\sum_{i \in \mathcal{N}} \mathbf{p}_{i,t,s} = 0, \qquad \forall\, t, s \tag{2.1b}$$

Generation limits The generation constraints ensure that generator outputs remain within specified operational limits. Specifically, the sum of generated power $\mathbf{pg}_{g,t,s}$, reserved power $\mathbf{res}_{g,t,s}$, and ramp-up capability $\mathbf{rc}^{\uparrow}_{g,t,s}$ must not exceed the maximum generation limit $\bar{\mathrm{pg}}_{g,t,s}$ as shown in Equation (2.2a). Conversely, the difference between generated power, reserved power, and ramp-down capability $\mathbf{rc}^{\downarrow}_{g,t,s}$ must remain above the minimum

generation limit $\underline{\mathrm{pg}}_{g,t,s}$, ensuring operational reliability and flexibility as stated in Equation (2.2b).

$$\mathbf{pg}_{g,t,s} + \mathbf{res}_{g,t,s} + \mathbf{rc}^{\uparrow}_{g,t,s} \leq \bar{\mathrm{pg}}_{g,t,s}, \qquad \forall\, g,t,s \tag{2.2a}$$

$$\mathbf{pg}_{g,t,s} - \mathbf{res}_{g,t,s} - \mathbf{rc}^{\downarrow}_{g,t,s} \geq \underline{\mathrm{pg}}_{g,t,s}, \qquad \forall\, g,t,s \tag{2.2b}$$

Generation ramp limits Ramping constraints for generators are critical to maintaining system stability and responsiveness. Equation (2.3a) ensures that the power output of any generator g at time t in scenario s must not exceed the previous time step's output incremented by the maximum ramp-up rate $\mathrm{r}^{\uparrow}_{g,t}$ scaled by the time duration Δ_t. Similarly, Equation (2.3b) ensures that the power output does not fall below the previous time step's output reduced by the maximum ramp-down rate $\mathrm{r}^{\downarrow}_{g,t}$ scaled by the same time step Δ_t.

$$\mathbf{pg}_{g,t,s} \leq \mathbf{pg}_{g,t-1,s} + \Delta_t\, \mathrm{r}^{\uparrow}_{g,t} \qquad \forall g,t,s \tag{2.3a}$$

$$\mathbf{pg}_{g,t,s} \geq \mathbf{pg}_{g,t-1,s} - \Delta_t\, \mathrm{r}^{\downarrow}_{g,t} \qquad \forall g,t,s \tag{2.3b}$$

Transmission Constraints Transmission constraints are essential for controlling the flow of electricity through transmission lines or transformers. They are formalized in Equations (2.4a) and (2.4b), which ensure that the weighted sum of net power injections $\mathbf{p}_{i,t,s}$ at each bus i does not exceed the maximum allowable flow $\bar{\mathrm{pf}}_{e,t}$ and does not drop below the minimum allowable flow $\underline{\mathrm{pf}}_{e,t}$. These limits are adjusted by $\boldsymbol{\delta}\mathbf{f}_{e,t,s}$ to account for operational deviations, helping to prevent overloads and maintain stable electricity distribution under varying operational scenarios.

$$\sum_{i \in \mathcal{N}} \sigma_{e,i}\, \mathbf{p}_{i,t,s} \leq \bar{\mathrm{pf}}_{e,t} + \boldsymbol{\delta}\mathbf{f}_{e,t,s}, \qquad \forall\, e,t,s \tag{2.4a}$$

$$\sum_{i \in \mathcal{N}} \sigma_{e,i}\, \mathbf{p}_{i,t,s} \geq \underline{\mathrm{pf}}_{e,t} - \boldsymbol{\delta}\mathbf{f}_{e,t,s}, \qquad \forall\, e,t,s \tag{2.4b}$$

Non-negativity Constraints Equation (2.5) ensures that reserves ($\mathbf{res}$), ramp capabilities ($\mathbf{rc}^{\uparrow}$ and $\mathbf{rc}^{\downarrow}$), ramp capability shortages ($\boldsymbol{\delta}\mathbf{rc}^{\uparrow}$ and $\boldsymbol{\delta}\mathbf{rc}^{\downarrow}$), and thermal limit deviations ($\boldsymbol{\delta}\mathbf{f}$)

must all be nonnegative.

$$\mathbf{res}, \mathbf{rc}^{\uparrow}, \mathbf{rc}^{\downarrow}, \boldsymbol{\delta}\mathbf{rc}^{\uparrow}, \boldsymbol{\delta}\mathbf{rc}^{\downarrow}, \boldsymbol{\delta}\mathbf{f} \geq \mathbf{0}. \tag{2.5}$$

Ramping capability constraints These constraints are also referred to as "Flexiramp" [29]. Flexiramp is designed to ensure sufficient flexible generation capacity to handle volatile net loads effectively. These constraints are included only in the SCED+RP formulation and not in other formulations. Equations (2.6a) and (2.6b) ensure that the total ramp-up and ramp-down capabilities of all generators meet the system's ramping requirements, $\mathrm{rr}_t^{\mathbf{rc}^{\uparrow}}$ and $\mathrm{rr}_t^{\mathbf{rc}^{\downarrow}}$, respectively. These are modeled as soft constraints where $\boldsymbol{\delta}\mathbf{rc}_{t,s}^{\uparrow}$ and $\boldsymbol{\delta}\mathbf{rc}_{t,s}^{\downarrow}$ represent the shortages in ramp-up and ramp-down capabilities. Additionally, Equations (2.6c) and (2.6d) ensure that the ramping capabilities do not exceed the product of the specified ramp duration Γ_t and the ramp rates. For instance, for a 30-minute ramping product, the ramping capabilities are limited by multiplying the 30-minute duration by the respective ramp rates in MW per minute.

$$\sum_{g \in \mathcal{G}} \left(\mathbf{rc}_{g,t,s}^{\uparrow} \right) \geq \mathrm{rr}_t^{\mathbf{rc}^{\uparrow}} - \boldsymbol{\delta}\mathbf{rc}_{t,s}^{\uparrow}, \qquad \forall\, t, s \tag{2.6a}$$

$$\sum_{g \in \mathcal{G}} \left(\mathbf{rc}_{g,t,s}^{\downarrow} \right) \geq \mathrm{rr}_t^{\mathbf{rc}^{\downarrow}} - \boldsymbol{\delta}\mathbf{rc}_{t,s}^{\downarrow}, \qquad \forall\, t, s \tag{2.6b}$$

$$\mathbf{rc}_{g,t,s}^{\uparrow} \leq \Gamma_t\, \mathrm{r}_{g,t}^{\uparrow}, \qquad \forall\, g, t, s \tag{2.6c}$$

$$\mathbf{rc}_{g,t,s}^{\downarrow} \leq \Gamma_t\, \mathrm{r}_{g,t}^{\downarrow}, \qquad \forall\, g, t, s \tag{2.6d}$$

Non-anticipatory Constraints In the stochastic formulation of our model, non-anticipatory constraints are crucial to ensure that decisions at the first time step remain consistent across all scenarios. Specifically, Equations (2.7a), (2.7b), (2.7c), and (2.7d) require that the power generation ($\mathbf{pg}$), reserves ($\mathbf{res}$), and ramping capabilities ($\mathbf{rc}^{\uparrow}$ and $\mathbf{rc}^{\downarrow}$) for any generator g

must be the same for all scenarios $\mathcal{S}$ in real-time operations ($t = 1$).

$$\mathbf{pg}_{g,1,s_1} = \mathbf{pg}_{g,1,s_2}, \qquad \forall\, g \in \mathcal{G}, s_1, s_2 \in \mathcal{S} \tag{2.7a}$$

$$\mathbf{res}_{g,1,s_1} = \mathbf{res}_{g,1,s_2}, \qquad \forall\, g \in \mathcal{G}, s_1, s_2 \in \mathcal{S} \tag{2.7b}$$

$$\mathbf{rc}^{\uparrow}_{g,1,s_1} = \mathbf{rc}^{\uparrow}_{g,1,s_2}, \qquad \forall\, g \in \mathcal{G}, s_1, s_2 \in \mathcal{S} \tag{2.7c}$$

$$\mathbf{rc}^{\downarrow}_{g,1,s_1} = \mathbf{rc}^{\downarrow}_{g,1,s_2}, \qquad \forall\, g \in \mathcal{G}, s_1, s_2 \in \mathcal{S} \tag{2.7d}$$

2.3.3 Objective Function

The objective function of the model is designed to minimize the total expected operational and penalty costs of the power system over all time periods t in set $\mathcal{T}$, and across all scenarios s in set $\mathcal{S}$. It is expressed in Equation (2.8) as the weighted sum of generation costs, reserve costs, and penalties associated with ramping capability shortages and thermal limit violations.

$$\sum_{t\in\mathcal{T}, s\in\mathcal{S}} p_s \Delta_t \left(\sum_{g\in\mathcal{G}} \left(\mathrm{c}^{\mathrm{pg}}_{g,t}\, \mathbf{pg}_{g,t,s} + \mathrm{c}^{\mathrm{res}}_{g,t}\, \mathbf{res}_{g,t,s} \right) + \pi^{\mathrm{rc}^{\uparrow}}\, \boldsymbol{\delta}\mathbf{rc}^{\uparrow}_{t,s} + \pi^{\mathrm{rc}^{\downarrow}}\, \boldsymbol{\delta}\mathbf{rc}^{\downarrow}_{t,s} + \sum_{e\in\mathcal{E}} {\pi^{\mathrm{f}}}_{e,t}\, \boldsymbol{\delta}\mathbf{f}_{e,t,s} \right) \tag{2.8}$$

2.3.4 Deterministic Formulations

Baseline RT-SCED The baseline SCED formulation considered in the chapter minimizes operational and penalty costs over a single time period and scenario. It is formulated as the linear programming problem

$$\text{(SCED)} \quad \begin{aligned} &\min \quad (2.8) \\ &\text{s.t.} \quad (2.1) - (2.5) \quad \forall i \in \mathcal{N}, g \in \mathcal{G}, t \in \mathcal{T}, s \in \mathcal{S} \end{aligned} \tag{2.9}$$

where $\mathcal{T} = \{1\}$ and $\mathcal{S} = \{1\}$.

This formulation is the basis of, e.g., real-time markets in MISO [65, 31]. The objective (2.8) computes the overall cost of operation and penalties. The constraints capture global power balance, generator and reserve min/max limits, reserve requirements, and transmis-

sion constraints. Additionally, the formulation includes ramping up and down requirements to ensure that changes in dispatch can be accommodated by the ramp-up rate.

RT-SCED with Flexiramp

Following the ramping products design outlined by MISO [26], the SCED with ramping products (SCED+RP) formulation incorporates 10-minute ramping-up and ramping-down capability products, along with 30-minute short-term ramping products. The ramping requirements are calculated based on changes and uncertainties in net load. Model (2.10) illustrates the corresponding formulation. Compared to Model (2.9), this formulation includes ramping capability constraints described in 2.3.2, also known as ramping products.

$$\text{(SCED+RP)} \quad \begin{aligned} \min \quad & (2.8) \\ \text{s.t.} \quad & (2.1) - (2.6), \quad \forall i \in \mathcal{N}, g \in \mathcal{G}, t \in \mathcal{T}, s \in \mathcal{S} \end{aligned} \tag{2.10}$$

where $\mathcal{T} = \{1\}, \mathcal{S} = \{1\}$.

Look-Ahead Dispatch

Look-Ahead Dispatch (LAD) extends single-period SCED to a multi-period economic dispatch formulation over a horizon of length T. The corresponding compact formulation is

$$\text{(LAD)} \quad \begin{aligned} \min \quad & (2.8) \\ \text{s.t.} \quad & (2.1) - (2.5) \quad \forall i \in \mathcal{N}, g \in \mathcal{G}, t \in \mathcal{T}, s \in \mathcal{S} \end{aligned} \tag{2.11}$$

where $\mathcal{T} = \{1, ..., T\}$ and $\mathcal{S} = \{1\}$.

The problem takes as inputs deterministic forecasts for load and renewable generation, and outputs the energy dispatch and reserves for each generator and each period. Transmission constraints and reserve requirements are also enforced for each period. Decisions at time t are linked to decisions at time $t + 1$ via ramping constraints for the generators (constraints (2.3a) and (2.3b)). If the ramping constraints are not binding, LAD decomposes into T independent SCED problems, and the solution at $t = 1$ is equivalent to the

single-period SCED solution.

2.3.5 Stochastic Formulation

Stochastic look-ahead dispatch (SLAD) further extends LAD by considering uncertainty in future electricity demand and renewable generation, i.e., SLAD considers multiple time steps and multiple scenarios. Namely, the formulation considered in this chapter is a two-stage formulation, where the first-stage decisions are the decision variables at $t = 1$ and the second-stage decisions are decision variables at times $t = 2, ..., T$. This reflects the fact that only dispatch decisions for $t = 1$ are communicated to generators and implemented; subsequent decisions ($t = 2, ..., T$) are only advisory. The resulting extended SLAD formulation reads

$$\text{(SLAD)} \quad \begin{aligned} &\min \quad (2.8) \\ &\text{s.t.} \quad (2.1) - (2.5), (2.7) \quad \forall i \in \mathcal{N}, g \in \mathcal{G}, t \in \mathcal{T}, s \in \mathcal{S} \end{aligned} \tag{2.12}$$

where $\mathcal{T} = \{1, ..., T\}$ and $\mathcal{S} = \{1, ..., S\}$. Naturally, if a single scenario is considered, i.e., $|\mathcal{S}| = 1$, SLAD is equivalent to LAD. Table 2.1 summarizes the ED formulations considered in the chapter.

Table 2.1: The Economic Dispatch Formulations.

Problem	Time steps ($\|\mathcal{T}\|$)	Scenarios ($\|\mathcal{S}\|$)	FRPs*
SCED	1	1	✗
SCED+RP	1	1	✓
LAD	>1	1	✗
SLAD	>1	>1	✗

*FRPs: Flexible Ramp Products.

Non-anticipatory constraints (2.7) ensure that the first-stage decisions $\mathbf{x}_{1,s}$ are the same across all scenarios. Problem (2.12) is a (large) linear problem, with $\mathcal{O}(T \times S)$ variables and constraints, which is tractable only for small to medium-sized instances. Large instances

require specialized algorithms such as Benders decomposition.

2.4 Accelerated Benders Decomposition

The extensive formulation (2.12) is not tractable for the large-scale instances considered in this work, especially for real-time operations. Instead, specialized algorithms are needed, such as Benders Decomposition (BD) [66] and the Progressive Hedging Algorithm (PHA) [67]. In preliminary experiments, Benders decomposition was found to be faster and exhibit better convergence than PHA, hence, the chapter focuses on BD. This section presents the Benders decomposition (BD) algorithm and acceleration techniques to improve its convergence for large SLAD instances.

For ease of reading, the proposed accelerated Benders decomposition is stated for a general two-stage stochastic program of the form

$$\min_{\mathbf{x},\mathbf{y}} \quad c^\top \mathbf{x} + \sum_{s\in\mathcal{S}} q_s^\top \mathbf{y}_s \tag{2.13a}$$

$$\text{s.t.} \quad A\mathbf{x} \geq b, \tag{2.13b}$$

$$H_s\mathbf{x} + W_s\mathbf{y}_s \geq h_s, \quad \forall s \in \mathcal{S} \tag{2.13c}$$

where x and y denote the first-stage and second-stage variables, respectively. The objective (2.13a) minimizes the sum of first-stage and expected second-stage costs; for compactness, the probability associated with scenario s is captured in the definition of q_s. Constraints (2.13b) and (2.13c) capture first-stage and second-stage constraints, respectively. Recall that, in the context of SLAD, first-stage variables $\mathbf{x}$ comprise all decision variables for the first time step $t = 1$, and second-stage variables y comprise all decision variables for time steps $t = 2, ..., T$.

2.4.1 Benders Decomposition

The Benders master problem considers only the first-stage variables and is of the form

$$\min_{\mathbf{x},\theta} \quad c^\top \mathbf{x} + \sum_{s\in\mathcal{S}} \theta_s \tag{2.14a}$$

$$\text{s.t.} \quad A\mathbf{x} \geq b, \tag{2.14b}$$

$$\theta_s \geq Q_s(\mathbf{x}), \qquad \forall s \in \mathcal{S} \tag{2.14c}$$

where $Q_s(\mathbf{x})$ denotes the second-stage cost of $\mathbf{x}$ under scenario s, defined as

$$Q_s(\mathbf{x}) = \min_{\mathbf{y}_s} \quad q_s^\top \mathbf{y}_s \tag{2.15a}$$

$$\text{s.t.} \quad W_s \mathbf{y}_s \geq h_s - H_s \mathbf{x}. \tag{2.15b}$$

In all economic dispatch formulations considered in the chapter, system-wide constraints such as power balance, reserve requirements and transmission constraints, are treated as soft constraints. Therefore, SLAD has relatively complete recourse, i.e., for any first-stage solution $\bar{\mathbf{x}}$ that satisfies constraints the sub-problem (2.15) is feasible

The second-stage cost function Q_s is the value function of a linear programming problem, hence, it is a convex, piece-wise linear function. However, in general, Q_s contains exponentially many pieces, making its direct representation intractable. Instead, Benders decomposition uses a cutting-plane approach to iteratively refine a piece-wise linear under-approximation of Q_s until convergence. Denoting by λ_s the dual variable associated with constraint (2.15b), each Benders cut has the form

$$\theta_s \geq \lambda_s^\top (h_s - H_s \mathbf{x}). \tag{2.16}$$

The cuts highlighted in (2.16) are also referred to as *Benders optimality cuts*. Because SLAD has relatively complete recourse, it is sufficient to consider optimality cuts. The reader is referred to [66] for the general treatment of so-called feasibility cuts, which are needed when subproblems can be infeasible. In what follows, using a slight abuse of notation, each BD cut is identified with the corresponding dual solution $\lambda_s \in \Lambda_s$ where, $\forall s \in \mathcal{S}$,

Λ_s denotes the dual-feasible set

$$\Lambda_s = \left\{ \lambda_s \geq 0 \;\middle|\; W_s^\top \lambda_s = q_s \right\}. \tag{2.17}$$

The Benders master problem is initialized with a small number of cuts $\bar{\Lambda} \subseteq \Lambda = (\Lambda_1, ..., \Lambda_S)$. The corresponding relaxed master problem (RMP) reads

$$RMP(\bar{\Lambda}) \qquad \min_{\mathbf{x},\theta} \quad c^\top \mathbf{x} + \sum_{s \in \mathcal{S}} \theta_s \tag{2.18a}$$

$$\text{s.t.} \quad A\mathbf{x} \geq b, \tag{2.18b}$$

$$\theta_s \geq \lambda_s^\top (h_s - H_s \mathbf{x}), \qquad \forall s \in \mathcal{S}, \forall \lambda_s \in \bar{\Lambda}_s. \tag{2.18c}$$

At each iteration, new cuts are identified by solving the Benders sub-problem (2.15) for each scenario, and retrieving the corresponding optimal dual solution. Violated cuts are added to the master problem, and the algorithm terminates with an optimal solution when no violated cut exists. Although BD is guaranteed to converge in a finite number of iterations, textbook implementations often suffer from slow convergence. Hence, the chapter implements several acceleration strategies described below.

2.4.2 Acceleration Strategies for the Subproblems

Two mechanisms are implemented to speed up the resolution of the BD subproblems. First, using shared or distributed memory architectures, the subproblems are solved in parallel, each subproblem being handled by an independent worker process. At each iteration, the candidate master solution is sent to each worker process, which returns the solution of the corresponding subproblem. This strategy yields significant speedups, especially since, for the case at hand, optimizing the subproblems is substantially more time-consuming than optimizing the master problem.

Second, each subproblem contains a large number of transmission constraints of the form (2.4), which are modeled via a Power Transfer Distribution Factor (PTDF) formulation. Because only a small number of transmission lines are congested in practice, PTDF

constraints are handled in a lazy fashion. Namely, each subproblem is initialized without any transmission constraints, and solved in an iterative fashion. Thereby, at each iteration, the current solution is checked for violations, and only violated transmission constraints are added to the problem. This process is repeated until all transmission constraints are satisfied. By leveraging the warm-start capabilities of the simplex algorithm, this strategy significantly speeds up the resolution of individual subproblems. Importantly, it does not impact the validity nor quality of Benders cuts. For the same reason, transmission constraints are also handled in lazy fashion when solving the Benders master problem.

2.4.3 In-out Separation

To alleviate the slow convergence and tailing off effect of a vanilla BD, the chapter implements an *in-out separation* technique [68, 69]. Thereby, at each iteration, BD cuts are generated by seeding the subproblems with a convex combination of the current master solution $\bar{\mathbf{x}}$ and a core point $\hat{\mathbf{x}}$, which resides in the interior of the feasible region. Namely, the subproblems receive $\tilde{\mathbf{x}} = \alpha\bar{\mathbf{x}} + (1-\alpha)\hat{\mathbf{x}}$, where $\alpha \in [0,1]$ is typically chosen by performing a line search over the segment $[0,1]$. This strategy is known to generate stronger cuts [70], thereby improving convergence, especially in later iterations.

In practice, one should be mindful of the following facts regarding the choice of α. On the one hand, a full line search comes at the cost of having to solve each subproblem multiple times per BD iteration, which may be computationally burdensome. On the other hand, using a fixed $\alpha < 1$ throughout the algorithm does not guarantee convergence.

To balance these issues, the chapter uses the following strategy. At each BD iteration, subproblems are first seeded with $\tilde{\mathbf{x}} = \bar{\alpha}\bar{\mathbf{x}} + (1-\bar{\alpha})\hat{\mathbf{x}}$, where $0 < \bar{\alpha} < 1$ is a user-specified parameter. If a violated cut is found, it is added to the master problem and the algorithm continues to next BD iteration. If no violated cut is found, the subproblems are re-seeded with the current master solution $\bar{\mathbf{x}}$, which is equivalent to setting $\alpha = 1$. This scheme guarantees finite convergence while ensuring that subproblems are solved at most twice at each

iteration. Indeed, if no violated cut is found after the second step, then the current master solution $\bar{x}$ is optimal and the algorithm terminates.

Another important aspect of the in-out strategy is the choice of core point $\hat{\mathbf{x}}$. In the chapter's implementation, $\hat{\mathbf{x}}$ is initialized as the solution of the initial master problem. Then, at each iteration, $\hat{\mathbf{x}}$ is set to the last solution that was sent to the subproblems. Thereby, $\hat{\mathbf{x}}$ is set to $\tilde{\mathbf{x}}$ if the first round of separation yielded a violated cut, and to $\bar{\mathbf{x}}$ otherwise. This aggressive strategy was found to be effective at preventing tailing off. Finally, note that, although this scheme does not guarantee that $\hat{\mathbf{x}}$ lies in the interior of the feasible region, it does not affect the validity of the algorithm.

2.4.4 Accelerated Benders Decomposition Algorithm

Algorithm 5 Accelerated Benders Decomposition

Require: Tolerance $\epsilon \geq 0$; in-out parameter $\bar{\alpha} \in (0, 1)$; initial set of cuts $\bar{\Lambda} = (\bar{\Lambda}_1, ..., \bar{\Lambda}_S)$

$(LB, UB) \leftarrow (-\infty, +\infty)$
Solve RMP($\bar{\Lambda}$); retrieve optimal solution $\bar{\mathbf{x}}$
$\hat{\mathbf{x}} \leftarrow \bar{\mathbf{x}}$
while $(UB - LB) \geq \epsilon$ **do**
    Solve RMP($\bar{\Lambda}$); retrieve optimal solution $(\bar{\mathbf{x}}, \bar{\theta})$
    $LB \leftarrow \max(LB, c^\top \bar{\mathbf{x}} + \sum_s \bar{\theta}_s)$
    $(\alpha, n^c) \leftarrow (\bar{\alpha}, 0)$
    while $n^c = 0$ **do**
        $\tilde{\mathbf{x}} \leftarrow \alpha \bar{\mathbf{x}} + (1 - \alpha)\hat{\mathbf{x}}$
        for $s = 1, 2, \ldots, S$ **do**
            Solve subproblem (2.15) to obtain $Q_s(\tilde{\mathbf{x}})$ and dual solution λ_s
            if $\bar{\theta}_s < \lambda_s^\top (h_s - H_s \bar{\mathbf{x}})$ **then**
                $\bar{\Lambda}_s \leftarrow \bar{\Lambda}_s \cup \{\lambda_s\}$
                $n^c \leftarrow n^c + 1$
            end if
        end for
        if $(n^c = 0)$ and $(\alpha < 1)$ **then**
            $\alpha \leftarrow 1$
        else if $(n^c = 0)$ and $(\alpha = 1)$ **then**
            break
        end if
    end while
    $UB \leftarrow \min\left(UB, c^\top \tilde{\mathbf{x}} + \sum_s Q_s(\tilde{\mathbf{x}})\right)$
    $\hat{\mathbf{x}} \leftarrow \tilde{\mathbf{x}}$
end while

The proposed Accelerated Benders Decomposition (ABD) algorithm, incorporating parallel solving of sub-problems, lazy thermal constraints, and in-out separation, is detailed

in Algorithm 5.

To prevent the relaxed master problem from being unbounded, it is initialized with trivial valid cuts of the form $\theta_s \geq M, \forall s$, where M is a large negative constant. The core point $\hat{\mathbf{x}}$ is initialized at line 3 using the initial RMP solution. Each ABD iteration solves the RMP to obtain $(\bar{\mathbf{x}}, \bar{\theta})$ (line 5), updates the lower bound (line 6), and enters an in-out separation loop. This loop generates Benders cuts by seeding the subproblems with $\tilde{\mathbf{x}} = \alpha\bar{\mathbf{x}} + (1-\alpha)\hat{\mathbf{x}}$ (line 9) using a convex weight α. Subproblems are solved for each scenario using $\tilde{\mathbf{x}}$ (lines 10–16), and violated cuts are added to the RMP (line 13). If no violated cuts are found during the first separation round (i.e. $n^c = 0$ and $\alpha = 1$), α is set to 1 at line 18, effectively moving the separation point $\tilde{\mathbf{x}}$ to the current master solution $\bar{\mathbf{x}}$ and triggering a re-solve of the sub-problems. If no violated cuts are found after setting α to 1, the separation loop is interrupted (line 20); note that this indicates that the current solution is optimal. This scheme ensures that subproblems are solved at most twice per iteration while retaining the validity of the overall algorithm. Finally, the upper bound is updated based on subproblem solutions (line 23), and the core point $\hat{\mathbf{x}}$ is updated. This iterative process is repeated until the optimality gap $(UB - LB)$ reaches the prescribed tolerance ϵ.

2.5 Illustrative Example

This section illustrates the behavior of each Economic Dispatch (ED) formulation on a small system over two time periods. The system at hand comprises two 20MW generators G_1 and G_2 and a single load; for simplicity, no reserve products nor transmission lines are considered. Table 2.2 reports the operating parameters of both generators. Generator G_1 has lower cost and higher ramping rate; generator G_2 has higher cost, and lower ramping rate.

Table 2.2: Generator operating parameters

Unit	Capacity	Cost	Ramp rate
G_1	20 MW	\$10/MW	4 MW/min
G_2	20 MW	\$20/MW	2 MW/min

The look-ahead horizon in LAD and SLAD formulations is $T = 2$, and the number of scenarios in SLAD is $S = 2$. Table 2.3 reports the demand forecast used in each formulation for $t, =, 1, 2$. At each time step, SCED and SCED+RP only consider the current known demand, which is the actual demand, whereas LAD and SLAD require future demand information and therefore rely on forecasts. SCED+RP additionally considers a 22MW requirement for a 5-minute flexible ramping product at $t = 1$. The deterministic forecast in LAD underestimates future demand. Finally, note that, at each time, the average of the two scenarios considered in SLAD is equal to the deterministic forecast used by LAD.

Table 2.3: Forecast demand and solutions for each ED formulation

	$t = 1$				$t = 2$			
Problem	Forecast	$\mathbf{pg}$	δ	Cost	Forecast	$\mathbf{pg}$	δ	Cost†
SCED	10	(10, 0)	0	100	35	(20, 10)	5	5400
SCED+RP	10	(8, 2)	0	120	35	(20, 12)	3	3440
LAD	(10, 33)	(7, 3)	0	130	(35, 29)	(20, 13)	2	2460
SLAD*	$\begin{pmatrix} 10 & 29 \\ 10 & 37 \end{pmatrix}$	(3, 7)	0	170	$\begin{pmatrix} 35 & 27 \\ 35 & 31 \end{pmatrix}$	(20, 15)	0	500

*each row corresponds to one scenario. †include \$1000 per MW of shortage.

Table 2.3 reports, for each formulation and each time $t = 1, 2$: the active power dispatch of both generators ($\mathbf{pg} \in \mathbb{R}^2$, in MW), the corresponding energy shortage (δ, in MW), and the operating costs (in \$). The latter comprises energy dispatch costs at time t, and includes

a penalty of $1000 per MW of energy shortage. Note that, for multi-period formulations, only the first-stage decision is considered when computing actual operating costs.

The results of Table 2.3 show that SCED achieves the lowest costs at $t = 1$, but the highest total costs because of energy shortages at $t = 2$, which result in large penalty costs. Similarly, SCED+RP and LAD also achieve lower costs at $t = 1$, but experiences energy shortages at $t = 2$, albeit less than SCED. On the other hand, SLAD, despite incurring a higher cost at $t = 1$, avoids energy shortages at $t = 2$ and therefore achieves lower costs over the horizon. This illustrates the fact that the short-term focus of SCED results in sub-optimal decisions in the long term.

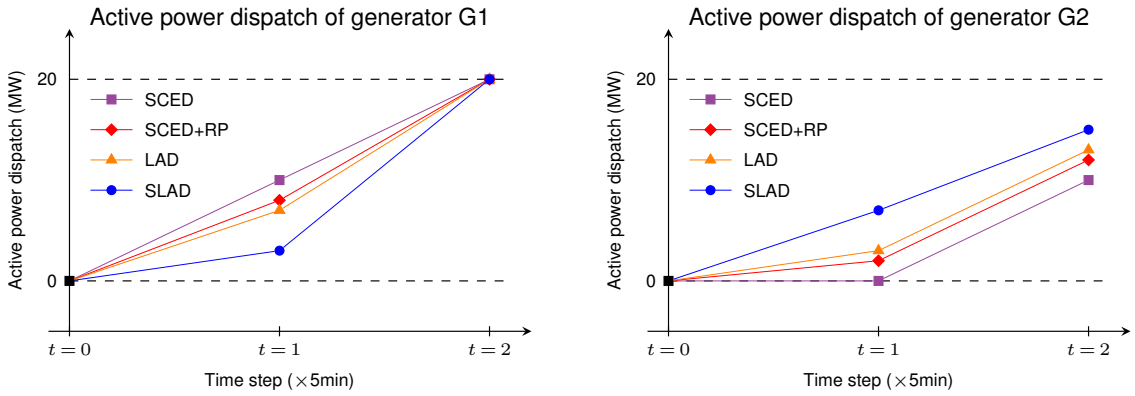


Figure 2.1: Active power dispatch of generator G1 (top) and G2 (bottom) time $t = 0, 1, 2$, under each economic dispatch formulation (**SCED**/**SCED+RP**/**LAD**/**SLAD**). The black square is the initial dispatch ($t = 0$).

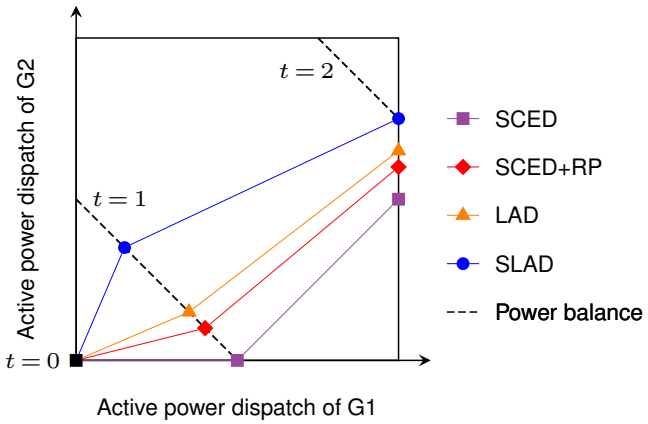


Figure 2.2: Illustration of the active power dispatch obtained by each economic dispatch formulation (SCED/SCED+RP/LAD/SLAD). Both generator's initial output (t=0, black square) is 0MW. At $t=1$, all formulations meet the 10MW demand. At $t=2$, both SCED and LAD fail to meet the 35MW demand, because generator G2 is not able to ramp sufficiently fast. SLAD is able to meet the power balance constraint because it had sufficiently ramped up G2 at $t=1$.

To better understand the behavior of each ED formulation, Figures 2.1 and 2.2 illustrate the resulting dispatch decisions at $t=1,2$. Namely, Figure 2.1 illustrates the dispatch of generator G1 and G2 (left and right pane, respectively) at time $t=0,1,2$. In particular, the figure shows that SLAD and, to a less extent, SCED+RP and LAD, pre-emptively ramp-up generator G2 at $t=1$, despite the immediate additional cost. This early ramp-up allows SLAD, SCED+RP and LAD to dispatch generator G2 to a higher value at $t=2$ than SCED, thereby reducing energy shortage penalty costs. Nevertheless, unlike SLAD, SCED+RP and LAD still experiences some energy shortages, albeit less than SCED. This is because the demand forecast used in LAD at $t=1$ underestimates future demand, thereby resulting in lower dispatch for generator G2 at $t=1$ than SLAD. Similarly, the flexible ramp requirement in SCED+RP at $t=1$ underestimates net load variability, resulting in a lower dispatch for generator G2 at $t=1$. Finally, Figure 2.2 displays the same solution in the (p_1,p_2) space. It is evident from Figure 2.2 that SCED, SCED+RP and LAD fail to meet the power balance requirement at $t=2$, and incur energy shortages of 5MW, 3MW and 2MW, respectively.

Naturally, the solution obtained by SCED+RP, and the corresponding costs, heavily depend on the amount of FRP procured by system operators. In this example, when the FRP requirement at $t = 1$ is no greater than 20MW, the SCED+RP formulation behaves identically to SCED: this is an indication that the FRP constraint (2.6a) is not binding. However, increasing the FRP amount from 20MW to 25MW results in a likewise reduction in power shortage at $t = 2$, from 5MW (the same value as SCED) to 0MW. Further increasing the FRP amount beyond 25MW also eliminates power shortages, but increases operating costs as more expensive generators are dispatched to jointly procure energy and flexible ramping. Therefore, as noted in [29], identifying the right amount of FRP to be procured is essential to ensure the reliability and economic efficiency of SCED+RP, and an important source of externalities since FRP requirements are set by system operators.

This small numerical example highlights the following lessons. First, a single-period formulation takes myopic decisions that may fail to account for future ramping needs. Second, the performance of SCED+RP is sensitive to the amount of FRP being procured, which must be selected a priori to balance reliability and economic costs. Third, LAD is also sensitive to forecasting errors, especially when future demand is underestimated. Fourth, SLAD is better at hedging against future uncertainty, without requiring perfect information.

2.6 Large-Scale Numerical Experiments

This section describes the large-scale experiments conducted on the French RTE system with 6708 buses. Detailed results are reported in Section 2.7.

2.6.1 Economic Dispatch Formulations

Table 2.4 summarizes the four economic dispatch formulations considered in the experiment. As described in Section 2.3, the SCED, SCED+RP, LAD and SLAD formulations are single-period, single-period with ramping products, multi-period deterministic, and multi-

period stochastic formulations, respectively. In all formulations, time is discretized in 5-minute intervals. The chapter also includes the so-called *perfect dispatch* (PD) [71]. The PD formulation takes as input the actual realization of load and renewable production for the entire day, and solves a LAD problem with a whole day 24-hour horizon. This is the best possible operation that can be attained through economic dispatch. Because it relies on a perfect prediction oracle, it cannot be implemented in practice; instead, PD is used as an *a posteriori* analysis tool [71].

Table 2.4: The Economic Dispatch Formulations Used in Experiments.

Problem	Horizon	# Time steps	# Scenarios	Source of input data
SCED	5 min	1	1	Current measurements
SCED+RP	5 min	1	1	Current measurements & ramping requirements
LAD	60 min	12	1	Deterministic forecast
SLAD	60 min	12	10*	Probabilistic forecast
PD	24 hr	288	1	Perfect forecast

All formulations use 5-minute time steps.

*SLAD considers uncertainty captured by 10 scenarios each for load, wind, and solar.

2.6.2 Experimental Setting

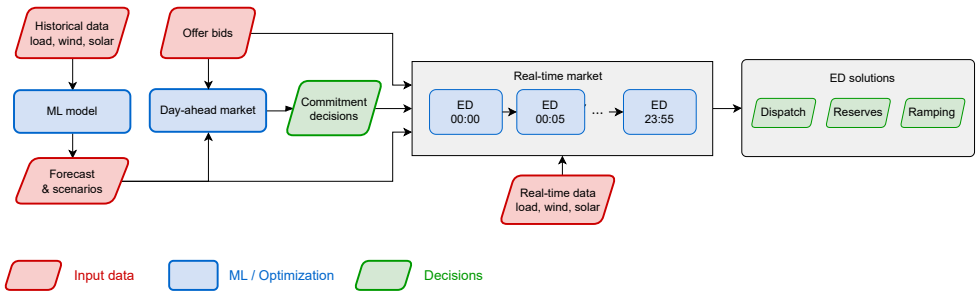


Figure 2.3: The Simulation Pipeline.

The performance of SLAD is evaluated on the over 6,000 buses RTE system for the entire year in 2018. Namely, the chapter executes a simulation for every day of the year, using

a realistic pipeline depicted in Figure 2.3, which reproduces the interaction between day-ahead and real-time markets in the US. The detailed setup of the case study is as follows. The real-time market pipeline is executed over the day, i.e., an economic dispatch problem is solved every 5 minutes, and it outputs the energy dispatch and reserves for each generator which become inputs for the following period.

The time series for load and renewable generation, as well as economic data for the generators, were reconstructed following the disaggregation methods in [72]. The case study sets high penalty costs with load shedding at $100,000/MWh, operating reserve shortages at $50,000/MWh, regulating reserve shortage at $55,000/MWh, transmission violation cost $1,500/MWh, ramping capability shortage at $30/MWh (which is the price used by some ISOs). Artificial generators that represent imports have a high capacity and a cost of $1000/MWh. Therefore, the system should not encounter load shedding or reserve shortages as import generators provide sufficient high-cost energy if needed.

All optimization problems are formulated in JuMP [73] and solved using Gurobi 10.0 with 10 threads. The SCED, SCED+RP and LAD problems employ Gurobi's barrier method (method parameter set to 2). SLAD's accelerated Benders algorithm is implemented in Julia, solving the BD subproblems in parallel—each with one core using the barrier method—and the master problem with 10 threads. Consequently, SLAD uses the same total of 10 threads as SCED, SCED+RP, and LAD. The BD algorithm uses a maximum number of iterations of 100, and a relative gap tolerance of $\epsilon = 10^{-5}$. Experiments are carried out on the Phoenix cluster at the Georgia Institute of Technology, Atlanta, Georgia [21].

2.6.3 Forecast Generation

Deep learning models were used to generate scenarios and forecasts for the RTE system [74]. The DeepAR model [75] is used to generate wind scenarios, and a Temporal Fusion Transformer model [76] is used for solar and load scenario generation. The DeepAR and

TFT models are probabilistic forecasting models, from whose prediction intervals one can sample multiple scenarios. To build confidence-aware prediction sets, we may also leverage more advanced techniques such as generative models [77] or conformal prediction [78, 79] in the future. For LAD, a deterministic forecast is obtained by taking the average of 500 random scenarios. For SLAD, 10 scenarios are considered. The interplay between the quality of the forecasts/scenarios and the performance of LAD and SLAD is further analyzed in Section 2.8.1.

2.6.4 Cost Comparison

The chapter uses the following methodology to evaluate costs and compare the performance of each ED formulation. First, at each time step of a simulation, operating costs are evaluated from generators' energy and reserve dispatch, and import costs. Then, constraint violation penalty costs are computed, which include reserve shortage costs, ramping products shortage costs, transmission violation penalties and energy shortage/surplus penalties. This yields a combination of operating and penalty costs at each time step, which are then aggregated over the day. Finally, recall that each simulation is executed once per ED formulation, which is achieved by using the same random seed when initializing the simulation environment. This ensures that each ED formulation is executed against identical realizations of load and renewable output, thereby enabling a fair comparison.

2.7 Result Analysis

This section presents the results of the large-scale experiment on the RTE system. Section 2.7.1 reports the computational performance of each formulation. Section 2.7.2 compares the economic performance of the various ED formulations, which is further detailed in Sections 2.7.3 and 2.7.3. Finally, Section 2.7.4 provides a detailed analysis of the behavior of SLAD.

2.7.1 Computing Times

Figure 2.4 reports the solution time of each formulation over the year. For each formulation, the figure displays the average computing time for each month. First, observe that computing times are higher during winter, and lower during summer. Indeed, demand is highest during the winter, thereby increasing network congestion and the number of PTDF constraints that must be added to the ED formulations. This naturally increases memory requirements and computing times. Second, most instances are solved within the 5-minute time frame of real-time operations. SCED and SCED+RP instances are typically solved within one second, and LAD instances are solved within 3 to 22 seconds, on average. Similarly, although they require longer computing times, SLAD instances are solved within an average of 15 to 140 seconds. In fact, 99.77% of SLAD instances reach the prescribed 0.001% optimality gap tolerance within 5 minutes. These results confirm that multi-period ED formulations, both deterministic and stochastic, are computationally tractable for real-time market operations.

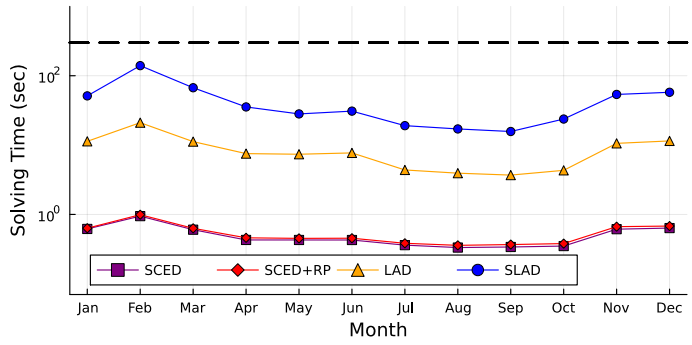


Figure 2.4: Average solution times of ED formulations of each month. The dashed **black** line denotes the 5-minute frequency of real-time market clearing.

2.7.2 Results Overview

Table 2.5: Comparison of daily operating costs in RTE over 2018. Low (resp. high) load days are days where the average load is less (resp. greater than) 50GW. All costs are in M$ per day. Savings are relative to SCED.

	Low load			High load			All		
Formulation	Cost	Savings	Savings(%)	Cost	Savings	Savings(%)	Cost	Savings	Savings(%)
SCED	13.15	0.00	–	34.84	0.00	–	23.79	0.00	–
SCED+RP	13.09	0.06	0.48 %	34.56	0.27	0.78 %	23.62	0.17	0.70 %
LAD	13.02	0.14	1.04 %	34.47	0.36	1.04 %	23.54	0.25	1.04 %
SLAD	**13.01**	**0.15**	**1.11** %	**34.43**	**0.41**	**1.17** %	**23.51**	**0.28**	**1.16** %
PD†	12.98	0.17	1.32 %	34.26	0.57	1.64 %	23.42	0.37	1.55 %

†PD is not practical for real-world application as it presupposes an accurate forecast over the next 24 hours. This is the best possible operation that can be attained through economic dispatch.

Table 2.5 reports, for each ED formulation, the average daily costs and savings over the year. Results are reported for low-load days and high-load days, which are defined as days where the average load is less than 50GW and greater than 50GW, respectively. Note that low- and high-load days follow a seasonal trend, with overall demand being higher in winter and lower in summer. The last three columns in Table 2.5 report results over the whole year. For day d, the daily savings of formulation X compared to SCED are given by

$$\frac{C_d^{\text{SCED}} - C_d^{\text{X}}}{C_d^{\text{SCED}}}, \tag{2.19}$$

where C_d^{X} (resp. C_d^{SCED}) denotes the total costs (including penalties) attained by formulation X (resp. SCED) on day d; positive values indicate net savings. For instance, SCED+RP achieves an average daily cost of 13.09M$ on low-load days, 34.56M$ on high-load days, and 23.62M$ over the year. The corresponding savings relative to SCED are 0.48% on low-load days, 0.78% on high-load days, and 0.70% over the whole year. Specifically, SLAD achieves 65.7% more savings overall compared to SCED+RP and further enhances performance over LAD by incorporating multiple scenarios, rather than depending on a

single deterministic forecast.

Table 2.6 provides an itemized cost comparison between all ED formulations, on three selected days: July 25th, October 20th, and January 30th. For each day and formulation, Table 2.6 reports the total cost of domestic energy production (Energy), total import costs (Import), and the sum of no-load costs, reserve procurement costs and constraint violation penalties (Other). Relative savings compared to SCED are also reported. Figure 2.5 depicts the evolution of import costs with the right-axis depicts the net load in GW over time.

July 25th is a summer day with low demand; the system has ample generation capacity and flexibility, and all formulations achieve almost the same cost. October 20th is a Fall day during which the system experiences moderate net load ramping, especially in the morning and evening hours, thereby requiring more flexibility from the conventional generation fleet. January 30th is a Winter day with high net load and high ramping needs in the morning. This day also experiences higher-than-usual errors in the day-ahead forecast for load and renewables, which results in insufficient commitments and capacity shortages throughout the day. A detailed analysis of this day is presented in Section 2.7.4.

Table 2.6: Itemized daily cost comparison for three representative days in 2018. Cost items and absolute savings relative to SCED are presented in millions of dollars (M$), with savings also expressed as a percentage (%).

Date	Cost item	SCED	SCED+RP	LAD	SLAD	PD†
07/25	Energy	9.74	9.74	9.74	9.74	9.74
	Import	0.00	0.00	0.00	0.00	0.00
	Other†	5.13	5.13	5.13	5.13	5.13
	Total	14.87	14.87	14.87	14.87	14.87
	Savings	0.00	0.00	0.00	0.00	0.00
	Savings(%)	0.00 %	-0.01 %	**0.00** %	**0.00** %	0.01 %
10/20	Energy	8.64	8.66	8.65	8.66	8.65
	Import	0.25	0.25	0.00	0.00	0.00
	Other†	5.11	5.10	5.12	5.11	5.11
	Total	14.00	14.01	13.77	13.77	13.76
	Savings	0.00	-0.01	0.23	0.23	0.24
	Savings(%)	0.00 %	-0.08 %	**1.70** %	1.69 %	1.71 %
01/30	Energy	16.71	16.73	16.74	16.75	16.76
	Import	2.23	1.27	0.87	0.80	0.22
	Other†	10.32	10.31	10.29	10.28	10.26
	Total	29.26	28.31	27.90	27.83	27.24
	Savings	0.00	0.95	1.36	1.43	2.02
	Savings(%)	0.00 %	3.25 %	4.64 %	**4.89** %	6.92 %

†PD is not practical for real-world application as it presupposes an accurate forecast over the next 24 hours.

This is the best possible operation that can be attained through economic dispatch.

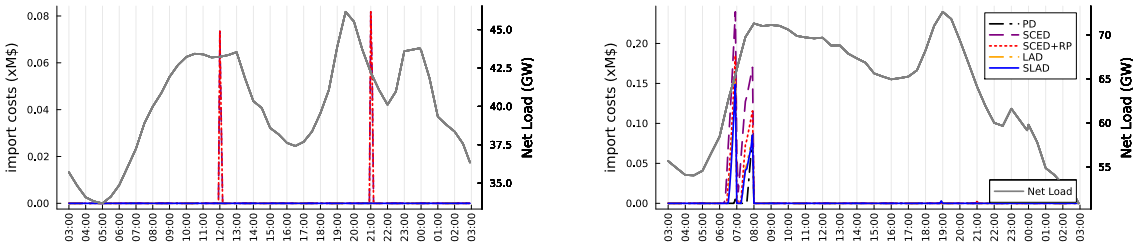


Figure 2.5: Import cost comparison on medium-load (10/20) and high-load (01/30) days. The import cost on the low-load day (07/25) is zero, so the corresponding graph is omitted here.

2.7.3 Individual Factor Analysis

The next sections further analyze the ED formulations by focusing on individual factors. Section 2.7.3 examines the effect of incorporating ramping products, Section 2.7.3 compares single-period versus multi-period formulations, and Section 2.7.3 explores the differences between deterministic and stochastic approaches. These analyses provide a deeper understanding of the elements that influence performance.

Without vs With Ramping Products

This section examines the impact of incorporating ramping products by comparing the SCED model with the SCED that includes ramping products (SCED+RP). As shown in the first two rows of Table 2.5, SCED+RP achieves a cost savings of 0.48% under low load conditions, increasing to 0.78% under high load conditions. On average, incorporating ramping products leads to a 0.70% reduction in costs. These results indicate that, by incorporating ramping products, the system gains additional flexibility to manage net load variability more effectively. The savings are consistent and most significant under high-load conditions, where the need for flexibility is greatest.

Single-Period vs Multi-Period

It is interesting to understand the benefits of using a deterministic multi-period economic dispatch formulation, such as LAD, instead of a deterministic single-period formulation. Note that, as illustrated in the example of Section 2.5, the performance of LAD is affected by the accuracy of the forecast it receives as input; this aspect is further analyzed in Section 2.8.1. Over the year, as noted in Table 2.5, LAD saves an average of 1.04% in daily costs compared to SCED.

Table 2.6 and Figure 2.5 enable a more granular comparison. On July 25th, there is no significant difference in cost between single- and multi-period formulations. On October 20th, the system experiences higher net load ramping, especially in the morning and evening hours, thereby requiring more flexibility from the generation fleet. While LAD is able to accommodate the ramping needs, and achieves the same cost as PD, SCED needs to import energy twice during the day, which results in higher total costs. Finally, recall that, on January 30th, because of errors in the day-ahead forecast used in the unit commitment, the online generation capacity is not enough to meet all energy and reserve requirements. As a result, all formulations import energy during the morning, but again SCED performs the worst, with LAD saving on imports and 4.64% overall compared to SCED. These results demonstrate that the multi-period formulations, compared to the ramping products, better anticipate future ramping needs which, although more expensive in the very-short term, yields savings in the long term.

Deterministic vs Stochastic

Consider now the potential benefits of a stochastic multi-period economic dispatch (SLAD) formulation instead of a deterministic multi-period formulation (LAD). Table 2.5 shows that SLAD saves an average of 1.16% per day in total costs. Savings are seasonal, with SLAD saving an average 1.17% on high-load days, and an average 1.11% on low-load days. These observations are further supported by the results in Table 2.6. On July 25th and

October 20th, SLAD and LAD achieve about the same costs, and match the performance of PD. However, on January 30th, which experiences higher ramping needs, SLAD reduces import costs by 5% compared to LAD, and total costs by 0.25%. Over the year, the savings achieved by SLAD are higher than those of LAD by 12% overall, 14% on high-load days, and 7% on low-load days. This difference demonstrates the benefits of considering multiple scenarios instead of a single point forecast: the former hedges against uncertainty whereas the latter is more sensitive to forecast errors.

2.7.4 Detailed Analysis

This section further investigates the behavior of SCED+RP and SLAD on Jan 30th, a Winter day with a high net load. As shown in Table 2.6 and Figure 2.5, the main difference in costs stems from imports, with SCED+RP incurring 60% more imports between 6am and 8am. Recall that, because all simulations use the same commitment decisions and load/renewable profiles, the differences in costs are purely a consequence of how dispatch decisions are made throughout the day. It is therefore important to understand the underlying factors explaining the difference in behavior between SCED+RP and SLAD. The rest of this section focuses on the morning hours between 5am and 8am, which is when most differences occur.

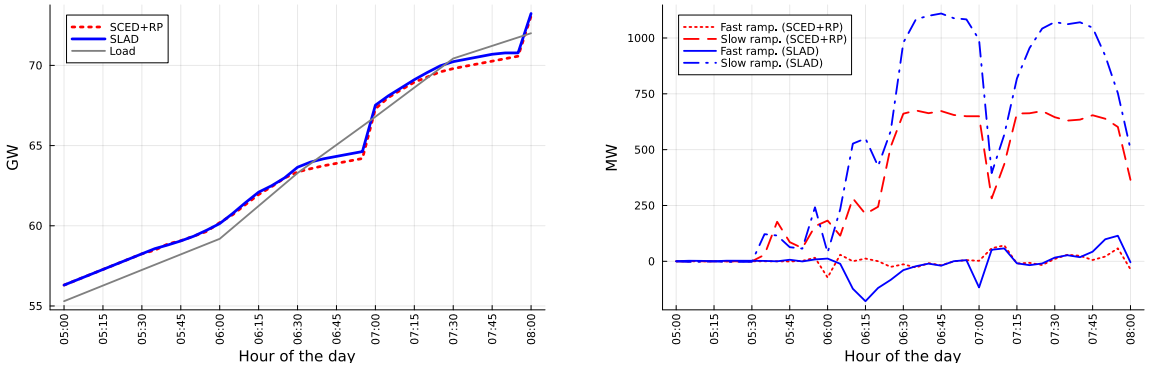


Figure 2.6: Left: Total available capacity of SCED+RP and SLAD between 5am and 8am on January 30th, 2018.
Right: Difference in available capacity between SCED+RP and SCED, SLAD and SCED for slow-ramping and fast-ramping generators, on January 30th, 2018. Slow-ramping (resp. fast-ramping) generators have a ramping rate below (resp. above) 1% of their total capacity per minute. Positive (resp. negative) numbers indicate the available capacity in SCED+RP or SLAD is higher (Resp. lower) than in SCED.

The available capacity of generator g at time t is denoted by $\mathbf{pg}_{g,t}^{\text{avail}}$. It is defined as the maximum amount of power this generator can produce, given its operating parameters and previous (5 minutes earlier) dispatch level. Namely,

$$\mathbf{pg}_{g,t}^{\text{avail}} = \min\left\{\bar{\text{pg}}_{g,t}, \mathbf{pg}_{g,t-1} + \text{r}_{g,t}^{\uparrow}\Delta_t\right\}, \tag{2.20}$$

where $\bar{\text{pg}}_{g,t}, \mathbf{pg}_{g,t-1}$ and $\text{r}_{g,t}^{\uparrow}$ are the generator's maximum output at time t, previous dispatch setpoint, and ramping rate, respectively. Note that a generator's dispatch cannot exceed its available capacity. Therefore, if the load exceeds the total available capacity, there must be imports to balance the system. Also note that imports may still occur even when the total available capacity exceeds loads, in order to satisfy reserve requirements and/or transmission constraints.

The left figure in Figure 2.6 displays the total available capacity of SCED+RP and SLAD between 5am and 8am, as well as the total demand. There are two important observations from the figure. On the one hand, the total load exceeds the available capacity of SCED+RP from 6:30am to 8am, except for a short period of time just after 7am. This corresponds exactly to the window during which SCED+RP imports, thus validating that

available capacity is a useful indicator of the system reliability. On the one hand, SLAD always has a higher available capacity than SCED+RP. The difference is first noticeable at 5:30am, about one hour before SCED starts importing. This delay matches the planning horizon of SLAD, which suggests that SLAD is able to anticipate the possibility of (costly) imports at 6:20am, and therefore takes preemptive action. Nevertheless, SLAD still incurs imports between 6:30am and 7am, albeit a lot less than SCED+RP, because of insufficient online generation capacity at that time. This demonstrates the importance of intra-day commitments for dealing with unforeseen conditions.

To better understand the above results, the right figure in Figure 2.6 displays the difference in available capacity between SLAD and SCED, SCED+RP and SCED over time, disaggregated between slow- and fast-ramping generators. Slow-ramping (resp. fast-ramping) generators are defined as generators whose ramping rate is less than (resp. greater than) 1% of their total capacity per minute. It is clear that the difference between SLAD and SCED+RP mostly stems from slow-ramping generators. The difference in behavior is most apparent after 5:30am, echoing the results of the left figure in Figure 2.6. Finally, the sharp drop in the difference between SLAD and SCED+RP around 7am and 8am is explained by generators starting up at those times.

The sharp difference in available capacity of slow-ramping generators seen in the right figure in Figure 2.6 can be explained by Equation (2.20). Indeed, in the RTE system, slow-ramping generators are mostly nuclear generators with a large (about 1GW) maximum capacity but slow ramping rates. Thus, their available capacity is constrained by ramping, i.e., $\mathbf{pg}^{\text{avail}}_{g,t} = \mathbf{pg}_{g,t-1} + \mathrm{r}^{\uparrow}_{g,t}\Delta_t$ in Eq. (2.20). In contrast, for fast-ramping generators, the relation $\mathbf{pg}^{\text{avail}}_{g,t} = \bar{\mathrm{pg}}_{g,t}$ typically holds. Therefore, at time t, the total available capacity is roughly equal to

$$\mathbf{pg}^{\text{avail}}_{t} = \sum_{g \in \mathcal{G}_s} \left(\mathbf{pg}_{g,t-1} + \mathrm{r}^{\uparrow}_{g,t}\Delta_t \right) + \sum_{g \in \mathcal{G}_f} \bar{\mathrm{pg}}_{g,t}, \tag{2.21}$$

where $\mathcal{G}_f$ and $\mathcal{G}_s$ are slow and fast-ramping generators, respectively.

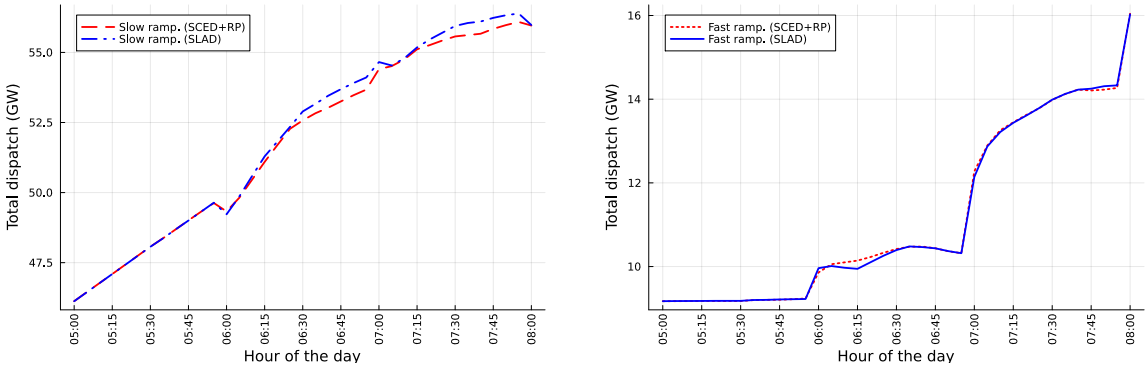


Figure 2.7: Total dispatch comparison between between SCED+RP and SLAD for slow-ramping (left) and fast-ramping (right) generators between 5:00 and 8:00 on January 30th, 2018

From Equation (2.21), it becomes evident that, to increase the total available capacity at time t, one must either (i) commit additional units, or (ii) preemptively increase the dispatch of slow-ramping generators. The former is achieved by intra-day commitments, which is out of the scope of the ED formulations considered in this work. For the latter, Figure 2.7 displays the total dispatch of slow- and fast-ramping generators in SLAD and SCED+RP throughout the study period. For slow-ramping generators, there is a clear difference between SLAD and SCED+RP, with the total dispatch of slow-ramping generators higher in SLAD than in SCED+RP.

Figure 2.8 illustrates SLAD and SCED+RP dispatches of nuclear generator `CHOO2N01` between 5am and 6am. The figure shows that SLAD starts ramping up `CHOO2N01` as early as 5:25am, roughly one hour before imports incur, which corresponds to SLAD planning horizon. SLAD foresees the possibility of having to rely on imports around 6:30am. To avoid these costly imports, SLAD preemptively ramps up several nuclear generators, thereby ensuring enough domestic production in the subsequent hours. This preemptive ramping up of nuclear generators enables SLAD to reduce the duration and volume of imports compared to SCED+RP, thereby realizing substantial savings. SCED+RP starts ramping up the slow-ramping rate generators later than SLAD, partly because the ramping products are designed for only 30 minutes and 10 minutes, durations that are insufficient to

adequately accommodate future ramping needs. Several other nuclear generators display a similar behavior, which accounts for most of the difference in total dispatch of slow-ramping generators between SLAD and SCED+RP.

It is worth noting that SCED+RP, as a single-period formulation, cannot take future commitment decisions into account. Therefore, for example, even if a cheap generator is online in the future, it will not be able to be used to procure current ramping requirements in SCED+RP. However, this information can be foreseen by the multi-period formulation SLAD, which is able to determine whether the ramping needs will be satisfied with the generators coming online.

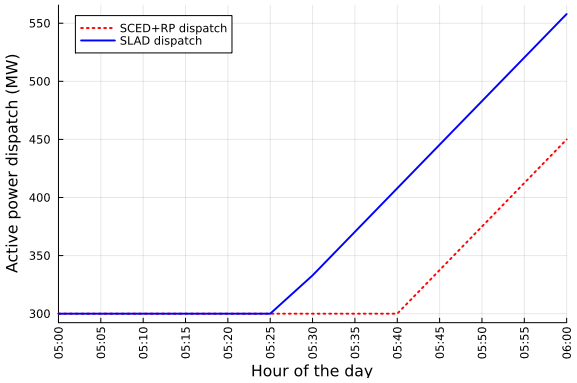


Figure 2.8: Dispatch of nuclear generator `CHOO2N01` from SCED+RP and SLAD solutions

2.8 Sensitivity Analysis

This section further analyzes the sensitivity of SLAD to the input probabilistic forecast, and to the length of the look-ahead horizon.

2.8.1 Impact of the probabilistic forecast

The experiments of Section 2.6 were also executed with a k-Nearest Neighbor (KNN) method to generate scenarios for LAD and SLAD [80]. The kNN model selects the k=10 closest historical scnearios.

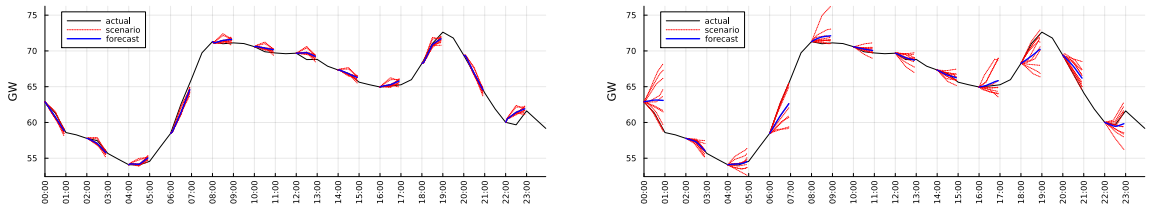


Figure 2.9: Comparison of probabilistic net load forecasts and scenarios on January 30th, 2018. Left: Deep Learning (DL) model. Right: K-Nearest Neighbors (KNN) model. The DL point prediction is more accurate than KNN, and the DL scenarios have lower variance than KNN.

Figure 2.9 illustrates net load probabilistic forecasts produced by the deep learning (DL) models described in Section 2.6.3 on January 30th, 2018. While the accuracy of the DL model's point forecast is higher than that of the KNN model, the scenarios produced by kNN have higher variance, and systematically cover the actual net load.

Table 2.7: Comparison of total operating costs of LAD and SLAD averaging over load level in 2018 using different forecasts and scenarios. All costs are in M$, and all savings are relative to SCED (see Table 2.5).

	Low load			High load			All		
Problem	Cost	Savings	Savings(%)	Cost	Savings	Savings(%)	Cost	Savings	Savings(%)
LAD+DL	13.02	0.14	1.04 %	34.47	0.36	1.04 %	23.54	0.25	1.04 %
LAD+KNN	13.05	0.10	0.77 %	34.75	0.09	0.25 %	23.69	0.09	0.40 %
SLAD+DL	**13.01**	**0.15**	**1.11** %	34.43	0.41	1.17 %	23.51	0.28	1.16 %
SLAD+KNN	**13.01**	**0.15**	**1.11** %	**34.37**	**0.47**	**1.35** %	**23.48**	**0.30**	**1.28** %

Next, Table 2.7 reports total operating costs for LAD and SLAD across the year, using both DL and KNN forecasts. Recall that results obtained with DL forecasts are the same as those reported in Table 2.5. First, when using the point forecast produced by KNN, the performance of LAD significantly degrades: on high-load days, LAD+KNN costs 0.8% more compared to LAD+DL. This highlights the sensitivity of LAD to the quality of the input forecast, i.e., a bad forecast may result in poor decisions. Indeed, shorter-term forecasts are usually more accurate than longer-term ones. Therefore, when forecasting errors are fac-

tored in, increasing the look-ahead horizon in LAD does not guarantee more cost savings. In contrast, the performance of SLAD is less sensitive to the input scenarios. Namely, the year-round average cost of SLAD+KNN is almost identical to that of SLAD+DL, with the former (1.28%) saving 82.9% more compared to SCED+RP, and the latter (1.16%) saving 65.7% more. This result suggests that the performance of SLAD is robust to the quality of the point forecast, as long as scenarios capture actual values.

2.8.2 Impact of the Look-Ahead Horizon

This section explores the impact of the look-ahead horizon on the performance of SLAD. Indeed, recall from Table 2.5 that, over the year, SLAD+DL and PD achieve overall savings of 1.16% and 1.55% compared to SCED, respectively. Also recall that PD cannot be implemented in practice, because it relies on a perfect forecast oracle for the full day. This motivates considering a longer horizon for SLAD, which may improve its overall economic performance.

To that end, the chapter evaluates the performance of SLAD with various look-ahead horizons on January 30th, 2018. The chapter also considers, as a baseline, a LAD formulation with the same horizon as SLAD, but seeded with a perfect forecast; this formulation is referred to as Perfect LAD (PLAD). It is important to recognize that PLAD formulations are also idealized as they use perfect forecasts and forecasts always incur errors in practice. Nevertheless, PLAD allows to evaluate the value of a perfect prediction oracle compared to SLAD under the same look-ahead horizons.

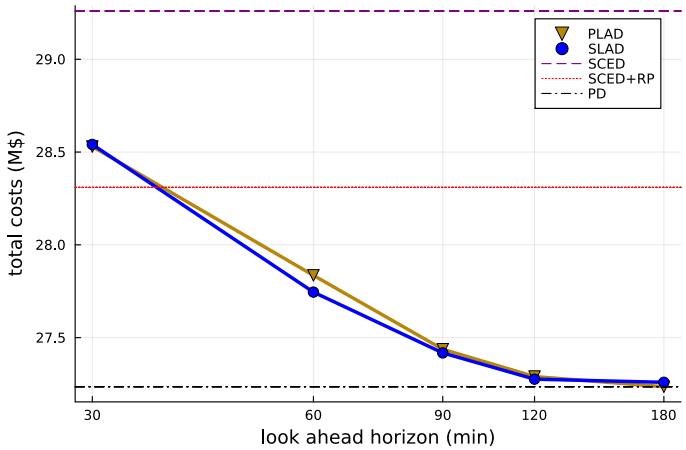


Figure 2.10: Comparison of SLAD and PLAD with different look-ahead horizons on January 30th, 2018. The black dashed line depicts the (unattainable) performance of the idealized Perfect Dispatch (PD). The red dashed line depicts the performance of SCED with ramping products (SCED+RP). The purple dashed line depicts the performance of SCED.

The experiments are conducted with a look-ahead ranging from 30 minutes to 3 hours (180 minutes), using scenarios produced by the KNN model. Figure 2.10 illustrates each formulation's total operating costs as a function of the look-ahead horizon. The figure also depicts the overall costs of PD as an (unreachable) baseline. First, as expected, the total operational cost of SLAD and PLAD decreases as the look-ahead horizon increases. Second, there is no significant difference in cost between PLAD with a 3-hour look-ahead and PD. This result suggests that, for the system at hand, a three-hour horizon is long enough to make nearly optimal dispatch decisions. Third, the performance of SLAD is almost identical (and sometimes exceeds) to that of PLAD, and almost matches that of PD when a 3-hour horizon is considered. This demonstrates that the proposed SLAD formulation can produce high-quality solutions without the need for a perfect forecast.

Finally, increasing the look-ahead horizon in SLAD does increase its computational burden, namely, it leads to higher memory requirements and computing time. Such trade-offs should thus be considered when determining the length of the look-ahead horizon.

2.9 Conclusion

This chapter studied the benefits of stochastic look-ahead dispatch in the context of the MISO's market-clearing pipeline, evaluated on the France RTE network. The chapter showed that multi-period formulations can yield significant benefits over a single-period formulation. Moreover, a stochastic formulation (SLAD) brings additional benefits, meets the market requirements in terms of computing speed, and may even outperform a multi-period LAD using a perfect forecast for the same horizon. The primary benefit of multi-period formulations is their ability to anticipate ramping events, reducing import costs and transmission line violations. The chapter also showed that SLAD bridges a significant portion of the gap between SCED and the potential benefits of a perfect dispatch over 24 hours. An interesting consequence of these results concerns the proliferation of ramping products in existing markets. Multi-period formulations should eliminate, or at least substantially decrease, the need for such ramping products.

In terms of future directions, one promising area for improvement is to further accelerate the SLAD solving process. Additionally, as renewable generation penetration and storage units continue to increase in power grids, applying SLAD to a system with a larger proportion of renewable generation capacity and storage could provide valuable insights for the future. Another potential direction is to add intra-day unit commitment to the pipeline, which could potentially eliminate some of the import costs highlighted in the chapter and lead to a more accurate representation of the proposed stochastic approach. Finally, a systematic comparison of stochastic and robust formulations in terms of computational efficiency, economic performance, and reliability, would be valuable.

CHAPTER 3
BOUND TIGHTENING USING ROLLING-HORIZON DECOMPOSITION FOR NEURAL NETWORK VERIFICATION

3.1 Introduction

Neural networks are being applied in critical systems and high-consequence decision-making settings, e.g., power systems [81] and autonomous driving [82]. How can we trust these models when the stakes are too high, when the price of failure is prohibitive or even life-threatening? To justify trust, these models need to provide robustness guarantees. For example, in the context of power grid applications, a guarantee that a slight change in input (power system state) will not lead to unreasonable fluctuations in output (control actions predicted by the network). Mathematical optimization can provide such guarantees. While local optimization methods are acceptable for training these models, global optimality—i.e., a formal certificate that no better solution exists—is needed to provide robustness guarantees. There has been significant recent interest in providing verifiable properties for neural networks using global methods such as MIP [83, 84, 85]. MIP solvers are appealing because they can, in principle, perform *complete* verification for many network architectures. Unfortunately, global optimization methods come with a hefty computational price tag, and using off-the-shelve solvers is not a scalable approach.

Improving the scalability of global methods for neural network verification is an active research area, with approaches such as mathematical reformulations [86], cutting planes [87], zonotopes [88], custom-built branch-and-bound algorithms and relaxations [89, 90, 91, 92, 93, 94, 95, 96], to name a few. In 2020, a community-driven effort led to the creation of the VNN competition [97], which has been held yearly since. The competition's goal is to "allow researchers to compare their neural network verifiers on a wide set of bench-

marks". The α, β-CROWN team has consistently won this competition since its inception [91, 92, 93, 94, 95, 96]. In this chapter, we are hoping to bring the use of MIP for neural network (NN) verification one step closer to viability. For this purpose, we tackle the verification problem as defined in [97], using a MIP decomposition approach combined with OBBT [98]. OBBT is extensively used in global optimization solvers to reduce variable domains, especially for nonconvex mixed-integer nonlinear programs (MINLPs) [99]. The OBBT algorithm solves two auxiliary optimization problems for each decision variable. In its initial form, OBBT relies on convexifying the feasible region and using variables' bounds as objective functions. In this work, we propose to preserve the mixed-integer nature of the subproblems, leveraging instead our proposed rolling horizon decomposition. One nice property of OBBT is its amenability to parallelization, since each auxiliary problem can be run independently. We take advantage of parallelization along with other speedup methods such as early termination using cutoff values as outlined in our approach below.

3.2 Problem Statement

In the domain of neural network verification, a "white-box" model is given, granting full visibility into the network's architecture and parameters [97]. Our verification challenge, based on the framework established by Bunel et al. [82], is to ascertain whether a neural network, denoted as function f with L layers, produces outputs that satisfy a desired property P for all inputs within a specified range $\mathcal{C}$.

Formally, we verify that for any input $\mathbf{x}_0 \in \mathcal{C}$, the network's output $\mathbf{y}^{(L)}$ adheres to the property $P(\mathbf{y}^{(L)})$, encapsulated by the implication:

$$\mathbf{x}_0 \in \mathcal{C} \Rightarrow P(\mathbf{y}^{(L)}).$$

For instance, in assessing local robustness, we determine whether all inputs within an ϵ-ball around a data point a with label y_a are classified as y_a by the network. This property

is widely used in image classification cases, where we assess if a network's output label remains consistent under perturbations within a small tolerance.

3.2.1 MIP Encoding for Trained Neural Networks

Transitioning neural network architectures into a MIP format is key for verification. This process, in line with Tjeng et al.'s [100] methodology, enables the application of mathematical programming for thorough network analysis. The MIP model thus becomes a crucial tool for effective verification strategies.

Consider a neural network with an input vector $\mathbf{x}^{(0)} := \mathbf{x}_0 \in \mathbb{R}^{n_0}$. In this network, n_i represents the number of neurons in the i-th layer. The network consists of L layers, where each layer i has an associated weight matrix $W^{(i)} \in \mathbb{R}^{n_i \times n_{i-1}}$ and a bias vector $b^{(i)} \in \mathbb{R}^{n_i}$, for $i \in \{1, \ldots, L\}$. Let $\mathbf{y}^{(i)}$ denote the pre-activation vector and $\mathbf{x}^{(i)}$ the post-activation vector at layer i, with $\mathbf{x}^{(i)} = \sigma(\mathbf{y}^{(i)})$. The output of the network is $\mathbf{y}^{(L)}$. Although σ could be any activation function, we will assume it is the Rectified Linear Unit (ReLU) function throughout this chapter. Figure 3.1 presents a fully connected neural network with ReLU activation functions.

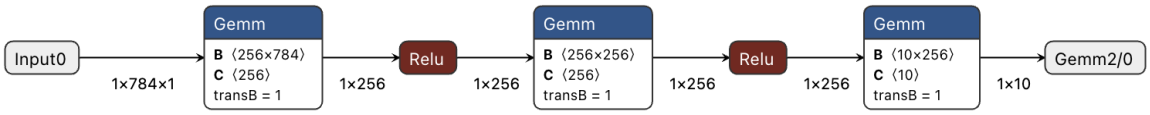


Figure 3.1: A fully connected neural network with ReLU activation functions, showcasing the architecture used for MNIST digit classification.

In network verification, we typically define f such that a non-negative outcome of $f(\mathbf{y}^{(L)}) \geq 0$ indicates the satisfaction of the desired property. Therefore, if an adversarial input $\mathbf{x} \in \mathcal{C}$ results in a negative value of f, this is taken as evidence that the verification instance fails to meet the required property, providing a counter-example. In short:

$$\begin{cases} \text{If } \exists\, \mathbf{x}^{(\mathbf{0})}_{\text{adv}} \in \mathcal{C} \text{ such that } f(\mathbf{y}^{(\mathbf{L})}_{\text{adv}}) < 0, \text{then } P(\mathbf{y}^{(\mathbf{L})}_{\text{adv}}) \text{ does not hold.} \\ \text{If } \forall\, \mathbf{x}^{(\mathbf{0})} \in \mathcal{C}, f(\mathbf{y}^{(\mathbf{L})}) \geq 0, \text{then } P(\mathbf{y}^{(\mathbf{L})}) \text{ holds.} \end{cases}$$

The optimization problem is as follows:

$$\begin{aligned} \min \quad & f(\mathbf{y}^{(L)}) \\ \text{s.t.} \quad & \mathbf{y}^{(i)} = W^{(i)}\mathbf{x}^{(i-1)} + b^{(i)}, \qquad \forall i \in \{1, \ldots, L\}, \\ & \mathbf{x}^{(i)} = \sigma(\mathbf{y}^{(i)}), \qquad \forall i \in \{1, \ldots, L-1\}, \\ & \mathbf{x}^{(0)} \in \mathcal{C}. \end{aligned} \tag{3.1}$$

Specifically, if all operators within the trained neural network are piecewise-linear, then the neural network can be linearly represented within the mixed-integer linear programming (MILP) framework. If the activation function is a ReLU function, the MILP formulation of $x = \sigma(y) = ReLU(y) = \max(0, y)$ is given by:

$$x \geq 0,\ x \geq y,\ x \leq y - l(1-z),\ x \leq u \cdot z,\ z \in \{0, 1\}$$

where l, u are the respective lower and upper bound on y. The binary variable z indicates the activation state of the ReLU.

A ReLU neuron unit can be classified into different categories based on its input domain $[l, u]$: it is deemed "Inactive" if $u \leq 0$, "Active" if $l \geq 0$, and "Unstabilized" when $l < 0$ and $u > 0$. The neuron is called "Stabilized" if it meets either the Active or Inactive condition.

3.2.2 Bound Tightening

In practice, solving the mixed-integer program (3.1) can be computationally prohibitive for large neural networks. As a result, it has been proven effective to introduce additional bounds on intermediate layers. Specifically, the problem states

$$\begin{aligned} \min \quad & f(\mathbf{y}^{(L)}) \\ \text{s.t.} \quad & \text{Constraints of model (3.1)}, \\ & \mathbf{x}_l^{(i)} \leq \mathbf{x}^{(i)} \leq \mathbf{x}_u^{(i)}, \quad \forall i \in \{1, \ldots, L-1\}, \\ & \mathbf{y}_l^{(i)} \leq \mathbf{y}^{(i)} \leq \mathbf{y}_u^{(i)}, \quad \forall i \in \{1, \ldots, L\}. \end{aligned} \tag{3.2}$$

In model (3.2), $\mathbf{y}_l^{(i)}$ and $\mathbf{y}_u^{(i)}$ denotes the vector lower and upper bound respectively for pre-activation output $\mathbf{y}^{(i)}$ at layer i; we similarly define these bounds for post-activation $\mathbf{x}^{(i)}$. Hence, the additional constraints form hyper-rectangles around the outputs. It is crucial to note that the bounds on $\mathbf{y}^{(i)}$ play a much more important role than those on $\mathbf{x}^{(i)}$, as they serve as the input to the ReLU layer and consequently determine the stabilization of the ReLU neuron. When a ReLU is stabilized, the binary variable indicating its active state becomes fixed to either 0 (inactive) or 1 (active). This stabilization leads to a reduction in the number of binary variables in the problem.

There are several methods to derive bounds for intermediate layers:

- **Interval bound propagation (IBP):** This method employs interval bound propagation to establish bounds for each layer [101].

- **DeepPoly:** This method uses a custom polyhedral abstract domain relaxation. It assigns concrete lower and upper bounds to every neuron in a neural network. Symbolic bounds are formulated as linear combinations of the neurons in the network's previous layer.[102].

- **CROWN:** This method efficiently leverages linear bound propagation to adaptively determine the lower and upper bounds of neural networks [89].

- **α-CROWN:** This approach builds upon and enhances CROWN, further tightening the linear bounds by utilizing gradients [92].

- **Optimization-Based Bound Tightening (OBBT):** This approach represents the exact bound tightening technique, which involves solving MIP problems [98].

The following remarks can be made about the above methods. First, it is widely accepted that IBP, while simple and fast, yields very weak bounds, especially for deep networks. Second, DeepPoly and CROWN are based on the same polyhedral relaxations, and therefore yield the same bounds [90]. Third, because α-CROWN augments CROWN

with a gradient-based procedure, it achieves (both in theory and in practice) tighter bounds than CROWN [90, 92]. Finally, as noted in [90], because IBP, CROWN/DeepPoly and α-CROWN are all based on polyhedral (linear) relaxations, they cannot break the convex relaxation barrier described in [90], though α-CROWN matches this theoretical limitation. Note that the convex relaxation barrier presented in [90] is equivalent to LP-based OBBT, wherein bounds are tightened iteratively by solving only the linear relaxation of each OBBT problem. Therefore, in order to further tighten bounds, any procedure must explicitly consider the binary variables associated to each ReLU neuron.

In our work, the primary objective is to obtain tighter bounds. To achieve this, we utilize optimization-based bound tightening, focusing on achieving the tightest box bounds for intermediate layers.

3.3 Methodology

We introduce the Optimization-Based Bound Tightening with Rolling Horizon (OBBT-RH) here. The method builds upon the MIP-based Optimization-Based Bound Tightening approach, as tighter bounds for intermediate layers in model (3.2) will in general result in faster solve times. Therefore, applying tighter bounds can speed up the verification process, allowing it to scale to larger models.

OBBT is formulated as two optimization subproblems for each neuron, seeking to find its maximum and minimum bound. Specifically, let $\mathbf{y}_k^{(t)}$ denote the k-th neuron at layer t subject to network constraints. Given $0 \leq s < t \leq L$, the problem states:

$$\begin{aligned}
\text{max/min} \quad & \mathbf{y}_k^{(t)} \\
\text{s.t.} \quad & \mathbf{y}^{(i)} = W^{(i)}\mathbf{x}^{(i-1)} + b^{(i)}, && \forall i \in \{s+1, \ldots, t\}, \\
& \mathbf{y}_l^{(i)} \leq \mathbf{y}^{(i)} \leq \mathbf{y}_u^{(i)}, && \forall i \in \{s+1, \ldots, t\}, \\
& \mathbf{x}^{(i)} = \sigma(\mathbf{y}^{(i)}), && \forall i \in \{s+1, \ldots, t-1\}, \\
& \mathbf{x}_l^{(i)} \leq \mathbf{x}^{(i)} \leq \mathbf{x}_u^{(i)} && \forall i \in \{s, \ldots, t-1\}.
\end{aligned} \tag{3.3}$$

However, as the number of neural network layers included in the OBBT instances (3.3) increases, the OBBT process becomes intractable. To address this issue, we propose a decomposition method in OBBT-RH to reduce the number of layers considered in each OBBT instance.

The proposed OBBT-RH in Algorithm 6 leverages problem (3.3) to present an effective method for the verification of deep neural networks. OBBT-RH sequentially decomposes the neural network into manageable problem size, focusing on smaller sub-graphs of the original network. This method hence applies optimization-based bound tightening within a rolling horizon framework, effectively balancing the achievement of tighter bounds against maintaining the tractability of the mixed-integer programming problems involved.

Algorithm 6 OBBT-RH

Require: Rolling horizon sequence $\mathcal{S} = \{(s_j, t_j) \mid j \in [J]\}$ where J is a positive integer

for $j = 1, \ldots, J$ **do**

for each neuron $k = 1, \ldots, n_i$ **do**

Solve problem (3.3) to obtain max and min values of $\mathbf{y}_k^{(t_j)}$.

end for

end for

Ensure: Lower/upper bounds $\{\mathbf{y}_l^{(i)}\}, \{\mathbf{y}_u^{(i)}\}$ for all $i = 1, \ldots, L$.

3.3.1 Rolling Horizon Sequence

We introduce a rolling horizon strategy for selecting sequences $\mathcal{S}$ of layers within a neural network for bound tightening. The rolling horizon length, denoted as H, determines the maximum number of General Matrix Multiplications (Gemm) layers to be included in each sub-graph. We begin with an initial pair (s_1, t_1) representing the starting and ending layers of the window, where s_1 is the starting layer and t_1 is the tightening layer. Each subsequent pair (s_j, t_j) in the sequence $\mathcal{S}$ is constructed by shifting the window toward the end layer, with H guiding the span of the window and indicating the layers selected for bound tight-

ening in each iteration. For instance, considering the ReLU neural network, we selected $t_j = \min(j+1, L-1)$ and $s_j = \max(0, t_j - H)$. Indeed, the choice of s_j and t_j can be adjusted for various neural network architectures, highlighting the inherent flexibility of our approach.

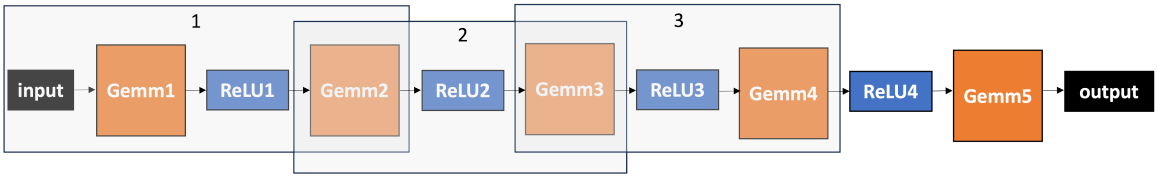


Figure 3.2: OBBT-RH with horizon length 2: $\mathcal{S} = \{(0,2),(1,3),(2,4)\}$, meaning that $(s_1,t_1) = (0,2), (s_2,t_2) = (1,3), (s_3,t_3) = (2,4)$.

The end goal is to tighten the neurons' bounds right before ReLU layers, thus each t in the sequence is a layer that precedes a ReLU layer. Figure 3.2 shows the OBBT-RH with horizon length $H = 2$ for a neural network with 5 layers. For $H = 2$, the sequence $\mathcal{S} = \{(0,2),(1,3),(2,4)\}$ includes pairs (Input, Gemm2), (Gemm1, Gemm3), and $(\text{Gemm2}, \text{Gemm4})$, each ending just before a ReLU layer. As H increases to 3, the sequence $\mathcal{S} = \{(0,2),(0,3),(1,4)\}$ includes pairs $(\text{Input}, \text{Gemm2})$, $(\text{Input}, \text{Gemm3})$ and $(\text{Gemm1}, \text{Gemm4})$, and for $H = 4$, $\mathcal{S} = \{(0,2),(0,3),(0,4)\}$ and it includes $\{(\text{Input}, \text{Gemm2}), (\text{Input}, \text{Gemm3}), (\text{Input}, \text{Gemm4})\}$, now encompassing all Gemm layers leading to a ReLU operator.

The length of the rolling horizon has direct impact on bound tightness and computational time: A longer horizon considers more layers simultaneously, which could help in stabilizing more ReLU neurons and thus simplifying the MIP problem. However, this benefit comes at the cost of increased computational resources. A shorter horizon will generally require less computation, beneficial for efficiency. However, it loses information in the previous layers and produces looser bounds.

One important advantage of this rolling horizon approach is the tightened bounds for neurons in previous layers can speed up the tightening process for neurons in subsequent layers. Note that IBP is equivalent to OBBT for the first ReLU layer if the intermediate

layer is linear. We take advantage of this observation to avoid building sub-MIPs leading to this layer.

3.3.2 Early Termination in the Case of ReLU Activation

Maximizing Neuron Output: When solving the MIP (3.3) to maximize the output of a neuron $y_k^{(t)}$, if the upper bound reaches zero, it indicates that the neuron will be inactive ($y_k^{(t)} \leq 0$). Hence, further bound tightening is unnecessary, and the process can be terminated early.

Minimizing Neuron Output: Similarly, when minimizing the output of a neuron, if the lower bound exceeds zero, it implies that the neuron will remain active ($y_k^{(t)} > 0$). In this case, the OBBT process can also be terminated early.

In either case, the ReLU neuron is stabilized. Therefore, introducing such an early termination criterion speeds up the bound tightening process. In this chapter, we simplify our model by assuming a linear structure for the neural network, wherein each layer is dependent solely on its immediate predecessor. However, real-world neural networks often exhibit more complex dependencies, where a given layer may depend not just on its immediate predecessor, but also on layers further back in the sequence. In such cases, the decomposition process becomes more complicated. We propose using Breadth-First Search (BFS) to navigate these complex dependencies. BFS can effectively trace the shortest path from the final layer t to an initial layer s, ensuring that all relevant inter-layer dependencies, including those extending over multiple layers, are appropriately captured for conducting bound tightening of intermediate layers.

3.3.3 Parallelization of OBBT sub-MIPs

We take advantage of the graph-structure of neural networks by recognizing that the bounds on each neuron in a given layer can be independently tightened. This observation allows

us to distribute the computational effort of solving sub-MIPs corresponding to each neuron using independent threads in parallel. Our implementation uses both the Message Passing Interface (MPI) for multi-machine clusters as well as single-machine multi-threading to parallelize our rolling-horizon algorithm.

3.4 Numerical Results

Table 3.1: Average number of neurons (inactive, active, stabilized, unstabilized) for each method in the benchmark.

Method	Inactive	Active	Stabilized	Unstabilized
IBP	118.21	4.21	122.42	633.53
CROWN	435.67	18.72	454.39	301.56
α-CROWN	466.67	19.93	486.60	269.35
OBBT	621.95	31.91	653.86	102.09
OBBT-RH	**622.25**	**31.93**	**654.18**	**101.78**

3.4.1 Setup

Benchmark dataset Numerical experiments were conducted on all 90 instances from the `mnist_fc` benchmark within the VNN-COMP benchmark library [97].

Comparison metrics We primarily focus on the following metrics. First, the number of ReLU neurons stabilized (Table 3.1). Second, bounds range of ReLU neurons input (Table 3.2). Third, LP relaxation bounds of the MIPs with bounds on intermediate layers (Table 3.3). Lastly, the proportion of instances that can be verified and their computing times (Table 3.4).

Implementation details The `auto_LiRPA` package is used to calculate IBP, CROWN, and α-CROWN bounds. A GPU V100 is used for generating these bounds. Note that the

code of `auto_LiRPA` returns errors for five instances (51, 54, 60, 76, 85) when computing the α-CROWN bounds. Therefore, for a fair comparison, these instances have been removed. The authors were not able to install DeepPoly and its required dependencies; this method is therefore not included in the results. Nevertheless, recall that DeepPoly and CROWN employ the same relaxations and thus yield numerically identical bounds [90]. The optimization problems were formulated with mathematical modeling tool Gravity (https://github.com/coin-or/Gravity) in C++ and solved using Gurobi 10.0.2. The experiments are carried out on the High-Performance Computing (HPC) platform provided by the Partnership for an Advanced Computing Environment (PACE) at Georgia Institute of Technology, Atlanta, Georgia. Within this benchmark, the setting for OBBT-RH horizon length is dynamically adjusted based on the network architecture. Specifically, for neural networks with three layers, the rolling horizon length H is set to two. In the case of networks with five layers, H is set to three, while for those with seven layers, H is set to five. OBBT can be seen as a special case of OBBT-RH, wherein the horizon length is equal to the depth of the neural network.

Additionally, during the initialization phase of OBBT-RH instances, IBP bounds are applied to intermediate layers. To further enhance efficiency, a parallel approach is employed for tightening the input of each ReLU layer. Given that each ReLU layer in this benchmark comprises 256 neurons, a total of 512 CPU threads are requested, with 2 threads used for each OBBT-RH instance. This strategy of parallelization enables the independent and simultaneous tightening of bounds for each ReLU layer, significantly boosting the process's overall efficiency. In addition, since the goal of each OBBT-RH instance is to tighten the bounds, we have set Gurobi's MIPFocus parameter to value 3 to enhance the quality of the bounds. Moreover, to avoid excessively long solving times for some instances, a 30-second time limit is enforced on each OBBT-RH instance. Therefore, with the need to solve both maximization and minimization problems for unstabilized neuron, the maximum time required for OBBT-RH in one layer is one minute.

3.4.2 Comparison with other bound tightening methods

Table 3.2: Average bounds range for each method in the benchmark.

Method	Bounds Range
IBP	2028.35
CROWN	96.16
α-CROWN	53.52
OBBT	12.87
OBBT-RH	**12.38**

The aim is to compare the bound tightness produced by various methods. Specifically, the goal is to maximize the number of stabilized neurons, thereby reducing the count of binary variables. Additionally, for those unstabilized neurons, the approach seeks to tighten their input bounds as much as possible, aiding in the stabilization of subsequent layer neurons. After determining the bounds preceding the ReLU layers, the LP relaxation bounds of the MIPs will be compared with the intermediate layers' generated bounds. Lastly, the evaluation will focus on the end-to-end comparison of the number of verified instances as well as the time taken across different methods.

In terms of results, Table 3.1 presents the average number of neurons categorized as inactive, active, stabilized and unstabilized for different bound tightening methods. The OBBT-RH method demonstrates a higher average across all layers from all instances. Table 3.2 shows the average bounds range for each verification method in a benchmark setting. OBBT-RH outperforms the other methods with the tightest bounds, indicating its effectiveness in bounding neuron activations within the network. Table 3.3 presents the average LP relaxation bounds for MIPs using the bounds produced by each method. These LP relaxation bounds indicate the tightness of the solution space. OBBT shows the least negative value, implying a closer approximation to the MIP's optimal solution, whereas OBBT-

RH has a slightly higher value than OBBT. Table 3.4 includes the end-to-end verification results by comparing the efficiency of MIP-based verification where variable bounds are computed using different bound tightening methods. BT stands for the bound tightening time and MIP stands for the final MIP solving time with the tightened bounds. A total time limit of 20 minutes is set for each approach. Overall, MIP with OBBT-RH bounds verifies the most instances (either by proving robustness or finding an adversarial example) than other approaches, and achieves a 2.2x speedup over CROWN-based methods. In addition, OBBT-RH yields 1.25x speedup over OBBT, mostly thanks to shorter bound-tightening times.

Table 3.3: Average LP relaxation bounds of the MIPs for each method in the benchmark.

Method	LP Bounds
IBP	-25232.25
CROWN	-1169.45
α-CROWN	-577.91
OBBT	**-0.64**
OBBT-RH	-24.77

Table 3.4: End-to-end comparison on all instances. All time are in seconds.

			Time (sec)		
Method	#Verified	#Timeout	BT	MIP	Total
IBP+MIP	37	48	**3.54**	708.60	712.14
CROWN+MIP	62	23	**3.54**	345.89	349.43
α-CROWN+MIP	61	24	4.93	340.74	345.66
OBBT+MIP	79	6	121.20	81.07	202.27
OBBT-RH+MIP	**80**	**5**	90.19	**64.66**	**154.85**

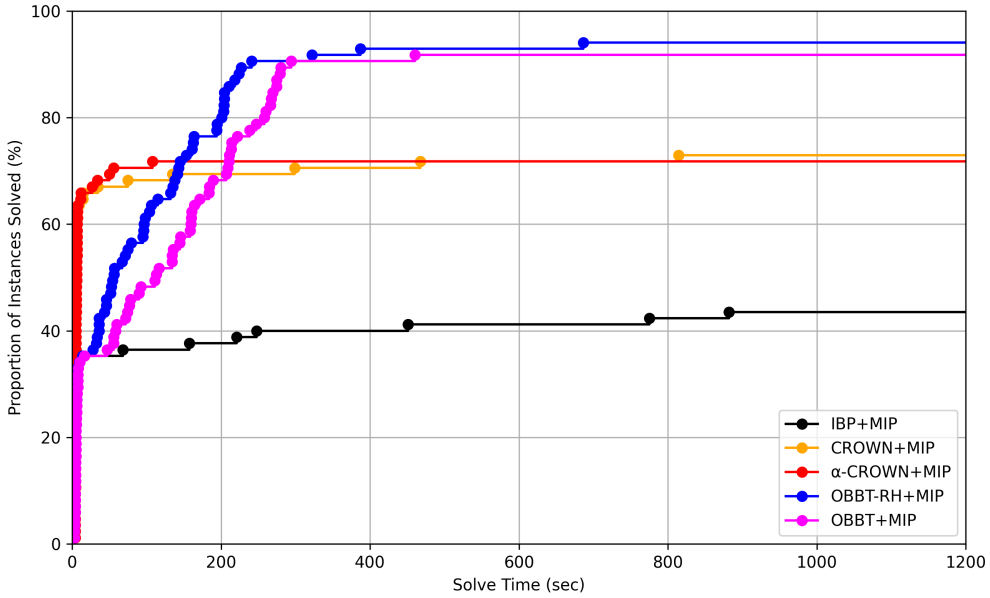


Figure 3.3: Performance comparison of the MIP-based verification for different bound-tightening methods.

3.4.3 Solving MIP with tight bounds

After computing bounds for the intermediate layers, we use these bounds to solve the MIP instances. This is achieved by first obtaining the variable bounds for intermediate layers and then constructing the MIPs using the model as outlined in (3.2) for each verification instance. In the verification context, tightening the domain of the input for ReLU activations strengthens the MIP formulation by decreasing the big-M coefficients. Furthermore, it can stabilize ReLU units—eliminating binary variables—when a ReLU's input is proven to be always non-negative or always non-positive. This combined effect—yielding stronger relaxations and reducing the number of binary variables—results in substantial improvements when solving the final MIPs corresponding to the verification problems with the tightened bounds. In addition to incorporating the tight bounds into the MIP, we also set the Gurobi cutoff parameters to 0. This parameter setting is critical as it ensures that nodes with a lower bound greater than 0 are pruned, thereby enhancing the efficiency of the verification

process.

The end-to-end results are shown in Table 3.4, with OBBT-RH outperforming the other methods as discussed above. Figure 3.3 shows the performance of various methods in verifying instances within the benchmark, highlighting the trade-off between tight bounds and verification time. For low time budgets, fast bound tightening methods such as IBP, CROWN, and α-CROWN are more effective, verifying a greater number of instances quickly. However, as the time budget increases, OBBT-RH and OBBT, which prioritize tighter bounds, begin to outperform the faster methods by verifying a higher proportion of instances. Notably, OBBT-RH is more efficient than OBBT, verifying more instances in less time. Moreover, our approach has successfully closed several challenging instances (48, 78, 83) that previously posed difficulties for state-of-the-art complete verifiers. With our proposed method for generating OBBT-RH bounds, we successfully closed instance 48 within five minutes, a feat not achieved by any complete verifiers from the VNN competition.

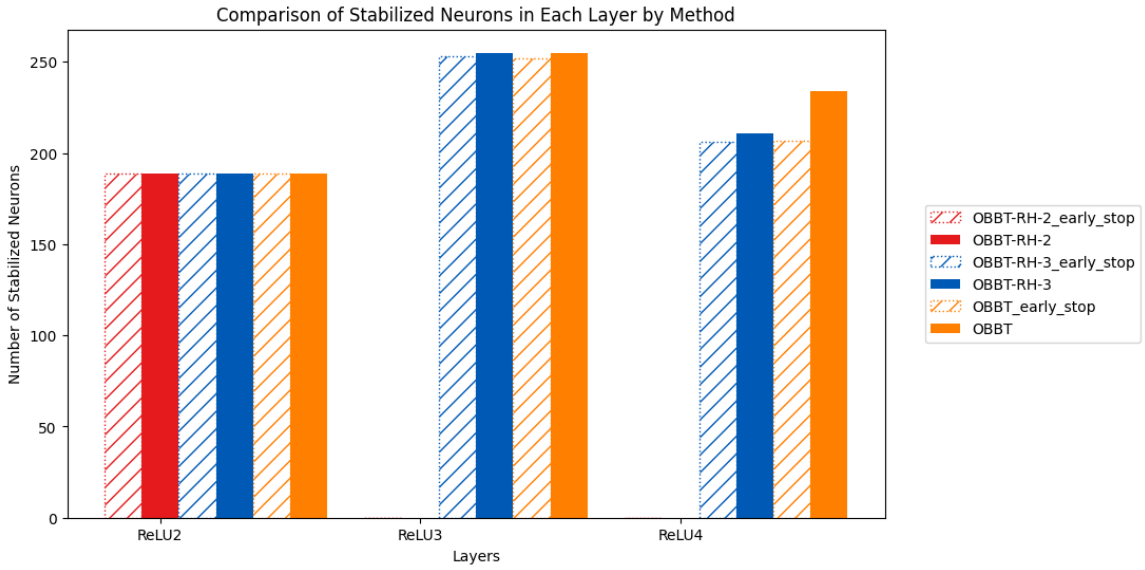


Figure 3.4: Comparison of stabilized neurons in each layer by different methods on instance 48.

3.4.4 Sensitivity Analysis 1: length of rolling horizon

To assess the impact of horizon length on the performance of OBBT-RH, we conducted a sensitivity analysis. Due to the high computational cost of solving each OBBT instance to optimality, the sensitivity analysis was only conducted on instance 48. The analysis involved varying the horizon length of OBBT-RH, denoted as OBBT-RH-H, where H represents the horizon length. For instance, OBBT-RH-3 indicates an OBBT-RH variant with a horizon length of 3. Figure 3.4 compares the number of stabilized neurons across each layer by OBBT-RH with varying horizon lengths. The analysis reveals a clear trend: OBBT-RH-3 demonstrates a significant increase in the number of stabilized ReLU units compared to its counterpart, OBBT-RH-2, which employs a rolling horizon of length 2. OBBT-RH-2 stabilizes 0 ReLUs in the 3rd and 4th ReLU layers. It should also be emphasized that, considering early-stopping, OBBT-RH-3 stabilizes almost as many neurons as OBBT. This suggests that our proposed MIP-based approach, with an appropriately set rolling horizon, can effectively enhance the tightness of neuron bounds.

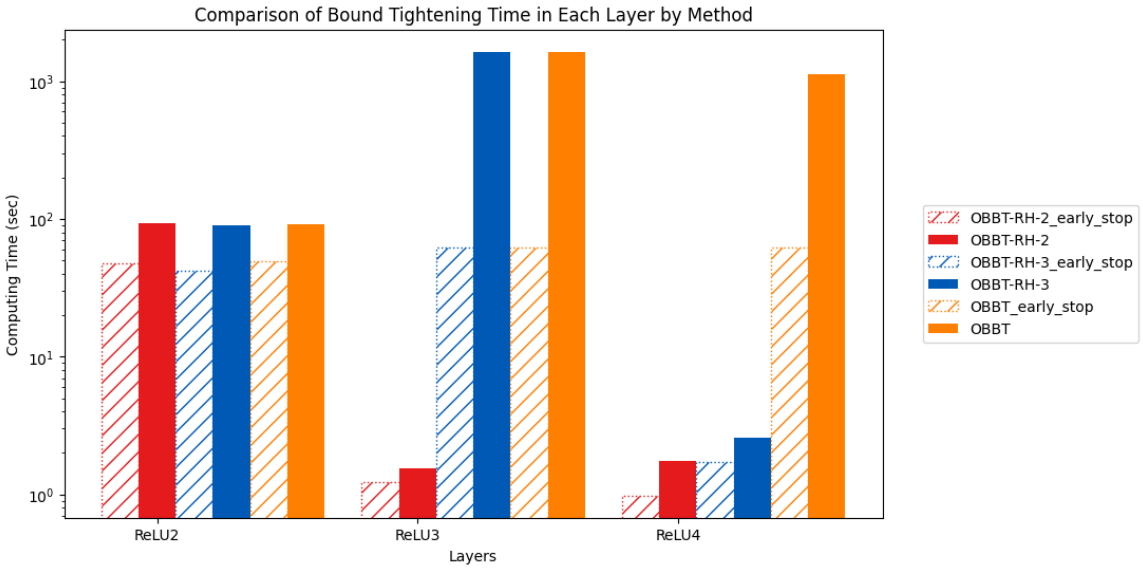


Figure 3.5: Comparison of bound tightening time for each layer by different OBBT-based methods on instance 48.

3.4.5 Sensitivity Analysis 2: the effect of early stop

In the process of bound tightening, OBBT-RH is utilized to refine the input bounds of each neuron. This task is computationally intensive, requiring the solution of MIPs for both the upper and lower bounds of potentially hundreds of neurons.

Early stopping is a technique introduced to reduce the computational burden. It achieves this by terminating the bound tightening process before reaching the optimal solution under certain conditions.

To prevent any single OBBT-RH instance from taking an excessively long time, we have set a 30-second time limit. This decision is informed by empirical evidence indicating that many bounds exhibit only marginal improvement after 30 seconds. Therefore, this time limit is selected to balance the achievement of high-quality bounds with the need for reasonable computing time.

Figure 3.5 illustrates the impact of early stopping on bound tightening time across different network layers, presented in a log-scale format. We cease further tightening of a neuron once it stabilizes, regardless of the application of early stopping. To assess the effectiveness of early stopping, we compare the number of stabilized neurons with and without this criterion. As depicted in the figure, early stopping—particularly with a 30-second time limit—does not significantly affect neuron stabilization. This finding supports the use of early stopping as an effective strategy to enhance the scalability of OBBT-RH without significantly compromising the quality of the bounds obtained. Due to the prolonged solving time required for tightening the inputs of the ReLU3 and ReLU4 layers with OBBT-RH-3 and OBBT without early stop (exceeding 4 hours), we resorted to using 24 threads to run a single OBBT instance. Therefore, the actual solving time with 2 threads would be even longer.

3.5 Conclusion

We present a new algorithm integrating bound-tightening and MIP with a rolling horizon strategy as a promising approach for verifying neural networks. Our method is particularly suited for handling networks where tighter bounds are needed but cannot be obtained from existing effective bound propagation methods. Future work will focus on extending this methodology to other types of nonlinear operators and further improving the computational efficiency of the verification process.

CHAPTER 4

COMPACT OPTIMALITY VERIFICATION FOR OPTIMIZATION PROXIES

4.1 Introduction

In recent years, there has been a surge of interest in optimization proxies, i.e., differentiable programs that approximate the input/output mappings of parametric optimization problems. Parametric optimization problems arise in many application areas, including power systems operations, supply chain management, and manufacturing. To be applicable in practice, these optimization proxies need to satisfy two key properties: they must return *feasible* solutions and they must return *high-quality* solutions. Feasibility has received significant attention in recent years. Constraint violations can be mitigated by incorporating them in the loss function [103, 104, 105, 106]. In some cases, the feasibility could be guaranteed by designing mask operations in the autoregression decoding [107, 108, 109] or designing projection procedures [110, 111, 112, 113, 114, 115, 116]. For better efficiency, the projection procedure can be approximated with unrolled first-order methods [117, 118, 119, 120, 121].

However, quality guarantees for optimization proxies have received much less attention and have remained largely empirical. To the authors' knowledge, [122] is the only work that provably provides worst-case guarantees for optimization proxies. They formalize the optimality verification problem as a bilevel optimization and reformulate it into a single level using traditional techniques from optimization theory. However, the resulting single-level formulation faces significant computational challenges as it introduces a large number of auxiliary decision variables and constraints. Moreover, it only applies to convex parametric optimization problems.

This chapter reconsiders the optimality verification problem and proposes a compact

formulation that applies to non-convex problems and is more advantageous from a computational standpoint. The formulation draws inspiration from robust optimization approaches, establishing a natural connection between neural network verification and robust decision-making under uncertainty, which is explored in depth in Section 4.5. Moreover, the chapter proposes an effective primal heuristic, based on a (parallelized) Projected Gradient Attack (PGA). When the problem is convex, PGA makes use of a conservative approximation of the value function of the optimization that leverages subgradients. To showcase the benefits of these modeling and algorithmic contributions, the chapter considers two applications: DCOPF problems and the knapsack problems. The DCOPF illustrates how the compact optimality verification formulation and PGA can be applied to realistic industrial-sized test cases, while the knapsack problem shows their applicability to a non-convex problem. The chapter also proposes novel MIP encoding of the feasibility layers of the DCOPF and knapsack optimization proxies. Experimental results are provided to highlight the benefits of the compact optimality verification formulation.

The contributions of this chapter are summarized as follows.

- The chapter proposes a *compact formulation for the optimality verification* of optimization proxies. The formulation provides computational benefits compared to the existing bilevel formulation and can be used to verify non-convex optimization problems.

- The chapter proposes a (parallelized) Projected Gradient Attack (PGA) that is effective in finding high-quality feasible solutions i.e., adversary points. The PGA features a novel conservative approximation of convex value functions.

- The chapter demonstrates how to apply the compact optimality verification to two problems: large-scale DCOPF and knapsack problems. The chapter contributes new feasibility restoration layers for the proxies and new encoding of these layers as MIP models for verification purposes.

- Extensive experiments show the compact formulation, together with the primal heuristic, can effectively verify optimization proxies for large-scale DCOPF problems and knapsack problems. The compact formulation also brings substantial computational benefits compared to the bilevel formulation.

The rest of the chapter is organized as follows. Section 4.2 surveys the relevant literature. Section 4.3 introduces background knowledge about parametric optimization, optimization proxies, and optimality verification. Section 4.4 presents the compact formulation for the optimality verification. Section 4.6 discusses the projected gradient attack with value function approximation. Section 4.7 presents the optimality verification on DCOPF proxies and reports the numerical results on realistic industrial-size instances. Section 4.8 presents the optimality verification for non-convex parametric optimization problems using the knapsack problem as a case study. Section 4.9 concludes the chapter and discusses future research directions.

4.2 Related Work

Neural networks have been increasingly used in safety-critical applications such as autonomous driving [123], aviation [124] and power systems [125]. However, it has become apparent that they may be highly sensitive to adversarial examples [126] i.e., a small input perturbation could cause a significant and undesired change in the output. To characterize the effects of such perturbations, recent verification algorithms typically employ one of two strategies: either they perform an exhaustive search of the input domain to identify a worst-case scenario [127, 122], or they adopt a less computationally intensive method to approximate an upper bound for the worst-case violation [128, 92, 129].

While optimality verification falls in the first category, it departs from the large body of the literature in two ways. First, evaluating the objective of the optimality verification problem involves the solving of an optimization problem, which brings unique modeling and computational challenges. Second, motivated by practical application in power sys-

tems, this chapter considers larger and more complex input perturbations than the small ℓ_p-norm perturbation often considered in existing work, as outlined in [130].

It is also important to mention the work on dual optimization proxies [131]. Dual optimization proxies provide dual feasible solutions at inference time for a particular instance. In contrast, optimality verification provides a worst-case guarantee for a distribution of instances. They complement each other naturally. [131] show how to derive dual optimization proxies for the second-order cone relaxation of ACOPF.

4.3 Preliminaries

4.3.1 Parametric Optimization

Consider a parametric optimization problem of the form

$$P(\boldsymbol{x}): \quad \min_{\boldsymbol{y}} \quad c(\boldsymbol{x}, \boldsymbol{y}) \tag{4.1a}$$

$$\text{s.t.} \quad g(\boldsymbol{x}, \boldsymbol{y}) \leq 0, \tag{4.1b}$$

where $\boldsymbol{x} \in \mathbb{R}^p$ is the input *parameter*, $\boldsymbol{y} \in \mathbb{R}^n$ is the decision variable, $c : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}$ is the cost function and $g : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}^m$ represents the constraints. The *feasible set* is denoted by $\mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathbb{R}^n \mid g(\boldsymbol{x}, \boldsymbol{y}) \leq 0\}$. The *optimal value* of $P(\boldsymbol{x})$ is denoted by $\Phi(\boldsymbol{x})$ and Φ is referred to as the *value function*. The set of optimal solutions is denoted by $\mathcal{Y}^*(\boldsymbol{x}) \subseteq \mathcal{Y}(\boldsymbol{x})$. The parametric optimization problem can be viewed as a mapping from the input parameter $\boldsymbol{x} \in \mathcal{X}$ to an optimal decision $\boldsymbol{y}^* \in \mathcal{Y}^*(\boldsymbol{x})$.

For instance, in the Optimal Power Flow (OPF) problem, the parameters are the electricity demand and generation costs. The optimization consists in finding the most cost-effective power generation that satisfies the physical and engineering constraints. In the Traveling Salesman Problem (TSP), the parameters are the locations and travel costs and the optimization consists in finding the shortest possible route that visits each location exactly once and returns to the starting point.

In a minimization problem, a feasible solution provides a primal bound, which is an

upper bound on the optimal value. A dual bound is obtained from a dual-feasible solution and/or via branch-and-bound, and establishes a lower bound on the optimal value. Dual bounds are central to proving global optimality. Note that, when maximizing, a primal (resp dual) bound is a lower (resp. upper) bound on the optimal value.

4.3.2 Optimization Proxies

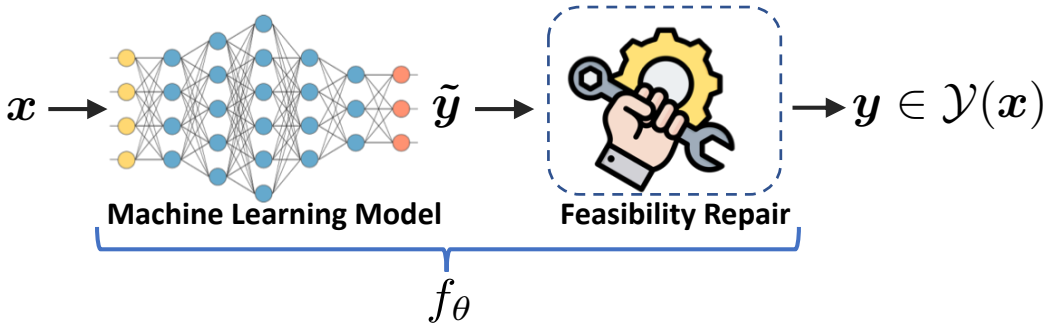


Figure 4.1: Optimization Proxies

Optimization proxies (see Figure 4.1) are differentiable programs that approximate the mapping from input parameters $\boldsymbol{x}$ to optimal decisions $\boldsymbol{y}^*$ of the parametric optimization problem. Because outputs of machine learning typically cannot satisfy complex constraints, optimization proxies consist of two parts, a machine learning model predicting the optimal decision and a feasibility repair step that projects the prediction into the feasible space. Trained optimization proxies are denoted by f_θ, with θ denoting the weights.

There has been significant progress in ensuring that optimization proxies produce feasible solutions e.g., [115, 118, 116]. Therefore, this chapter assumes that optimization proxies provide feasible solutions, i.e.,

$$\forall \boldsymbol{x} \in \mathcal{X},\ f_\theta(\boldsymbol{x}) \in \mathcal{Y}(\boldsymbol{x}). \tag{4.2}$$

Depending on the parametrization of the machine learning models and the modeling of the feasibility repair steps, optimization proxies could be trained using supervised learning [110, 132, 133, 125], reinforcement learning [107, 108, 109] and unsupervised learning [134, 135, 118, 136, 116].

4.3.3 Optimality Verification

Let $\mathcal{X} \subseteq \mathbb{R}^p$ denote the set of possible inputs and f_θ be an optimization proxy. The *Optimality Verification Problem* can be formalized as

$$(P_o): \quad \max_{\boldsymbol{x} \in \mathcal{X}} \quad c(\boldsymbol{x}, f_\theta(\boldsymbol{x})) - \Phi(\boldsymbol{x}) \tag{4.3}$$

Since the proxy is feasible, i.e., $f_\theta(\boldsymbol{x}) \in \mathcal{Y}(\boldsymbol{x})$, it follows that $c(\boldsymbol{x}, f_\theta(\boldsymbol{x})) \geq \Phi(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{X}$. In general, there is no analytical formula for Φ, which is typically not differentiable or even continuous. Hence P_o cannot be solved directly.

[122] introduced a bilevel formulation for the optimality verification problem

$$\max_{\boldsymbol{x} \in \mathcal{X}} \quad c(\boldsymbol{x}, f_\theta(\boldsymbol{x})) - c(\boldsymbol{x}, \boldsymbol{y}^*) \tag{4.4a}$$

$$\text{s.t.} \quad \boldsymbol{y}^* \in \operatorname*{arg\,min}_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \quad c(\boldsymbol{x}, \boldsymbol{y}). \tag{4.4b}$$

In (4.4), the leader (upper level) chooses input $\boldsymbol{x} \in \mathcal{X}$ and computes the neural network output $f_\theta(\boldsymbol{x})$ via a MIP encoding. The follower (lower level) then computes an optimal solution $\boldsymbol{y}^*$ of the optimization problem (4.4b). When the lower level is linear or quadratic, the bilevel formulation admits a single-level reformulation using the KKT conditions and mixed-complementarity constraints. Assuming that the proxy is a ReLU-based DNN, the overall verification problem can then be cast as an mixed-integer linear programming (MILP) and solved with off-the-shelf optimization solvers like Gurobi. More generally, a single-level reformulation requires strong duality assumptions and introduces additional variables and constraints, especially non-convex complementarity constraints. It creates some computational challenges as documented in the experiments. Moreover, the reformulation is not available when problem P is non-convex, since KKT conditions are not sufficient for optimality, and may not apply if, e.g., the lower level is discrete. In general, bilevel optimization with non-convex lower-level problems is Σ_2^P hard [137]. No existing solver can solve such problems efficiently.

4.4 Compact Optimality Verification

The core contribution of this chapter is a compact formulation for the optimality verification problem, that addresses the shortcomings of the bilevel approach. Namely, the chapter proposes to formulate the optimality verification as

$$(P_c) \quad \max_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{y}} \quad c(\boldsymbol{x},\hat{\boldsymbol{y}}) - c(\boldsymbol{x},\boldsymbol{y}) \tag{4.5a}$$

$$s.t. \quad \hat{\boldsymbol{y}} = f_\theta(\boldsymbol{x}), \tag{4.5b}$$

$$\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}). \tag{4.5c}$$

The main difference between (4.5) and (4.4) is that the inner problem (4.4b) is replaced with the simpler constraint (4.5c). In fact, the proposed formulation (4.5) is the so-called high-point relaxation [138] of the bilevel formulation (4.4). Theorem 4.4.1 shows that (4.5) has the same optimum as (4.4), i.e., the high-point relaxation is exact. To the authors' knowledge, this result is novel.

Theorem 4.4.1. *Problems* (4.4) *and* (4.5) *have same optimum.*

Proof. Recall that the compact formulation (4.5) is a relaxation of (4.4). Therefore, it suffices to show that an optimal solution to (4.5), denoted by $(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})$, is feasible for (4.4).

By definition, $\tilde{\boldsymbol{y}} \in \mathcal{Y}(\tilde{\boldsymbol{x}})$, i.e., $\tilde{\boldsymbol{y}}$ is feasible for the lower-level problem (4.4b) with upper-level decision $\tilde{\boldsymbol{x}}$. Next, assume $\tilde{\boldsymbol{y}}$ is not optimal for (4.4b), i.e., there exists $\hat{\boldsymbol{y}} \in \mathcal{Y}(\tilde{\boldsymbol{x}})$ such that $c(\tilde{\boldsymbol{x}},\hat{\boldsymbol{y}}) < c(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})$. By construction, $(\tilde{\boldsymbol{x}},\hat{\boldsymbol{y}})$ is feasible for (4.5) with objective value strictly better than $(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})$, which contradicts the optimality of $(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})$. □

The proposed compact formulation (4.5) has several advantages. First, it naturally supports non-convex constraints and objectives, In contrast, the standard approach of reformulating bilevel problems into a single-level problem with complementarity constraints, is not possible when the lower-level problem (4.4b) is non-convex. *This is the first tractable exact formulation for verifying the optimality of non-convex optimization proxies.* Second, even

when P is convex and a single-level reformulation is possible, the compact formulation avoids the additional variables and constraints that come with a single-level reformulation. In particular, it eliminates the need for complementarity constraints, which are notoriously difficult to solve. This last point is demonstrated in the experiments of Section 4.7.

4.5 Connection to Robust Optimization

This section explores the intrinsic relationship between neural network verification and robust optimization (RO), with particular focus on how our compact optimality verification formulation fits within this broader context.

Neural Network Verification provides formal, mathematical guarantees about network behavior. Unlike traditional testing which samples input-output behavior, NNV proves properties for entire sets of inputs using techniques rooted in formal methods, logic, and mathematics. Key verification goals include robustness verification, which ensures that network outputs remain stable (e.g., maintaining classification) despite input perturbations within a defined neighborhood; safety verification, which guarantees that network outputs always remain within predefined safe regions for all inputs from a specified domain; and optimality verification, the focus of this chapter, which certifies bounds on the optimality gap of neural networks that approximate optimization problems.

4.5.1 Robust Optimization Foundations

Robust Optimization provides a mature mathematical framework for decision-making under uncertainty [139, 140]. Unlike stochastic optimization which relies on probabilistic descriptions, RO takes a worst-case approach, seeking solutions that remain feasible and perform well under all possible realizations within a predefined uncertainty set.

The canonical form of a robust optimization problem is:

$$\min_{\boldsymbol{y}} \max_{\boldsymbol{x} \in \mathcal{U}} c(\boldsymbol{x}, \boldsymbol{y}) \tag{4.6}$$

where $\boldsymbol{y}$ represents the decision variables, $\boldsymbol{x}$ represents uncertain parameters confined to uncertainty set $\mathcal{U}$, and $c(\boldsymbol{x}, \boldsymbol{y})$ is the objective function. This min-max formulation seeks decisions that minimize the worst-case objective value.

Several key concepts underpin the RO framework. Uncertainty sets provide mathematical representations of the range of possible parameter values, such as intervals, polyhedra, or ellipsoids. The robust counterpart offers a deterministic reformulation of an uncertain problem, ensuring feasibility under all possible data realizations in the uncertainty set. The price of robustness represents the objective value sacrifice incurred to achieve robustness compared to the nominal solution—a fundamental trade-off in robust decision-making.

4.5.2 Linking Neural Network Verification and Robust Optimization

The core conceptual link between NNV and RO lies in interpreting neural network input perturbations as data uncertainty within the RO framework. In NNV, a central question is whether a network maintains correct behavior when its input $\boldsymbol{x}$ is perturbed by some δ within a bounded set $\mathcal{S}$. The set of possible inputs is thus $\boldsymbol{x} + \mathcal{S} = \{\boldsymbol{x} + \delta | \delta \in \mathcal{S}\}$.

This directly parallels RO, where the key question is whether a solution remains feasible or optimal when parameters vary within an uncertainty set. The nominal input $\boldsymbol{x}$ corresponds to nominal data, the perturbation δ corresponds to data uncertainty, and the perturbation set $\mathcal{S}$ corresponds to the uncertainty set $\mathcal{U}$.

For verification of classification networks, the robustness property can be expressed as finding the worst-case violation over the perturbation set $\mathcal{S}$, mirroring the inner maximization problems found in RO.

4.5.3 The Compact Formulation as a Robust Framework

Our compact optimality verification formulation presented in Section 4.4 naturally aligns with robust optimization principles but with a distinctive structure. Recall that the formu-

lation is:

$$\max_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{y}} \quad c(\boldsymbol{x},\hat{\boldsymbol{y}}) - c(\boldsymbol{x},\boldsymbol{y}) \tag{4.7a}$$

$$\text{s.t.} \quad \hat{\boldsymbol{y}} = f_\theta(\boldsymbol{x}), \tag{4.7b}$$

$$\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}). \tag{4.7c}$$

This can be viewed as a max-max problem rather than the traditional min-max structure of robust optimization:

$$\max_{\boldsymbol{x}\in\mathcal{X}} \max_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})} \quad c(\boldsymbol{x}, f_\theta(\boldsymbol{x})) - c(\boldsymbol{x},\boldsymbol{y}) \tag{4.8}$$

The inner maximization seeks the best possible solution $\boldsymbol{y}$ for a given parameter $\boldsymbol{x}$, effectively computing $-\Phi(\boldsymbol{x})$. The outer maximization then finds the parameter $\boldsymbol{x}$ that maximizes the gap between the proxy solution and the optimal solution.

4.5.4 Robust Optimization Techniques in Neural Network Verification

Several techniques from RO have been adapted for NNV problems. Convex relaxations (e.g. LP, SOCP, or SDP) replace the non-convex verification problem with a related, computationally tractable convex optimization problem. The Projected Gradient Attack-approach proposed in Section 4.6 draws inspiration from adversarial attack methods and first-order optimization techniques common in RO, demonstrating the practical application of these concepts.

4.6 Projected Gradient Attack

Algorithm 7 Projected Gradient Attack with Value Function Approximation (PGA-VFA)

Input: Input space $\mathcal{X}$, initial point $\boldsymbol{x}_0$, value function approximation $\widehat{\Phi}(\cdot)$, number of iterations T

for $t = 0$ **to** T **do**

$\hat{\boldsymbol{x}}_{t+1} = \boldsymbol{x}_t + \lambda \cdot \nabla_{\boldsymbol{x}} \left[c(\boldsymbol{x}_t, f_\theta(\boldsymbol{x}_t)) - \widehat{\Phi}(\boldsymbol{x}_t) \right]$

$\boldsymbol{x}_{t+1} = \text{Proj}_{\mathcal{X}}(\hat{\boldsymbol{x}}_t)$

end for

The projected gradient attack is highly effective in finding high-quality feasible solutions, i.e., adversarial examples, for verification problems. It can be formalized as

$$\hat{\boldsymbol{x}}_{t+1} = \boldsymbol{x}_t + \lambda \nabla_{\boldsymbol{x}} \left[c(\boldsymbol{x}_t, f_\theta(\boldsymbol{x}_t)) - \Phi(\boldsymbol{x}_t) \right], \tag{4.9a}$$

$$\boldsymbol{x}_{t+1} = \text{Proj}_{\mathcal{X}}(\hat{\boldsymbol{x}}_{t+1}). \tag{4.9b}$$

When $\mathcal{X}$ is an ℓ_p ball, the projection step in (4.9b) can be computed in close form.

A challenge in implementing the projected gradient attack is the computation of the gradient $\nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x})$. One possible approach is to use the implicit function theorem on the KKT conditions [111, 112]. However, this gradient computation involves the solving of many optimization instances, which may be computationally intensive for large-scale problems as those studied in this chapter. *A second key contribution of the chapter is the use in the projected gradient attack of a piece-wise linear conservative approximation of convex value function.* It builds on the following well-known result.

Theorem 4.6.1. *[141] Consider a convex parametric optimization P where (i) the objective c and left-hand side g do not depend on $\boldsymbol{x}$ and (ii) the constraints' right-hand side is an affine function of $\boldsymbol{x}$. Then, the value function Φ of P is convex.*

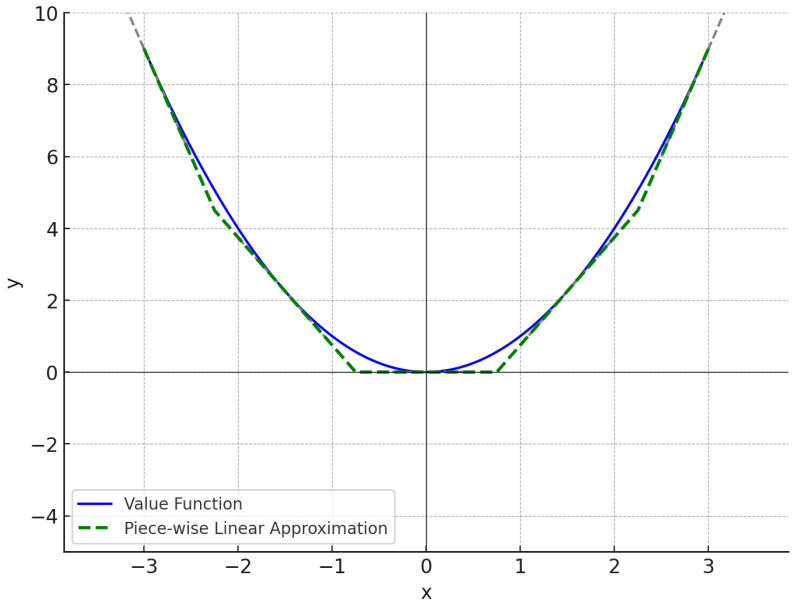


Figure 4.2: Value Function and its Piece-wise Linear Approximation

Theorem 4.6.1 implies that a convex value function can be outer-approximated by a closed-formed piecewise linear function:

$$\widehat{\Phi}(\boldsymbol{x}) = \max_{i \in [N]} \Phi(\boldsymbol{x}_i) + \nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x}_i)^\intercal (\boldsymbol{x} - \boldsymbol{x}_i), \tag{4.10}$$

where $\{\boldsymbol{x}_i\}_{i=1}^N$ denotes a set of parameters of historical parametric optimization instances. Moreover, the optimal dual $\boldsymbol{\lambda}_i$ of an instance with parameter $\boldsymbol{x}_i$ represents a subgradient of the value function with respect to $\boldsymbol{x}_i$. If the optimal dual is unique, then $\nabla_{\boldsymbol{x}} \Phi(\boldsymbol{x}_i) = \boldsymbol{\lambda}_i$ [142]. Therefore, the approximate value function reads:

$$\widehat{\Phi}(\boldsymbol{x}) = \max_{i \in [N]} \Phi(\boldsymbol{x}_i) + \boldsymbol{\lambda}_i^\intercal (\boldsymbol{x} - \boldsymbol{x}_i), \tag{4.11}$$

The piece-wise linear approximation on the convex value function is illustrated in Figure 4.2. Other models such as Input Convex Neural Networks [143] could also be used for the value function approximation.

The projected gradient attack may converge to a low-quality solution due to becoming trapped in local optima. This work proposed several techniques aimed at enhancing the search process. One acceleration technique consists in sampling a set of starting points

in input domain $\{\boldsymbol{x}_0^m\}_{m=1}^M$ and run PGA-VFA($\mathcal{X}$, $\boldsymbol{x}_0^m$) for every starting point. Another acceleration technique to improve the search is to partition the input domain into subregions, and then run Algorithm (7) on different subregions $\{\mathcal{X}_p\}_{p=1}^P$ in parallel i.e., running PGA-VFA($\mathcal{X}_p$, $\boldsymbol{x}_0$) for every subregion.

4.7 DC Optimal Power Flow

4.7.1 DCOPF Formulation

DCOPF is a fundamental problem for modern power system operations. It aims at determining the least-cost generator setpoints that meet grid demands while satisfying physical and operational constraints. With the penetration of renewable energy and distributed energy resources, the system operators must continuously monitor risk in real-time, i.e., they must quickly assess the system's behavior under various changes in load and renewables by solving a large volume of DCOPF problems. However, traditional optimization solvers may not be capable of solving them quickly enough for large-scale power networks [144]. Recent advancements in learning-based methods have accelerated the process of finding feasible and empirically near-optimal solutions considerably faster than conventional approaches [116, 145, 115]. *This chapter aims at providing formal quality guarantees for optimization proxies in this space to complement existing primal learning methods.*

Consider the DCOPF formulation

$$\min_{\mathbf{pg},\boldsymbol{\xi}^{\text{th}}} \quad \boldsymbol{c}^\top \mathbf{pg} + M^{\text{th}} \boldsymbol{e}^\top \boldsymbol{\xi}^{\text{th}} \tag{4.12a}$$

$$\text{s.t.} \quad \boldsymbol{e}^\top \mathbf{pg} = \boldsymbol{e}^\top \text{pd}, \tag{4.12b}$$

$$+ \boldsymbol{H}\mathbf{pg} + \boldsymbol{\xi}^{\text{th}} \geq -\bar{\text{pf}} + \boldsymbol{H}\text{pd}, \tag{4.12c}$$

$$- \boldsymbol{H}\mathbf{pg} + \boldsymbol{\xi}^{\text{th}} \geq -\bar{\text{pf}} - \boldsymbol{H}\text{pd}, \tag{4.12d}$$

$$\underline{\text{pg}} \leq \mathbf{pg} \leq \bar{\text{pg}}, \tag{4.12e}$$

$$\boldsymbol{p} \in \mathbb{R}^B, \boldsymbol{\xi}^{\text{th}} \in \mathbb{R}_+^E \tag{4.12f}$$

where B, E denote the total number of buses and transmission lines in the power grid, respectively, $\boldsymbol{d} \in \mathbb{R}^B$ denotes the electricity load, $\mathbf{pg}$ denotes the decision variables capturing energy dispatches, $\boldsymbol{\xi}^{\text{th}}$ denotes the thermal limit violations, $\bar{\text{pf}}$ corresponds to the flow limits on the transmission lines and $\boldsymbol{H} \in \mathbb{R}^{E \times B}$ denotes the power transfer distribution factors. The price of electricity generation is represented by $\boldsymbol{c}$, and M^{th} is the price of violating thermal constraints. The vector $\boldsymbol{e}$ consists of all ones. Constraint (4.12b) ensures the global power balance i.e., the total load equals the total supply. Constraint (4.12c) and (4.12d) measure the thermal violations. Constraint (4.12e) ensures that the outputs of the generators remain within their physical limits.

4.7.2 Compact Optimality Verification

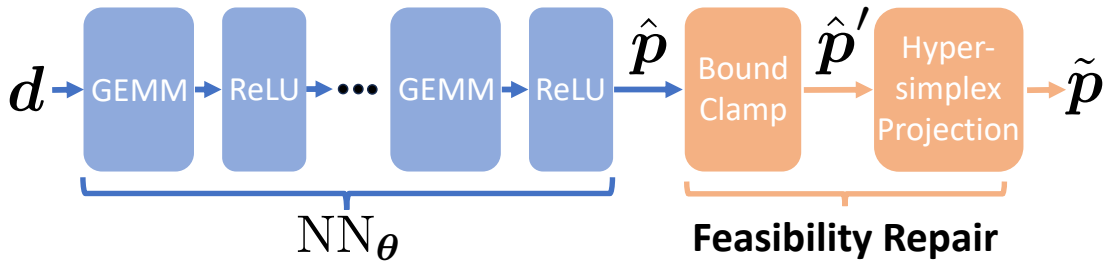


Figure 4.3: Optimization Proxies for DCOPF

This work considers the optimization proxy proposed in [116] and illustrated in Figure 4.3. It consists of a fully-connected neural network with ReLU activation to predict the optimal dispatches and feasibility layers to ensure that the outputs satisfy the hard constraints. The feasibility layer consists of two parts, a bound-clamping layer, which employs a hard-sigmoid function to ensure that dispatch decisions remain within the generators' physical upper and lower generation limits (4.12e) and a hypersimplex layer, which uses a differentiable binary search to guarantee that total power generation matches the total electricity demand, addressing constraint (4.12b). Constraints (4.12c) and (4.12d) are soft and are penalized in the loss function when training the ML models. *Another contribution of this chapter is a compact encoding of these layers into a MIP formulation.* The detailed mod-

eling of the compact formulation is deferred in Appendix B.1.

4.7.3 Empirical Evaluation

Experiment Setup

The optimality verification for DCOPF proxies is evaluated over IEEE 57-/118-/300-bus and Pegase 1354-bus test cases from the PGLib library [146]. The data generation follows [116]. Denote by $\boldsymbol{d}^{\text{ref}}$ the nodal load vector from the reference PGLib case. The instances are generated by perturbing the reference load vector. Namely, for instance i, $\boldsymbol{d}^{(i)} = (\gamma^{(i)} + \boldsymbol{\eta^{(i)}}) \times \boldsymbol{d}^{\text{ref}}$, where $\gamma^{(i)} \in \mathbb{R}$ is a global scaling factor and $\boldsymbol{\eta^{(i)}} \in \mathbb{R}^B$ denotes element-wise noises. The γ is sampled from uniform distribution $U[80\%, 120\%]$ and for each load, η is sampled from a uniform distribution $U[-5\%, 5\%]$. This distribution captures system-wide correlations (γ), while allowing for local variability ($\boldsymbol{\eta}$). The optimization proxies are trained using the self-supervised learning algorithm in [116]. It is important to note that the verification problem only depends on the weights of the trained proxy.

The parameter input domain $\mathcal{X}$ reflects the support of the distribution of instances described above. Namely,

$$\mathcal{X} = \{(\alpha + \boldsymbol{\beta}) \cdot \boldsymbol{d}^{\text{ref}} | -u \leq \alpha - 1 \leq u, -\mathbf{5}\% \leq \boldsymbol{\beta} \leq \mathbf{5}\%\},$$

where $\alpha \in \mathbb{R}$, $\boldsymbol{\beta} \in \mathbb{R}^B$ capture the distribution of $\gamma, \boldsymbol{\eta}$, and $u \in \{0, 1\%, 2\%, 5\%, 10\%, 20\%\}$ controls the size of the input domain. Each value of u yields a different optimality verification instance; note that larger values of u make the instances harder to verify.

The value function approximation $\hat{\Phi}$, used in PGA-VFA (Algorithm 7), is constructed using primal and dual solutions of 50,000 instances, generated using the above distribution. PGA-VFA is executed in parallel, across 200 threads, using the acceleration techniques of Section 4.6. The initial step size is 10^{-3}, and is reduced by a factor 10 if no improvement is recorded over 10 iterations. Finally, PGA-VFA is stopped if no improved solution is found after 20 consecutive iterations, or a maximum of 500 iterations is reached.

All verification problems are solved with Gurobi 10.0 [20] using 16 threads and a 6-hour time limit. Preliminary experiments revealed that Gurobi struggles to find primal-feasible solutions. Therefore, unless specified otherwise, the $\boldsymbol{d}^{\text{ref}} \in \mathcal{X}$ is always passed as a warm-start to the solver. In addition, this work uses optimization-based bound tightening [98] to improve the MILP relaxation, by tightening the input domain of ReLU neurons. Finally, each verification instance is solved using 5 different seeds, and results are averaged using the shifted geometric mean

$$\mu_s(x_1, \cdots, x_n) = \sqrt[n]{\Pi_i(x_i + s)} - s.$$

The chapter uses a shift s of 1% for optimality gaps and 1 for other values. Solving times are reported in wall clock. Experiments are conducted on dual Intel Xeon 6226@2.7GHz machines running Linux on the cluster.

Numerical Results

Effectiveness of Compact Formulation Table 4.1 reports the size (number of variables and constraints) of the compact formulation and bilevel formulation, after being presolved by Gurobi. Table 4.1 shows that the compact formulation results in substantially fewer binary decision variables than its bilevel counterpart, especially on large systems.

Table 4.1: Comparison of presolved model size for Bilevel and Compact (proposed) formulations. Statistics are averages across 30 instances (6 distinct values of u and 5 unique seeds).

	#ConVars		#BinVars		#Constraints	
System	Bilevel	Compact	Bilevel	Compact	Bilevel	Compact
57	**255**	259	69	**64**	350	**342**
118	609	**522**	218	**100**	862	**619**
300	1798	**1320**	861	**325**	2982	**1821**
1354	6739	**4655**	7281	**1353**	15373	**6777**

The size reduction of the compact formulation is reflected in the solving process. For ease of comparison, the verification instances are split into two groups. On the one hand, *closed* instances are instances that can be solved by at least one approach within the prescribed time limit. They include all 57-bus and 118-bus instances, as well as the 300-bus instances with $u = 0, 1\%, 2\%, 5\%$. On the other hand, *open* instances are those that cannot be solved by either formulation. They include the 300-bus instances with $u = 10\%, 20\%$, and all the 1354-bus instances.

Table 4.2 reports the solving times of the compact formulation and bilevel formulation on solved instances, and indicate the relative speedup of the compact formulation, defined as $\frac{t_{\text{Bilevel}} - t_{\text{Compact}}}{t_{\text{Compact}}} \times 100\%$. The compact formulation consistently outperforms the bilevel formulation, except for two small instances which are solved by both formulations in under 2 seconds. Notably, the compact formulation achieves higher speedups on larger systems, with up to 209.5% speedup on the 300-bus system with 2% input perturbation.

Table 4.2: Comparison of solution times(s) for Bilevel and Compact (proposed) formulations across closed (i.e., solved) instances.

System	$\%u$	Bilevel	Compact	Speedup*
57	0	0.4	**0.3**	22.9%
	1	**0.6**	0.6	-3.4%
	2	0.6	**0.5**	10.4%
	5	1.9	**1.1**	75.8%
	10	**1.2**	1.2	-1.7%
	20	0.7	**0.5**	27.6%
118	0	1.7	**1.2**	43.3%
	1	3.9	**2.1**	89.7%
	2	4.7	**3.1**	53.6%
	5	28.4	**12.1**	135.2%
	10	18.7	**14.1**	32.4%
	20	76.8	**59.4**	29.3%
300	0	119.8	**63.4**	88.9%
	1	298.6	**166.0**	79.9%
	2	1302.7	**420.8**	209.5%
	5	17585.4	**10507.9**	67.4%

*Average speedup compared to bilevel formulation.

Next, Table 4.3 reports the performance of two formulations on open instances, where none of the formulations can solve the instances to optimality within 6 hours, primarily due to the weak linear relaxation. In this case, the comparison focuses on the final primal and dual objectives. The gains in percentage under the Primal and Dual columns report the performance improvements on the primal and dual side, respectively: $\frac{\text{Primal}_{\text{Compact}} - \text{Primal}_{\text{Bilevel}}}{\text{Primal}_{\text{Bilevel}}} \times 100\%$ and $\frac{\text{Dual}_{\text{Bilevel}} - \text{Dual}_{\text{Compact}}}{\text{Dual}_{\text{Bilevel}}} \times 100\%$. As reported in Table 4.3, the compact formulation finds substantially better primal solutions than the bilevel formulation, especially for harder

instances. This is further supported in Figure 4.4, where the compact formulation consistently converges faster. On the dual side, the compact formulation is slightly better than its bilevel counterpart.

Table 4.3: Primal and dual objective values attained by the Bilevel and Compact formulation on open (i.e., unsolved) instances.

		Primal bound (K$)			Dual bound (M$)		
System	$\%u$	Bilevel	Compact	%Gain†	Bilevel	Compact	%Gain†
300	10	466.76	596.98	27.90	4.45	3.82	14.08
	20	410.19	582.81	42.08	10.79	9.29	13.88
1,354	0	94.22	360.03	282.13	29.64	26.00	12.25
	1	84.44	365.50	332.83	36.35	35.39	2.63
	2	50.12	369.68	637.65	47.64	45.13	5.26
	5	54.32	371.52	584.02	78.79	74.31	5.69
	10	46.63	353.20	657.42	135.32	129.99	3.94
	20	49.52	359.72	626.40	249.02	235.79	5.31

†Relative improvement compared to bilevel formulation, in %.

Since the compact formulation systematically outperforms the bilevel formulation, the next experiments focus on the compact formulation.

Effectiveness of PGA-VFA Table 4.4 focuses on the performance of the proposed PGA-VFA and compact formulation in identifying feasible solutions on large power systems (300 bus and 1354 bus).

Table 4.4: Comparisons of PGA-VFA and the compact formulation on final objective and solving times for large power systems.

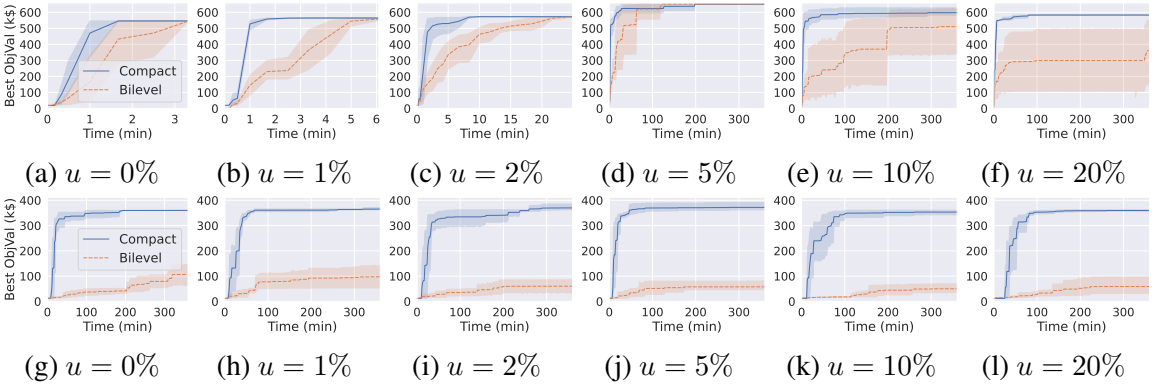
		PGA-VFA		Compact		
System	$\%u$	Obj. ($)	Time (s)	Obj. ($)	Time (s)	†T2PGA (s)
300	0	544923.75	4.53	546038.13	63.41	52.06
	1	562348.33	7.09	564263.39	166.01	114.79
	2	572166.01	37.30	572464.17	420.83	307.40
	5	572166.01	49.26	651163.59	10507.88	872.25
	10	572166.01	41.41	596983.07	t.o.	2646.21
	20	572166.01	17.86	582808.23	t.o.	2646.28
1,354	0	380673.60	67.26	360031.97	t.o.	t.o.
	1	399441.04	63.54	365497.51	t.o.	t.o.
	2	403725.74	41.98	369678.23	t.o.	t.o.
	5	403725.74	31.88	371524.52	t.o.	t.o.
	10	571252.51	85.88	353196.16	t.o.	t.o.
	20	571252.51	76.43	359722.30	t.o.	t.o.

†Time taken by the compact formulation to reach the same primal objective as PGA; "t.o." indicate timeouts.

The performance on small systems is deferred to Appendix B.3 since the compact formulation solves those instances within 60 seconds. PGA-VFA is particularly good at finding high-quality primal solutions within short computing time. It finds near-optimal solutions on the 300-system orders of magnitude faster than the compact formulation with a reference load vector as the warm-start. For the 300-bus system with 1% perturbation, PGA-VFA finds a feasible solution with objective 562348.33 in 7.09s. Meanwhile, Gurobi solves the compact MILP formulation in 166.01s, reporting an optimal value of 564263.39. It takes Gurobi 114.79s to find a feasible solution at least as good as the one found by PGA-VFA (in 7.09s). The compact model cannot find the same quality of primal solutions

as PGA-VFA on the 1354-systems within 6 hours.

Finally, Table 4.5 reports the impact of warm-starting the compact formulation with PGA-VFA. The compact model with the PGA-VFA solution as a warm start always outperforms its counterparts with the reference load as the warm start. Observe also that the optimization barely improves on the PGA-VFA solution.



(a) $u = 0\%$ (b) $u = 1\%$ (c) $u = 2\%$ (d) $u = 5\%$ (e) $u = 10\%$ (f) $u = 20\%$

(g) $u = 0\%$ (h) $u = 1\%$ (i) $u = 2\%$ (j) $u = 5\%$ (k) $u = 10\%$ (l) $u = 20\%$

Figure 4.4: Evolution of primal objective value for Compact and Bilevel formulations over time on the 300 IEEE system (top row) and the 1354 Pegase system (bottom row) with different sizes of input domains. Higher values indicate better outcomes.

Table 4.5: Comparisons of PGA-VFA and the compact formulation for different warm starts on final objective and solving times for large power systems

System	$\%u$	PGA-VFA	Compact+Nom	Compact+PGA
300	0	544923.75	546038.13	546038.13
	1	562348.32	564263.39	564263.44
	2	572166.01	572464.17	572464.17
	5	572166.01	651163.59	651163.38
	10	572166.01	596983.07	596983.90
	20	572166.01	582808.23	582808.47
1,354	0	380673.60	360031.97	380757.88
	1	399441.04	365497.51	399579.63
	2	403725.74	369678.23	406165.38
	5	403725.74	371524.52	406164.74
	10	571252.51	353196.16	571715.99
	20	571252.51	359722.30	578732.15

4.8 Optimality Verification for Knapsack

One major benefit of the proposed compact formulation is that it may provide quality guarantees for optimization proxies that approximate non-convex parametric optimization problems. This section illustrates this capability on the knapsack problem.

4.8.1 Knapsack Formulation

Consider a knapsack problem with K items

$$\max_{\boldsymbol{y}} \quad \left\{ \boldsymbol{v}^T \boldsymbol{y} \;\middle|\; \boldsymbol{w}^T \boldsymbol{y} \leq l, \boldsymbol{y} \in \{0,1\}^K \right\}, \tag{4.13}$$

where $\boldsymbol{v}$ denotes the value of the items, l is the knapsack capacity and $\boldsymbol{w}$ denotes the weight of the items. The binary decision variable $\boldsymbol{y}_k = 1$ indicates putting the item k in the knapsack and vice versa.

4.8.2 Optimality Verification

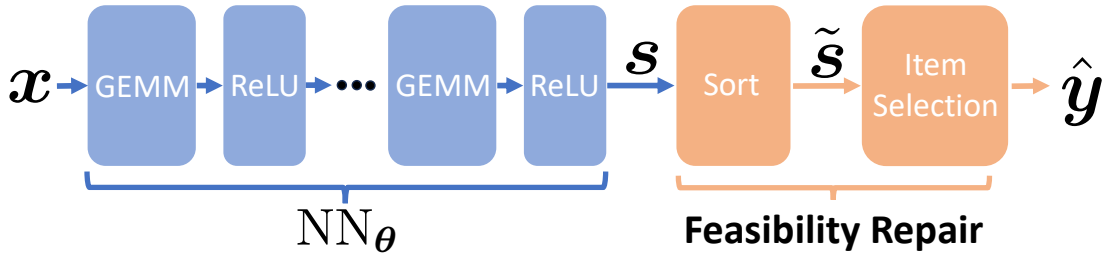


Figure 4.5: An Optimization Proxy for the Knapsack Problem.

This chapter considers the optimality verification of an optimization proxy for the knapsack problem. The proxy is summarized in Figure 4.5. First, a fully-connected neural network with ReLU activation predicts a score for each item, where a higher score indicating higher desirability of the item. Then, a novel feasibility repair step sorts the items by the predicted score in a descending order and adds items to the knapsack following the order until reaching the knapsack's capacity limit, thus enforcing $\boldsymbol{w}^T\boldsymbol{y} \leq l$. *Another contribution of this chapter is a compact formulation of this repair layer that is presented in Appendix C.1.* Because the parametric optimization is non-convex, the bilevel formulation in (4.4) cannot be reformulated into a single level.

4.8.3 Numerical Evaluation

The data for the experiments is generated by perturbing the total capacity of the knapsack and the item values, where the scaler l of the total capacity and the item values $\boldsymbol{v}$ are sampled from the uniform distribution $U[80\%, 120\%]$. More information of the knapsack proxies is detailed in the Appendix C. The input domain is defined as:

$$\mathcal{X} = \{\alpha l, \boldsymbol{\beta v} \mid -u \leq \alpha - 1 \leq u, -\mathbf{u} \leq \boldsymbol{\beta} - 1 \leq \mathbf{u}\},$$

where u controls the size of the input domain.

Table 4.6: Optimality Verification for Knapsack

#items	%u	Gap (%)	Time (s)
10	1	0.0	0.4
	5	0.0	0.4
	10	0.0	0.4
	20	0.0	0.7
50	1	0.0	89.1
	5	0.0	117.3
	10	0.0	642.8
	20	142.0	t.o.
80	1	0.0	1928.3
	5	0.0	5997.8
	10	230.0	t.o.
	20	1350.0	t.o.

* Solved with 16 threads and time limits of 6 hours.

Table 4.6 demonstrates the compact formulation can verify optimization proxies for Knapsack. *It highlights a key benefit of the compact formulation: its ability to verify non-convex optimization problems.*

4.9 Conclusion

The chapter presents a novel compact formulation for optimality verification of optimization proxies. It offers substantial computational benefits over the traditional bilevel formulations and can verify non-convex optimization problems. The chapter also introduces a Projected Gradient Attack with a value function approximation as a primal heuristic for effectively finding high-quality primal solutions. The methodology is applied to large-scale DCOPF and knapsack problems, incorporating new MILP encodings for the feasibility lay-

ers. Extensive experiments demonstrate the efficacy of the methodology in verifying proxies for DCOPF and knapsack problems, highlighting its computational advantages. Future works will investigate the scalability of the methodology on large industrial instances with the coupling with spatial branch and bound and α, β-CROWN, and extend the verification on auto-regression-based optimization proxies.

Limitations The proposed compact formulation is more general than prior state-of-the-art, and has fewer variables and constraints. Nevertheless, this exact verification scheme, which eventually relies on solving MIP problems, shares common limitations with existing exact verification methods, as highlighted in [130, 95, 147]. State-of-the-art exact verification solvers primarily focus on ReLU networks, and do not always support arbitrary linear/discrete/nonlinear constraints which are needed for optimality verification. Thus, MIP solvers are the only existing tools capable of solving optimality verification problems. MIP solvers are known to struggle when solving large verification instances, especially for finding high-quality solutions (which correspond to adversarial examples). Therefore, scalability issues still exist when solving large-scale verification problems, especially with deep neural networks and large input space. Exact optimality verification on large-scale industrial instances, and exploring its effectiveness for nonlinear activations and non-perceptron architectures, e.g.,Transformers, are promising research directions.

Appendices

APPENDIX A

FULL FORMULATION OF SLAD

A.1 Indices and Sets

$i \in \mathcal{N}$ Buses

$g \in \mathcal{G}$ Generators

$\mathcal{G}_i$ Generators at bus i

$e \in \mathcal{E}$ Branches

$t \in \mathcal{T}$ Time steps

$s \in \mathcal{S}$ Scenarios

A.2 Variables

$\mathbf{p}_{i,t,s}$ Net active power injection at bus i at time t in scenario s (MW)

$\mathbf{pf}_{e,t,s}$ Power flow on branch e at time t in scenario s (MW)

$\mathbf{pg}_{g,t,s}$ Active power dispatch of generator g at time t in scenario s (MW)

$\mathbf{pg}^{k}_{g,t,s}$ Active power dispatch for piecewise segment k of generator g at time t in scenario s (MW)

$\mathbf{r}^{\text{reg}}_{g,t,s}$ Regulating reserve dispatch of generator g at time t in scenario s (MW)

$\mathbf{r}^{\text{spin}}_{g,t,s}$ Spinning reserve dispatch of generator g at time t in scenario s (MW)

$\mathbf{r}^{\text{s,on}}_{g,t,s}$ Supplemental reserve dispatch of generator g at time t in scenario s (MW)

$\mathbf{r}^{\text{s,off}}_{g,t,s}$ Offline supplemental reserve dispatch of generator g at time t in scenario s (MW)

$\mathbf{rc}^{\uparrow}_{g,t,s}$ Ramping-up capability of generator g at time t in scenario s (MW)

$\mathbf{rc}^{\downarrow}_{g,t,s}$ Ramping-down capability of generator g at time t in scenario s (MW)

$\mathbf{rc}^{\text{str}}_{g,t,s}$ Short-term ramping capability of generator g at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{p}^{+}_{t,s}$ Global power balance surplus at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{p}^{-}_{t,s}$ Global power balance shortage at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{r}^{\text{reg}}_{t,s}$ Regulating reserve shortage at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{r}^{\text{rspin}}_{t,s}$ Regulating-plus-spinning reserve shortage at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{r}^{\text{op}}_{t,s}$ Operating reserve shortage at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{rc}^{\uparrow}_{t,s}$ Ramping-up capability shortage at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{rc}^{\downarrow}_{t,s}$ Ramping-down capability shortage at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{rc}^{\text{str}}_{t,s}$ Short-term ramping capability shortage at time t in scenario s (MW)

$\boldsymbol{\delta}\mathbf{f}_{e,t,s}$ Thermal limit violation on branch e at time t in scenario s (MW)

A.3 Parameters

Δ_t Duration of a time period (min)

p_s The probability of scenario s

$\text{pd}_{i,t,s}$ Active power demand at bus i at time t in scenario s (MW)

$\sigma_{e,i}$ PTDF coefficient for branch e and bus i

$\underline{\text{pf}}_{e,t}$ Lower thermal limit on branch e at time t (MW)

$\overline{\text{pf}}_{e,t}$ Upper thermal limit on branch e at time t (MW)

$\text{CF}_{g,t}$ Commitment flag of generator g at time t (binary)

$\mathrm{RF}_{g,t}$ Regulation flag of generator g at time t (binary)

$\mathrm{IF}_{g,t}$ Intermittent flag of generator g at time t (binary)

$\mathrm{RA}^{\mathrm{reg}}_{g,t}$ Whether generator g can provide regulating reserves at time t (binary)

$\mathrm{RA}^{\mathrm{spin}}_{g,t}$ Whether generator g can provide spinning reserves at time t (binary)

$\mathrm{RA}^{\mathrm{s,on}}_{g,t}$ Whether generator g can provide online supplemental reserves at time t (binary)

$\mathrm{RA}^{\mathrm{s,off}}_{g,t}$ Whether generator g can provide offline supplemental reserves at time t (binary)

$\mathrm{pg}_{g,0}$ Initial output of generator g (MW)

$\underline{\mathrm{pg}}_{g,t,s}$ Minimum output of generator g at time t in scenario s (MW)

$\bar{\mathrm{pg}}_{g,t,s}$ Maximum output of generator g at time t in scenario s (MW)

$\bar{\mathrm{pg}}^{k}_{g,t}$ Width of k^{th} bid step for generator g at time t (MW)

$\mathrm{r}^{\downarrow}_{g,t}$ Ramping-down rate of generator g at time t (MW/min)

$\mathrm{r}^{\uparrow}_{g,t}$ Ramping-up rate of generator g at time t (MW/min)

$\bar{\mathrm{r}}^{\mathrm{reg}}_{g,t}$ Maximum regulating reserve dispatch of generator g at time t (MW)

$\bar{\mathrm{r}}^{\mathrm{spin}}_{g,t}$ Maximum spinning reserve dispatch of generator g at time t (MW)

$\bar{\mathrm{r}}^{\mathrm{s,on}}_{g,t}$ Maximum online supplemental reserve dispatch of generator g at time t (MW)

$\bar{\mathrm{r}}^{\mathrm{s,off}}_{g,t}$ Maximum offline supplemental reserve dispatch of generator g at time t (MW)

$\mathrm{c}^{\mathrm{NL}}_{g,t}$ No-load cost of generator g at time t ($)

$C^{k}_{g,t}$ Energy dispatch cost at bid step k for generator g at time t ($/MW)

$\mathrm{c}^{\mathrm{reg}}_{g,t}$ Regulating reserve cost for generator g at time t ($/MW)

$\mathrm{c}^{\mathrm{spin}}_{g,t}$ Spinning reserve cost for generator g at time t ($/MW)

$\mathrm{c}^{\mathrm{s,on}}_{g,t}$ Supplemental reserve cost for generator g at time t ($/MW)

$\mathrm{c}^{\mathrm{s,off}}_{g,t}$ Offline supplemental reserve cost for generator g at time t (\$/MW)

$\mathrm{rr}^{\mathrm{reg}}_{t}$ Regulating reserve requirement at time t (MW)

$\mathrm{rr}^{\mathrm{rspin}}_{t}$ Regulating-plus-spinning reserve requirement at time t (MW)

$\mathrm{rr}^{\mathrm{op}}_{t}$ Operating reserve requirement at time t (MW)

$\mathrm{rr}^{\mathbf{rc}^{\uparrow}}_{t}$ Ramping-up capability requirement at time t (MW)

$\mathrm{rr}^{\mathbf{rc}^{\downarrow}}_{t}$ Ramping-down capability requirement at time t (MW)

$\mathrm{rr}^{\mathbf{rc}^{\mathrm{str}}}_{t}$ Short-term ramping capability requirement at time t (MW)

π^{-} Global energy shortage penalty price (\$/MW)

π^{+} Global energy surplus penalty price (\$/MW)

π^{reg} Regulating reserve shortage penalty price (\$/MW)

π^{rspin} Regulating plus spinning reserve shortage penalty price (\$/MW)

π^{op} Operating reserve shortage penalty price (\$/MW)

$\pi^{\mathrm{rc}^{\uparrow}}$ Ramping-up capability shortage penalty price (\$/MW)

$\pi^{\mathrm{rc}^{\downarrow}}$ Ramping-down capability shortage penalty price (\$/MW)

$\pi^{\mathrm{rc}^{\mathrm{str}}}$ Short-term ramping capability shortage penalty price (\$/MW)

$\pi^{\mathrm{f}}{}_{e,t}$ Transmission violation penalty price for branch e at time t (\$/MW)

A.4 Constraints

Generation resource constraints

1. Energy step clearing

$$\mathbf{pg}^{k}_{g,t,s} \leq \mathrm{CF}_{g,t}\bar{\mathrm{pg}}^{k}_{g,t,s} \qquad \forall g,t,s,k \qquad \text{(A.1)}$$

2. Energy dispatch target

$$\mathbf{pg}_{g,t,s} = \sum_{k} \mathbf{pg}^{k}_{g,t,s} + \mathrm{CF}_{g,t}\underline{\mathrm{pg}}_{g,t,s} \qquad \forall g,t,s \qquad \text{(A.2)}$$

3. Cleared regulating reserve

$$2\mathbf{r}^{\mathrm{reg}}_{g,t,s} \leq \mathrm{RF}_{g,t}\mathrm{RA}^{\mathrm{reg}}_{g,t}(\bar{\mathrm{pg}}_{g,t,s} - \underline{\mathrm{pg}}_{g,t,s}) \qquad \forall g,t,s \qquad \text{(A.3)}$$

4. Cleared contingency reserve

$$\mathbf{r}^{\mathrm{spin}}_{g,t,s} \leq \mathrm{CF}_{g,t}\mathrm{RA}^{\mathrm{spin}}_{g,t}(\bar{\mathrm{pg}}_{g,t,s} - \underline{\mathrm{pg}}_{g,t,s}) \qquad \forall g,t,s \qquad \text{(A.4)}$$

$$\mathbf{r}^{\mathrm{s,on}}_{g,t,s} \leq \mathrm{CF}_{g,t}\mathrm{RA}^{\mathrm{s,on}}_{g,t}(\bar{\mathrm{pg}}_{g,t,s} - \underline{\mathrm{pg}}_{g,t,s}) \qquad \forall g,t,s \qquad \text{(A.5)}$$

$$\mathbf{r}^{\mathrm{s,off}}_{g,t,s} \leq (1 - \mathrm{CF}_{g,t})\mathrm{RA}^{\mathrm{s,off}}_{g,t}\bar{\mathrm{r}}^{\mathrm{s,off}}_{g,t} \qquad \forall g,t,s \qquad \text{(A.6)}$$

5. Maximum and minimum limit

$$\mathbf{pg}_{g,t,s} - \mathbf{r}^{\mathrm{reg}}_{g,t,s} - \mathbf{rc}^{\downarrow}_{g,t,s} \geq \mathrm{CF}_{g,t}\underline{\mathrm{pg}}_{g,t,s} \qquad \forall g,t,s \qquad \text{(A.7)}$$

$$\mathbf{pg}_{g,t,s} + \mathbf{r}^{\mathrm{reg}}_{g,t,s} + \mathbf{r}^{\mathrm{spin}}_{g,t,s} + \mathbf{r}^{\mathrm{s,on}}_{g,t,s} + \mathbf{rc}^{\uparrow}_{g,t,s} \leq \mathrm{CF}_{g,t}\bar{\mathrm{pg}}_{g,t,s} \qquad \forall g,t,s \qquad \text{(A.8)}$$

6. Ramp-up and -down

$$\mathbf{pg}_{g,t+1,s} \leq \mathbf{pg}_{g,t,s} + \Delta_t\, \mathrm{r}^{\uparrow}_{g,t} \qquad \forall g,t,s \qquad \text{(A.9)}$$

$$\mathbf{pg}_{g,t+1,s} \geq \mathbf{pg}_{g,t,s} - \Delta_t\, \mathrm{r}^{\downarrow}_{g,t} \qquad \forall g,t,s \qquad \text{(A.10)}$$

7. Regulating reserve ramp: regulating reserves must be deployable within 5 minutes

$$\mathbf{r}^{\mathrm{reg}}_{g,t,s} \leq 5 \times \mathrm{r}^{\uparrow}_{g,t} \qquad \forall g,t,s \qquad \text{(A.11)}$$

8. Contingency reserve ramp: contingency reserves must be deployable within 10 minutes

$$\mathbf{r}^{\mathrm{spin}}_{g,t,s} + \mathbf{r}^{\mathrm{s,on}}_{g,t,s} \leq 10 \times \mathrm{r}^{\uparrow}_{g,t} \qquad \forall g,t,s \qquad \text{(A.12)}$$

9. Ramping-up and -down capability: ramping-up and -down capability must be de-

ployable within 10 minutes

$$\mathbf{rc}^{\uparrow}_{g,t,s} \leq 10 \times (1 - \mathrm{IF}_{g,t})\Delta_t \, \mathrm{r}^{\uparrow}_{g,t} \qquad \forall g, t, s \qquad \text{(A.13)}$$

$$\mathbf{rc}^{\downarrow}_{g,t,s} \leq 10 \times (1 - \mathrm{IF}_{g,t})\Delta_t \, \mathrm{r}^{\downarrow}_{g,t} \qquad \forall g, t, s \qquad \text{(A.14)}$$

10. Short-term ramping capability: short-term ramping capability must be deployable within 30 minutes

$$\mathbf{rc}^{\mathrm{str}}_{g,t,s} \leq 30 \times (1 - \mathrm{IF}_{g,t})\Delta_t \, \mathrm{r}^{\uparrow}_{g,t} \qquad \forall g, t, s \qquad \text{(A.15)}$$

Reliability constraints

1. Nodal injection

$$\mathbf{p}_{i,t,s} = \sum_{g \in \mathcal{G}_i} \mathbf{pg}_{g,t,s} - \mathrm{pd}_{i,t,s} \qquad \forall i, t, s \qquad \text{(A.16)}$$

2. Global power balance

$$\sum_{i \in \mathcal{N}} \mathbf{p}_{i,t,s} = \boldsymbol{\delta}\mathbf{p}^{+}_{t,s} - \boldsymbol{\delta}\mathbf{p}^{-}_{t,s} \qquad \forall t, s \qquad \text{(A.17)}$$

3. Regulating reserve requirements

$$\sum_{g \in \mathcal{G}} \mathbf{r}^{\mathrm{reg}}_{g,t,s} \geq \mathrm{rr}^{\mathrm{reg}}_{t} - \boldsymbol{\delta}\mathbf{r}^{\mathrm{reg}}_{t,s} \qquad \forall t, s \qquad \text{(A.18)}$$

4. Regulating plus spinning reserve requirements

$$\sum_{g \in \mathcal{G}} \left(\mathbf{r}^{\mathrm{reg}}_{g,t,s} + \mathbf{r}^{\mathrm{spin}}_{g,t,s} \right) \geq \mathrm{rr}^{\mathrm{rspin}}_{t} - \boldsymbol{\delta}\mathbf{r}^{\mathrm{rspin}}_{t,s} \qquad \forall t, s \qquad \text{(A.19)}$$

5. Operating reserve requirements

$$\sum_{g \in \mathcal{G}} \left(\mathbf{r}^{\mathrm{reg}}_{g,t,s} + \mathbf{r}^{\mathrm{spin}}_{g,t,s} + \mathbf{r}^{\mathrm{s,on}}_{g,t,s} + \mathbf{r}^{\mathrm{s,off}}_{g,t,s} \right) \geq \mathrm{rr}^{\mathrm{op}}_{t} - \boldsymbol{\delta}\mathbf{r}^{\mathrm{op}}_{t,s} \qquad \forall t, s \qquad \text{(A.20)}$$

6. Ramping-up and -down capability reserve requirements

$$\sum_{g\in\mathcal{G}}\left(\mathbf{rc}^{\uparrow}_{g,t,s}\right) \geq \mathrm{rr}^{\mathbf{rc}^{\uparrow}}_{t} - \boldsymbol{\delta}\mathbf{rc}^{\uparrow}_{t,s} \qquad \forall t,s \qquad \text{(A.21)}$$

$$\sum_{g\in\mathcal{G}}\left(\mathbf{rc}^{\downarrow}_{g,t,s}\right) \geq \mathrm{rr}^{\mathbf{rc}^{\downarrow}}_{t} - \boldsymbol{\delta}\mathbf{rc}^{\downarrow}_{t,s} \qquad \forall t,s \qquad \text{(A.22)}$$

7. Short-term ramping capability reserve requirements

$$\sum_{g\in\mathcal{G}}\left(\mathbf{rc}^{\mathrm{str}}_{g,t,s}\right) \geq \mathrm{rr}^{\mathbf{rc}^{\mathrm{str}}}_{t} - \boldsymbol{\delta}\mathbf{rc}^{\mathrm{str}}_{t,s} \qquad \forall t,s \qquad \text{(A.23)}$$

Transmission constraints

1. PTDF

$$\mathbf{pf}_{e,t,s} = \sum_{i\in\mathcal{N}} \sigma_{e,i}\mathbf{p}_{i,t,s} \qquad \forall e,t,s \qquad \text{(A.24)}$$

2. Flowgate limits

$$\underline{\mathrm{pf}}_{e,t} - \boldsymbol{\delta}\mathbf{f}_{e,t,s} \leq \mathbf{pf}_{e,t,s} \leq \bar{\mathrm{pf}}_{e,t} + \boldsymbol{\delta}\mathbf{f}_{e,t,s} \qquad \forall e,t,s \qquad \text{(A.25)}$$

Non-anticipatory constraints

$$\mathbf{pg}_{g,1,s_1} = \mathbf{pg}_{g,1,s_2} \qquad \forall g\in\mathcal{G}, s_1, s_2 \in \mathcal{S} \qquad \text{(A.26)}$$

$$\mathbf{r}^{\mathrm{reg}}_{g,1,s_1} = \mathbf{r}^{\mathrm{reg}}_{g,1,s_2} \qquad \forall g\in\mathcal{G}, s_1, s_2 \in \mathcal{S} \qquad \text{(A.27)}$$

$$\mathbf{r}^{\mathrm{spin}}_{g,1,s_1} = \mathbf{r}^{\mathrm{spin}}_{g,1,s_2} \qquad \forall g\in\mathcal{G}, s_1, s_2 \in \mathcal{S} \qquad \text{(A.28)}$$

$$\mathbf{r}^{\mathrm{s,on}}_{g,1,s_1} = \mathbf{r}^{\mathrm{s,on}}_{g,1,s_2} \qquad \forall g\in\mathcal{G}, s_1, s_2 \in \mathcal{S} \qquad \text{(A.29)}$$

$$\mathbf{r}^{\mathrm{s,off}}_{g,1,s_1} = \mathbf{r}^{\mathrm{s,off}}_{g,1,s_2} \qquad \forall g\in\mathcal{G}, s_1, s_2 \in \mathcal{S} \qquad \text{(A.30)}$$

$$\mathbf{rc}^{\uparrow}_{g,1,s_1} = \mathbf{rc}^{\uparrow}_{g,1,s_2} \qquad \forall g\in\mathcal{G}, s_1, s_2 \in \mathcal{S} \qquad \text{(A.31)}$$

$$\mathbf{rc}^{\downarrow}_{g,1,s_1} = \mathbf{rc}^{\downarrow}_{g,1,s_2} \qquad \forall g\in\mathcal{G}, s_1, s_2 \in \mathcal{S} \qquad \text{(A.32)}$$

$$\mathbf{rc}^{\mathrm{str}}_{g,1,s_1} = \mathbf{rc}^{\mathrm{str}}_{g,1,s_2} \qquad \forall g\in\mathcal{G}, s_1, s_2 \in \mathcal{S} \qquad \text{(A.33)}$$

Non-negative variables constraints

$$\bar{\mathrm{pg}}^{k}_{g,t,s} \geq 0 \qquad \forall g,t,s,k \tag{A.34}$$

$$\mathbf{r}^{\text{reg}}_{g,t,s} \geq 0 \qquad \forall g,t,s \tag{A.35}$$

$$\mathbf{r}^{\text{spin}}_{g,t,s} \geq 0 \qquad \forall g,t,s \tag{A.36}$$

$$\mathbf{r}^{\text{s,on}}_{g,t,s} \geq 0 \qquad \forall g,t,s \tag{A.37}$$

$$\mathbf{r}^{\text{s,off}}_{g,t,s} \geq 0 \qquad \forall g,t,s \tag{A.38}$$

$$\mathbf{rc}^{\uparrow}_{g,t,s} \geq 0 \qquad \forall g,t,s \tag{A.39}$$

$$\mathbf{rc}^{\downarrow}_{g,t,s} \geq 0 \qquad \forall g,t,s \tag{A.40}$$

$$\mathbf{rc}^{\text{str}}_{g,t,s} \geq 0 \qquad \forall g,t,s \tag{A.41}$$

$$\boldsymbol{\delta}\mathbf{p}^{+}_{t,s} \geq 0 \qquad \forall t,s \tag{A.42}$$

$$\boldsymbol{\delta}\mathbf{p}^{-}_{t,s} \geq 0 \qquad \forall t,s \tag{A.43}$$

$$\boldsymbol{\delta}\mathbf{r}^{\text{reg}}_{t,s} \geq 0 \qquad \forall t,s \tag{A.44}$$

$$\boldsymbol{\delta}\mathbf{r}^{\text{rspin}}_{t,s} \geq 0 \qquad \forall t,s \tag{A.45}$$

$$\boldsymbol{\delta}\mathbf{r}^{\text{op}}_{t,s} \geq 0 \qquad \forall t,s \tag{A.46}$$

$$\boldsymbol{\delta}\mathbf{rc}^{\uparrow}_{t,s} \geq 0 \qquad \forall t,s \tag{A.47}$$

$$\boldsymbol{\delta}\mathbf{rc}^{\downarrow}_{t,s} \geq 0 \qquad \forall t,s \tag{A.48}$$

$$\boldsymbol{\delta}\mathbf{rc}^{\text{str}}_{t,s} \geq 0 \qquad \forall t,s \tag{A.49}$$

$$\boldsymbol{\delta}\mathbf{f}_{e,t,s} \geq 0 \qquad \forall e,t,s \tag{A.50}$$

A.5 Objective

The objective is the sum of the following objective terms.

1. No-load costs

$$\sum_{t\in\mathcal{T}}\sum_{g\in\mathcal{G}} \Delta_t \cdot \mathrm{CF}_{g,t} \cdot \mathrm{c}^{\text{NL}}_{g,t} \tag{A.51}$$

2. Energy dispatch costs

$$\sum_{g\in\mathcal{G}}\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}}\sum_{k} p_s \Delta_t C^k_{g,t} \mathbf{pg}^k_{g,t,s} \tag{A.52}$$

3. Reserve procurement costs

$$\sum_{g\in\mathcal{G}}\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} p_s \cdot \Delta_t \cdot \mathrm{c}_{g,t,s} \tag{A.53}$$

where

$$\mathrm{c}_{g,t,s} = \mathrm{c}^{\text{reg}}_{g,t}\mathbf{r}^{\text{reg}}_{g,t,s} + \mathrm{c}^{\text{spin}}_{g,t}\mathbf{r}^{\text{spin}}_{g,t,s} + \mathrm{c}^{\text{s,on}}_{g,t}\mathbf{r}^{\text{s,on}}_{g,t,s} + \mathrm{c}^{\text{s,off}}_{g,t}\mathbf{r}^{\text{s,off}}_{g,t,s}$$

4. Power balance violation penalties

$$\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} p_s\, \Delta_t\, \left(\pi^{-}\boldsymbol{\delta}\mathbf{p}^{-}_{t,s} + \pi^{+}\boldsymbol{\delta}\mathbf{p}^{+}_{t,s}\right) \tag{A.54}$$

5. Regulating reserve shortage penalties

$$\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} p_s\, \Delta_t\, \pi^{\text{reg}}\boldsymbol{\delta}\mathbf{r}^{\text{reg}}_{t,s} \tag{A.55}$$

6. Regulating-plus-spinning reserve shortage penalties

$$\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} p_s\, \Delta_t\, \pi^{\text{rspin}}\boldsymbol{\delta}\mathbf{r}^{\text{rspin}}_{t,s} \tag{A.56}$$

7. Operating reserve shortage penalties

$$\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} p_s\, \Delta_t\, \pi^{\text{op}}\boldsymbol{\delta}\mathbf{r}^{\text{op}}_{t,s} \tag{A.57}$$

8. Ramping-up and -down capability shortage penalties

$$\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} p_s\, \Delta_t\, \pi^{\text{rc}^{\uparrow}}\boldsymbol{\delta}\mathbf{rc}^{\uparrow}_{t,s} \tag{A.58}$$

$$\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} p_s\, \Delta_t\, \pi^{\text{rc}^{\downarrow}}\boldsymbol{\delta}\mathbf{rc}^{\downarrow}_{t,s} \tag{A.59}$$

9. Short-term ramping capability shortage penalties

$$\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} p_s\, \Delta_t\, \pi^{\text{rc}^{\text{str}}}\boldsymbol{\delta}\mathbf{rc}^{\text{str}}_{t,s} \tag{A.60}$$

10. Thermal limit violation penalties

$$\sum_{e\in\mathcal{E}}\sum_{t\in\mathcal{T}}\sum_{s\in\mathcal{S}} p_s\,\Delta_t\,\pi^{\mathrm{f}}{}_{e,t}\,\boldsymbol{\delta}\mathbf{f}_{e,t,s} \tag{A.61}$$

APPENDIX B

MORE INFORMATION OF DCOPF OPTIMIZATION PROXIES

B.1 Detailed Modeling of Compact Formulation

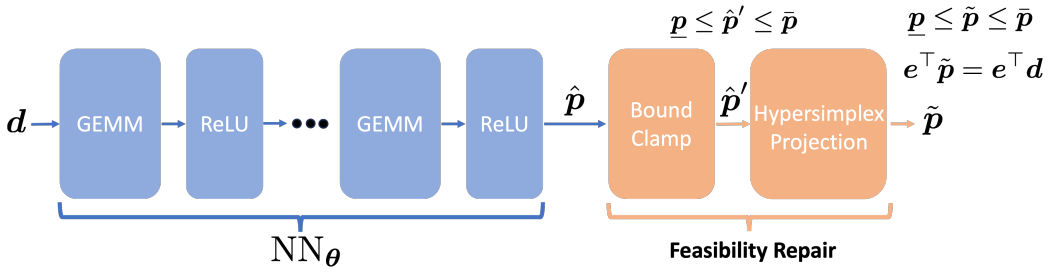


Figure B.1: Optimization Proxies for DCOPF

As shown in Figure B.1, the proposed DCOPF proxy integrates a multilayer perceptron (MLP), a bound clamping, and a hypersimplex projection layer. The MLP can be translated into a MILP by modeling its general matrix multiplication (GEMM) blocks with linear constraints and its ReLU activations using binary variables and linear constraints, as detailed in prior works [100, 82].

Bound Clamp The bound clamping layer is designed to enforce minimum and maximum generation limits (4.12e). It can be expressed as the composition of two ReLU layers, effectively clamping the input between specified maximum and minimum power generation limits:

$$\hat{\boldsymbol{p}}' = \mathrm{clamp}(\hat{\boldsymbol{p}}, \underline{\mathrm{pg}}, \bar{\mathrm{pg}}) = \min(\max(\hat{\boldsymbol{p}}, \underline{\mathrm{pg}}), \bar{\mathrm{pg}}) \tag{B.1a}$$

$$= -\mathrm{ReLU}(-\mathrm{ReLU}(\hat{\boldsymbol{p}} - \underline{\mathrm{pg}}) - \underline{\mathrm{pg}} + \bar{\mathrm{pg}}) + \bar{\mathrm{pg}}. \tag{B.1b}$$

The clamp operator clips all entries in $\hat{\boldsymbol{p}}$ into the range $[\bar{\mathrm{pg}}, \underline{\mathrm{pg}}]$. It can be formulated as linear constraints and binary decision variables, leveraging the ReLU MILP representation

from the aforementioned studies [100, 82].

Hypersimplex Projection The Hypersimplex Projection takes as input $\boldsymbol{p} \in \mathcal{H} = \{\mathbf{pg} \in \mathbb{R}^B | \underline{\mathrm{pg}} \leq \mathbf{pg} \leq \bar{\mathrm{pg}}\}$, and outputs $\tilde{\boldsymbol{p}} \in \mathcal{S}$, where $\mathcal{S}$ is the hypersimplex

$$\mathcal{S} = \{\mathbf{pg} \in \mathbb{R}^B | \underline{\mathrm{pg}} \leq \mathbf{pg} \leq \bar{\mathrm{pg}}, \boldsymbol{e}^\top \mathbf{pg} = \boldsymbol{e}^\top \mathrm{pd}\}.$$

The input vector $\mathbf{pg}$ must satisfy minimum and maximum generation limits (4.12e), and the output vector $\tilde{\boldsymbol{p}}$ jointly satisfies minimum/maximum generation limits (4.12e) and power balance constraint (4.12b). The projection adjusts all entries in $\mathbf{pg}$ uniformly until either the total power generation matches the total demand, or the entries in $\mathbf{pg}$ reach their bounds. Formally, $\tilde{\boldsymbol{p}}$ is obtained as the unique solution of the system of equations

$$\tilde{\mathbf{pg}} = \mathrm{clamp}(\hat{\mathbf{pg}}' + \delta, \underline{\mathrm{pg}}, \bar{\mathrm{pg}}), \tag{B.3a}$$

$$\boldsymbol{e}^\top \tilde{\mathbf{pg}} = \boldsymbol{e}^\top \mathrm{pd}, \tag{B.3b}$$

where $\delta \in \mathbb{R}$ is a scalar. Note that (B.3) can reduced to a uni-dimensional problem in δ by substituting out $\tilde{\boldsymbol{p}}$. Namely, letting $f(\delta) = \boldsymbol{e}^\top \mathrm{clamp}(\hat{\mathbf{pg}}' + \delta, \bar{\mathrm{pg}}, \underline{\mathrm{pg}})$, (B.3b) reduces to $f(\delta) = \boldsymbol{e}^\top \mathrm{pd}$. The uniqueness of the solution to (B.3) then follows from the fact that f is monotonically increasing. δ can be effectively computed using binary search (see Algorithm 8). It can be easily parallelized in PyTorch [148] with its subgradient computed using auto-differentiation.

Algorithm 8 Hypersimplex Projection via Binary Search

Input: Initial dispatch $\tilde{\text{pg}}$, Dispatch bounds $\bar{\text{pg}}$, $\underline{\text{pg}}$, demand pd, numerical tolerance ϵ

Initialize $\bar{\delta} = \max(\bar{\text{pg}} - \underline{\text{pg}})$, $\underline{\delta} = -\max(\bar{\text{pg}} - \underline{\text{pg}})$, $\delta = (\underline{\delta} - \bar{\delta})/2$, $D = \boldsymbol{e}^\top \text{pd}$

while $|\underline{\delta} - \bar{\delta}| \geq \epsilon$ or $|f(\delta) - D| \geq \epsilon$ **do**

  if $f(\delta) \geq D$ **then**

    $\bar{\delta} = \delta$

  else

    $\underline{\delta} = \delta$

  end if

  $\delta = (\underline{\delta} - \bar{\delta})/2$

end while

return δ

The Hypersimplex projection layer is then represented as an MILP by encoding the system of equations (B.3). Namely, constraint (B.3a) is encoded as an MILP following Equation (B.1), and constraint (B.3b) is linear. The compact optimality verification problem for DCOPF proxies thus reads:

$$\max_{\text{pd} \in \mathcal{X}} \quad \boldsymbol{c}^\top \mathbf{pg} + M^{\text{th}} \boldsymbol{e}^\top \boldsymbol{\xi}^{\text{th}} - \boldsymbol{c}^\top \tilde{\mathbf{pg}} - M^{\text{th}} \boldsymbol{e}^\top \tilde{\xi}^{\text{th}} \tag{B.4a}$$

$$\text{s.t.} \quad \hat{\mathbf{pg}} = \text{NN}_{\boldsymbol{\theta}}(\text{pd}), \tag{B.4b}$$

$$\tilde{\boldsymbol{f}} = \boldsymbol{H}(\tilde{\mathbf{pg}} - \text{pd}), \tag{B.4c}$$

$$\tilde{\xi}^{\text{th}} = \max\{\tilde{\boldsymbol{f}} - \bar{\text{pf}}, -\bar{\text{pf}} - \tilde{\boldsymbol{f}}, \mathbf{0}\}, \tag{B.4d}$$

$$(B.1), (B.3), \tag{B.4e}$$

$$(4.12b) - (4.12e), \tag{B.4f}$$

$$\boldsymbol{p} \in \mathbb{R}^B, \hat{\mathbf{pg}} \in \mathbb{R}^B, \tilde{\mathbf{pg}} \in \mathbb{R}^B, \boldsymbol{\xi}^{\text{th}} \in \mathbb{R}^E_+, \tilde{\xi}^{\text{th}} \in \mathbb{R}^E_+ \tag{B.4g}$$

Constraint (B.4b) encodes the inference of the neural network, which could be linearized by introducing binary decision variables [100, 82]. Constraint (B.4c) and (B.4d) compute

the thermal violation of the predicted dispatch. Before solving formulation (B.4), all clamp and $\max$ operators are linearized by introducing binary decision variables, which leads to a MILP.

B.2 Detailed Modeling of Bilevel Formulation

The Bilevel formulation of DCOPF proxy reads:

$$\max_{\text{pd}\in\mathcal{X}} \quad \boldsymbol{c}^\top \mathbf{pg} + M^{\text{th}} \boldsymbol{e}^\top \boldsymbol{\xi}^{\text{th}} - \boldsymbol{c}^\top \tilde{\mathbf{pg}} - M^{\text{th}} \boldsymbol{e}^\top \tilde{\xi}^{\text{th}} \tag{B.5a}$$

$$\text{s.t.} \quad (B.4b) - (B.4d), (B.4g) \tag{B.5b}$$

$$(B.1), (B.3), \tag{B.5c}$$

$$\mathbf{pg}, \boldsymbol{\xi}^{\text{th}} = \text{DCOPF}(\text{pd}) \tag{B.5d}$$

where lower level DCOPF(pd) outputs the optimal dispatch and thermal violations by solving formulation (4.12) with input pd.

To enhance the tractability, the work in [122] reformulates DCOPF(pd) with KKT conditions:

$$(4.12b) - (4.12f) \tag{B.6a}$$

$$\boldsymbol{e}\lambda + \boldsymbol{H}^\top \underline{\nu} - \boldsymbol{H}^\top \bar{\nu} + \underline{\mu} - \bar{\mu} = c \tag{B.6b}$$

$$\underline{\nu} + \bar{\nu} + \zeta = M^{\text{th}} \boldsymbol{e} \tag{B.6c}$$

$$\underline{\mu}, \bar{\mu}, \underline{\nu}, \bar{\nu}, \zeta \geq 0 \tag{B.6d}$$

$$\underline{\nu}^\top (\boldsymbol{H}\mathbf{pg} + \boldsymbol{\xi}^{\text{th}} + \bar{\text{pf}} - \boldsymbol{H}\text{pd}) = 0 \tag{B.6e}$$

$$\bar{\nu}^\top (-\boldsymbol{H}\mathbf{pg} + \boldsymbol{\xi}^{\text{th}} + \bar{\text{pf}} + \boldsymbol{H}\text{pd}) = 0 \tag{B.6f}$$

$$\underline{\mu}^\top (\mathbf{pg} - \underline{\text{pg}}) = 0 \tag{B.6g}$$

$$\bar{\mu}^\top (\bar{\text{pg}} - \mathbf{pg}) = 0 \tag{B.6h}$$

$$\zeta^\top \boldsymbol{\xi}^{\text{th}} = 0 \tag{B.6i}$$

Constraints (4.12b) - (4.12f) model the primal feasibility. Constraints (B.6b) - (B.6d) model the dual feasibility, where $\bar{\mu}$ and $\underline{\mu}$ denote the dual variables for the upper and lower bounds in Constraint (4.12e), respectively. The $\underline{\nu}, \bar{\nu}$ are the dual variables for the upper and lower thermal limits in Constraints (4.12d) and (4.12c), respectively. λ denotes the dual variable for the power balance constraint (4.12b). ζ denotes the dual variable for the non-negativity of thermal violations. Constraints (B.6e)-(B.6i) model the complementary slackness, which can be reformulated as an MILP using the standard big-M formulation. Note that valid bounds on all primal variables can be derived from the primal formulation. Dual variables $\underline{\nu}, \bar{\nu}, \zeta$ are naturally bounded by M^{th}. For the remaining dual variables, a large M value is selected.

Finally, the reformulated Bilevel formulation reads

$$\max_{\text{pd}\in\mathcal{X}} \quad \boldsymbol{c}^\top \mathbf{pg} + M^{\text{th}} \boldsymbol{e}^\top \boldsymbol{\xi}^{\text{th}} - \boldsymbol{c}^\top \tilde{\mathbf{pg}} - M^{\text{th}} \boldsymbol{e}^\top \tilde{\xi}^{\text{th}} \tag{B.7a}$$

$$\text{s.t.} \quad (B.4b) - (B.4d), (B.4g) \tag{B.7b}$$

$$(B.1), (B.3), \tag{B.7c}$$

$$(4.12b) - (4.12f) \tag{B.7d}$$

$$(B.6b) - (B.6i) \tag{B.7e}$$

B.3 More results on optimality verification of DCOPF proxies

Table B.1 reports the final objective and solving time on small power systems, where Compact formulation can solve all instances to optimality within 1 minute.

Table B.1: Comparisons of PGA-VFA and Compact formulation on final objective and solving time on small power systems. T2PGA reports the time that the compact formulation takes to reach the same primal objective of the proposed PGA-VFA. Compact formulation is very effective at finding optimal solutions for small systems i.e., it solves all instances to optimality within 1 minute.

		PGA-VFA		Compact		
System	$\%u$	ObjVal ($)	Time (s)	ObjVal ($)	Time (s)	T2PGA (s)
57	0	127.36	2.17	141.36	0.31	0.10
	1	137.35	1.44	142.29	0.59	0.11
	2	143.70	4.31	151.54	0.54	0.12
	5	171.53	3.35	179.12	1.10	1.10
	10	175.89	2.28	432.20	1.21	1.10
	20	1527.72	5.59	3837.33	0.54	0.30
118	0	752.29	1.43	6216.97	1.20	0.74
	1	4615.37	6.96	7917.56	2.07	1.17
	2	4615.37	7.61	9607.68	3.08	2.00
	5	4615.37	4.44	14329.88	12.08	3.00
	10	5448.69	6.26	68556.32	14.14	6.67
	20	6048.77	9.04	69109.46	59.41	17.27

APPENDIX C

MORE INFORMATION OF KNAPSACK

Recall the knapsack problem formulation

$$\max_{\boldsymbol{y}} \quad \boldsymbol{v}^T \boldsymbol{y} \tag{C.1a}$$

$$s.t. \quad \boldsymbol{w}^T \boldsymbol{y} \leq l, \tag{C.1b}$$

$$\boldsymbol{y} \in \{0,1\}^K. \tag{C.1c}$$

C.1 Detailed Modeling of Compact Formulation

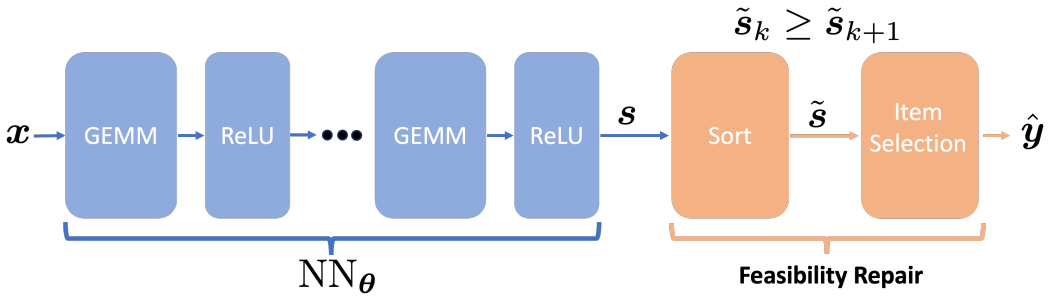


Figure C.1: Optimization Proxies for Knapsack

Figure C.1 illustrates the Knapsack proxies. First, a fully-connected neural network with ReLU activation predicts a score for each item, where a higher score indicating higher desirability of the item. Then, a feasibility repair step sorts the items by the predicted score in a descending order and adds items to the knapsack following the order until reaching the knapsack's capacity limit, addressing constraint (C.1b). The MLP is trained disjointly with the feasibility repair steps in the training due to their non-differentiability. The neural network is trained using supervised learning to minimize the distances of the predicted scores to the ground truth: $l(\boldsymbol{s}, \boldsymbol{s}^*) = \|\boldsymbol{s} - \boldsymbol{s}^*\|_2$. The ground truth score is computed heuristically: $\boldsymbol{s}^* = \frac{\boldsymbol{v}}{\boldsymbol{w}}$, where $\boldsymbol{v}$ denotes the value of the items and $\mathbf{w}$ denotes the weight

of the items. Note that the score is exactly the solution of the continuous relaxation of Knapsack problems.

The following part focuses on formulating the sorting and item selection operators into MILP.

Sorting Denote the predicted score $\boldsymbol{s}$ from the neural network, the sorting operation sorts the item id with the score descendingly. The sorting operation is modeled by introducing a permutation matrix:

$$\tilde{\boldsymbol{s}} = \boldsymbol{p}\boldsymbol{s}, \tag{C.2a}$$

$$\tilde{\boldsymbol{s}}_k \geq \tilde{\boldsymbol{s}}_{k+1}, \forall k \in [K] \tag{C.2b}$$

$$\sum_{i}^{K} \boldsymbol{p}_{i,k} = 1, \forall k \in [K], \tag{C.2c}$$

$$\sum_{k}^{K} \boldsymbol{p}_{i,k} = 1, \forall i \in [K], \tag{C.2d}$$

$$\boldsymbol{p} \in \{0,1\}^{K \times K}, \tag{C.2e}$$

$$\boldsymbol{s} \in \mathbb{R}_{+}^{K}, \tilde{\boldsymbol{s}} \in \mathbb{R}_{+}^{K}. \tag{C.2f}$$

The $\tilde{\boldsymbol{s}}$ denotes the score after the sortation. The $\boldsymbol{p}$ denotes the permutation matrix to convert predicted score $\boldsymbol{s}$ to sorted score $\tilde{\boldsymbol{s}}$. Constraint (C.2c)-(C.2e) define the permutation matrix and constraint (C.2a) and (C.2b) ensure the perturbation matrix encodes the orders of sorting the score descendingly.

Item Selection After obtaining the permutation matrix, the item selection operator could be modeled as:

$$\tilde{\boldsymbol{y}} = \boldsymbol{p}\hat{\boldsymbol{y}}, \tag{C.3a}$$

$$\tilde{\boldsymbol{w}} = \boldsymbol{p}\boldsymbol{w}, \tag{C.3b}$$

$$\boldsymbol{w}^T\hat{\boldsymbol{y}} \leq l, \tag{C.3c}$$

$$\tilde{\boldsymbol{y}}_k \geq \tilde{\boldsymbol{y}}_{k+1}, \forall k \in [K] \tag{C.3d}$$

$$\sum_{i=1}^{k} \tilde{\boldsymbol{w}}_i \geq (1 - \tilde{\boldsymbol{y}}_k)(l+1), \forall k \in [K] \tag{C.3e}$$

$$\tilde{\boldsymbol{y}} \in \{0,1\}^K, \hat{\boldsymbol{y}} \in \{0,1\}^K. \tag{C.3f}$$

Constraint (C.3a) and (C.3b) permutate the item action and weights. Constraint (C.3c) ensures the total weight of the added items is less than the knapsack capacity. Constraint (C.3d) ensures items are added sequentially following the sorted scores Constraint (C.3e) ensures that the addition of items continues only if adding another item does not exceed the knapsack's capacity. Finally, following the reformulation of the operations and Formulation (4.5), the compact optimality verification for Knapsack proxies could be written as a MILP and readily be solved by solvers like Gurobi [20].

Finally, the optimality verification formulation reads:

$$\max_{\boldsymbol{x}=\{\boldsymbol{v},l\}\in\mathcal{X}} \quad \boldsymbol{v}^T\boldsymbol{y} - \boldsymbol{v}^T\hat{\boldsymbol{y}} \tag{C.4a}$$

$$s.t. \quad \boldsymbol{s} = \text{NN}_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{C.4b}$$

$$(C.2), (C.3), \tag{C.4c}$$

$$(C.1b), (C.1c) \tag{C.4d}$$

The objective function (C.4a), constraints (C.4b) and (C.4c) are linearized by introducing binary decision variables, which gives a MILP.

REFERENCES

[1] D. Bienstock and S. Mattia, "Using mixed-integer programming to solve power grid blackout problems," *Discret. Optim.*, vol. 4, pp. 115–141, 2007.

[2] P. Van Hentenryck, C. Coffrin, and R. Bent, "Vehicle routing for the last mile of power system restoration," 2010.

[3] A. Wood and B. Wollenberg, "Power generation, operation, and control," 1984.

[4] B. Stott, J. Jardim, and O. Alsaç, "Dc power flow revisited," *IEEE Transactions on Power Systems*, vol. 24, pp. 1290–1300, 2009.

[5] F. Capitanescu *et al.*, "State-of-the-art, challenges, and future trends in security constrained optimal power flow," *Electric Power Systems Research*, vol. 81, no. 8, pp. 1731–1741, 2011.

[6] B. M. Mithun, S. Muthyala, and S. Maheswarapu, "Security constraint optimal power flow (scopf) – a comprehensive survey," *International Journal of Computer Applications*, vol. 11, pp. 24–28, 2010.

[7] T. N. dos Santos and A. Diniz, "A dynamic piecewise linear model for dc transmission losses in optimal scheduling problems," *IEEE Transactions on Power Systems*, vol. 26, pp. 508–519, 2011.

[8] D. Fitiwi, L. Olmos, M. Rivier, F. Cuadra, and I. Pérez-Arriaga, "Finding a representative network losses model for large-scale transmission expansion planning with renewable energy sources," *Energy*, vol. 101, pp. 343–358, 2016.

[9] B. Eldridge, R. O'Neill, and A. Castillo, "An improved method for the dcopf with losses," *IEEE Transactions on Power Systems*, vol. 33, pp. 3779–3788, 2018.

[10] A. Helseth, "A linear optimal power flow model considering nodal distribution of losses," *2012 9th International Conference on the European Energy Market*, pp. 1–8, 2012.

[11] Y.-C. Chang, W.-T. Yang, and C.-C. Liu, "A new method for calculating loss coefficients [of power systems]," *IEEE Transactions on Power Systems*, vol. 9, no. 3, pp. 1665–1671, 1994.

[12] T. N. dos Santos and A. L. Diniz, "A dynamic piecewise linear model for dc transmission losses in optimal scheduling problems," *IEEE Transactions on Power Systems*, vol. 26, no. 2, pp. 508–519, 2011.

[13] M. Shahidehpour, H. Yamin, and Z. Li, *Market operations in electric power systems: forecasting, scheduling, and risk management*. John Wiley & Sons, 2003.

[14] E. Litvinov, T. Zheng, G. Rosenwald, and P. Shamsollahi, "Marginal loss modeling in lmp calculation," *IEEE Transactions on Power Systems*, vol. 19, pp. 880–888, 2004.

[15] B. Eldridge, R. P. O'Neill, and A. Castillo, "Marginal loss calculations for the dcopf," 2017.

[16] C. Coffrin, P. Van Hentenryck, and R. Bent, "Approximating line losses and apparent power in ac power flow linearizations," *2012 IEEE Power and Energy Society General Meeting*, pp. 1–8, 2012.

[17] X. Ma, Y. Chen, and J. Wan, "Midwest iso co-optimization based real-time dispatch and pricing of energy and ancillary services," in *2009 IEEE Power Energy Society General Meeting*, 2009, pp. 1–6.

[18] MISO, "Real-time energy and operating reserve market software formulations and business logic," MISO, Business Practices Manual Energy and Operating Reserve Markets Attachment D.

[19] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[20] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2021.

[21] PACE, *Partnership for an Advanced Computing Environment (PACE)*, 2017.

[22] S. Babaeinejadsarookolaee *et al.*, *The power grid library for benchmarking ac optimal power flow algorithms*, 2021. arXiv: 1908.02788 [math.OC].

[23] T. W. K. Mak, L. Shi, and P. Van Hentenryck, *Phase transitions for optimality gaps in optimal power flows a study on the french transmission network*, 2018.

[24] X. Ma, H. Song, M. Hong, J. Wan, Y. Chen, and E. Zak, "The security-constrained commitment and dispatch for Midwest ISO day-ahead co-optimized energy and ancillary service market," in *2009 IEEE Power & Energy Society General Meeting*, 2009, pp. 1–8.

[25] C. Wang, C. Hansen, D. Chatterjee, R. Merring, J. Li, and S. Li, *Ramp Capability Modeling in MISO Dispatch and Pricing*, Jun. 2016.

[26] Y. Chen, "Addressing uncertainties through improved reserve product design," *IEEE Transactions on Power Systems*, vol. 38, no. 4, pp. 3911–3923, 2023.

[27] MISO, *Short-term reserves*, Mar. 2021.

[28] California ISO, *Business practice manual for market operations*, 2024.

[29] B. Wang and B. F. Hobbs, "Flexiramp market design for real-time operations: Can it approach the stochastic optimization ideal?" In *2013 IEEE Power Energy Society General Meeting*, 2013, pp. 1–5.

[30] New York ISO, *Manual 12 transmission and dispatch operations manual*, 2024.

[31] MISO, *Real-time energy and operating reserve market software formulations and business logic*, Business Practices Manual Energy and Operating Reserve Markets Attachment D, 2024.

[32] ERCOT, *ERCOT Nodal Protocols – Section 6: Adjustment Period and Real-Time Operations*, 2024.

[33] M. E. Khodayar, S. D. Manshadi, H. Wu, and J. Lin, "Multiple period ramping processes in day-ahead electricity markets," *IEEE Transactions on Sustainable Energy*, vol. 7, no. 4, pp. 1634–1645, 2016.

[34] S. Sreekumar, S. Yamujala, K. C. Sharma, R. Bhakar, S. P. Simon, and A. S. Rana, "Flexible ramp products: A solution to enhance power system flexibility," *Renewable and Sustainable Energy Reviews*, vol. 162, p. 112 429, 2022.

[35] V. Dvorkin, S. Delikaraoglou, and J. M. Morales, "Setting reserve requirements to approximate the efficiency of the stochastic dispatch," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1524–1536, 2019.

[36] L. Xie and M. D. Ilic, "Model predictive economic/environmental dispatch of power systems with intermittent resources," in *2009 IEEE Power & Energy Society General Meeting*, 2009, pp. 1–6.

[37] Y. Chen, V. Ganugula, J. Williams, J. Wan, and Y. Xiao, "Resource transition model under miso mip based look ahead commitment," in *2012 IEEE Power and Energy Society General Meeting*, 2012, pp. 1–8.

[38] H. Hui, C.-N. Yu, and S. Moorty, "Reliability unit commitment in the new ERCOT nodal electricity market," in *2009 IEEE Power & Energy Society General Meeting*, 2009, pp. 1–8.

[39] A. L. Ott, "Evolution of computing requirements in the PJM market: Past and future," in *IEEE PES General Meeting*, 2010, pp. 1–4.

[40] L. Xie, X. Luo, and O. Obadina, "Look-ahead dispatch in ERCOT: Case study," in *2011 IEEE Power and Energy Society General Meeting*, 2011, pp. 1–3.

[41] H. Hui, C.-N. Yu, R. Surendran, F. Gao, S. Moorty, and X. Xu, "Look ahead to the unforeseen: ERCOT's nonbinding look-ahead SCED study," in *2013 IEEE Power & Energy Society General Meeting*, 2013.

[42] D.-H. Choi and L. Xie, "Data perturbation-based sensitivity analysis of real-time look-ahead economic dispatch," *IEEE Transactions on Power Systems*, vol. 32, no. 3, pp. 2072–2082, 2017.

[43] L. Varawala, G. Dán, M. R. Hesamzadeh, and R. Baldick, "A Generalised Approach for Efficient Computation of Look Ahead Security Constrained Optimal Power Flow," *European Journal of Operational Research*, vol. 310, no. 2, pp. 477–494, 2023.

[44] L. A. Roald, D. Pozo, A. Papavasiliou, D. K. Molzahn, J. Kazempour, and A. Conejo, "Power systems optimization under uncertainty: A review of methods and applications," *Electric Power Systems Research*, vol. 214, p. 108 725, 2023.

[45] Z. Li, W. Wu, B. Zhang, and B. Wang, "Adjustable Robust Real-Time Power Dispatch With Large-Scale Wind Power Integration," *IEEE Transactions on Sustainable Energy*, vol. 6, no. 2, pp. 357–368, 2015.

[46] A. Lorca and X. A. Sun, "Adaptive robust optimization with dynamic uncertainty sets for multi-period economic dispatch under significant wind," *IEEE Transactions on Power Systems*, vol. 30, no. 4, pp. 1702–1713, 2015.

[47] U. Yıldıran, "Robust multi-stage economic dispatch with renewable generation and storage," *European Journal of Operational Research*, vol. 309, no. 2, pp. 890–909, 2023.

[48] C. Zhao, Y. Chen, Y. Guan, Q. Wang, and X. Wang, "Short-Term Power Generation Scheduling via Robust Optimization†," in *Handbook of Clean Energy Systems*. John Wiley & Sons, Ltd, 2015, pp. 1–12, ISBN: 9781118991978.

[49] A. Mashlakov, T. Kuronen, L. Lensu, A. Kaarna, and S. Honkapuro, "Assessing the performance of deep learning models for multivariate probabilistic energy forecasting," *Applied Energy*, vol. 285, p. 116 405, 2021.

[50] J. W. Messner, P. Pinson, J. Browell, M. B. Bjerregård, and I. Schicker, "Evaluation of wind power forecasts – An up-to-date view," *Wind Energy*, vol. 23, no. 6, pp. 1461–1481, 2020.

[51] T. Werho, J. Zhang, V. Vittal, Y. Chen, A. Thatte, and L. Zhao, *Scenario generation of wind farm power for real-time system operation*, 2021. arXiv: 2106.09105 [stat.AP].

[52] M. Reynolds, I. Satkauskas, J. Maack, D. Sigler, and W. Jones, "Scenario creation and power-conditioning strategies for operating power grids with two-stage stochastic economic dispatch," in *2020 IEEE Power & Energy Society General Meeting (PESGM)*, 2020, pp. 1–5.

[53] A. Papavasiliou, Y. Mou, L. Cambier, and D. Scieur, "Application of stochastic dual dynamic programming to the real-time dispatch of storage under renewable supply uncertainty," *IEEE Transactions on Sustainable Energy*, vol. 9, no. 2, pp. 547–558, 2018.

[54] G. Zhang and J. McCalley, "Stochastic look-ahead economic dispatch with flexible ramping product," in *2015 IEEE Power & Energy Society General Meeting*, 2015, pp. 1–5.

[55] Y. Gu and L. Xie, "Stochastic look-ahead economic dispatch with variable generation resources," *IEEE Transactions on Power Systems*, vol. 32, no. 1, pp. 17–29, 2016.

[56] A. J. Conejo, E. Castillo, R. Minguez, and R. Garcia-Bertrand, *Decomposition techniques in mathematical programming: engineering and science applications*. Springer Science & Business Media, 2006.

[57] M. Pereira, "Optimal stochastic operations scheduling of large hydroelectric systems," *International Journal of Electrical Power & Energy Systems*, vol. 11, no. 3, pp. 161–169, 1989.

[58] M. Pereira and L. Pinto, "Multi-stage stochastic optimization applied to energy planning," *Mathematical Programming*, vol. 52, no. 1-3, pp. 359–375, 1991, Cited by: 830.

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

[60] C. Chen, Y. Guo, and L. Tong, "Pricing multi-interval dispatch under uncertainty part ii: Generalization and performance," *IEEE Transactions on Power Systems*, vol. 36, no. 5, pp. 3878–3886, 2020.

[61] Y. Guo, C. Chen, and L. Tong, "Pricing multi-interval dispatch under uncertainty part i: Dispatch-following incentives," *IEEE Transactions on Power Systems*, vol. 36, no. 5, pp. 3865–3877, 2021.

[62] B. Hua, D. A. Schiro, T. Zheng, R. Baldick, and E. Litvinov, "Pricing in multi-interval real-time markets," *IEEE Transactions on Power Systems*, vol. 34, no. 4, pp. 2696–2705, 2019.

[63] Z. Tan, T. Cheng, Y. Liu, and H. Zhong, "Extensions of the locational marginal price theory in evolving power systems: A review," *IET Generation, Transmission & Distribution*, vol. 16, no. 7, pp. 1277–1291, 2022.

[64] J. Zhao, T. Zheng, and E. Litvinov, "A multi-period market design for markets with intertemporal constraints," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 3015–3025, 2020.

[65] X. Ma, Y. Chen, and J. Wan, "The security-constrained economic dispatch for mid-west iso's real-time co-optimized energy and ancillary service market," in *IEEE PES General Meeting*, 2009.

[66] J. Benders, "Partitioning procedures for solving mixed-variables programming problems.," *Numerische Mathematik*, vol. 4, pp. 238–252, 1962.

[67] R. T. Rockafellar and R. J.-B. Wets, "Scenarios and policy aggregation in optimization under uncertainty," *Mathematics of operations research*, vol. 16, no. 1, pp. 119–147, 1991.

[68] M. Fischetti, I. Ljubić, and M. Sinnl, "Benders decomposition without separability: A computational study for capacitated facility location problems," *European Journal of Operational Research*, vol. 253, no. 3, pp. 557–569, 2016.

[69] M. Fischetti and D. Salvagnin, "An in-out approach to disjunctive optimization," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, A. Lodi, M. Milano, and P. Toth, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 136–140, ISBN: 978-3-642-13520-0.

[70] P. Bonami, D. Salvagnin, and A. Tramontani, "Implementing automatic benders decomposition in a modern mip solver," in *Integer Programming and Combinatorial Optimization*, D. Bienstock and G. Zambelli, Eds., Cham: Springer International Publishing, 2020, pp. 78–90, ISBN: 978-3-030-45771-6.

[71] B. Gisin, Q. Gu, J. V. Mitsche, S. Tam, and H. Chen, ""Perfect Dispatch" - as the measure of PJM real time grid operational performance," in *IEEE PES General Meeting*, 2010, pp. 1–8.

[72] M. Chatzos, M. Tanneau, and P. Van Hentenryck, "Data-driven time series reconstruction for modern power systems research," *Electric Power Systems Research*, vol. 212, p. 108 589, 2022.

[73] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017.

[74] H. Zhang, R. Zandehshahvar, M. Tanneau, and P. Van Hentenryck, "Weather-informed probabilistic forecasting and scenario generation in power systems," *arXiv preprint arXiv:2409.07637*, 2024.

[75] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[76] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, "Temporal fusion transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[77] N. Colombo, "Normalizing flows for conformal regression," in *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

[78] C. Xu, H. Jiang, and Y. Xie, "Conformal prediction for multi-dimensional time series by ellipsoidal sets," in *Forty-first International Conference on Machine Learning*, 2024.

[79] C. Xu and Y. Xie, "Sequential predictive conformal inference for time series," in *Proceedings of the 40th International Conference on Machine Learning*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., ser. Proceedings of Machine Learning Research, vol. 202, PMLR, Jul. 2023, pp. 38 707–38 727.

[80] P. Cunningham and S. J. Delany, "K-Nearest Neighbour classifiers – A tutorial," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–25, 2021.

[81] ARPA-e, U.S. Department of Energy, "The GO Competition," https://https://gocompetition.energy.gov/, 2020.

[82] R. Bunel, I. Turkaslan, P. H. S. Torr, P. Kohli, and M. P. Kumar, *A unified view of piecewise linear neural network verification*, 2018. arXiv: 1711.00455 [cs.AI].

[83] S. Gowal, K. Dvijotham, R. Stanforth, T. Mann, and P. Kohli, "A dual approach to verify and train deep networks,"

[84] K. D. Dvijotham, R. Stanforth, S. Gowal, C. Qin, S. De, and P. Kohli, "Efficient neural network verification with exactness characterization," in *Uncertainty in Artificial Intelligence*, PMLR, 2020, pp. 497–507.

[85] S. Dathathri *et al.*, "Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming," in *Advances in Neural Information*

Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 5318–5331.

[86] C. Tsay, J. Kronqvist, A. Thebelt, and R. Misener, "Partition-based formulations for mixed-integer optimization of trained reLU neural networks," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[87] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma, *Strong mixed-integer programming formulations for trained neural networks*, 2020. arXiv: 1811.01988 [math.OC].

[88] N. Kochdumper, C. Schilling, M. Althoff, and S. Bak, "Open-and closed-loop neural network verification using polynomial zonotopes," in *NASA Formal Methods Symposium*, Springer, 2023, pp. 16–36.

[89] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Advances in Neural Information Processing Systems*, vol. 31, pp. 4939–4948, 2018.

[90] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang, "A convex relaxation barrier to tight robustness verification of neural networks," *Advances in Neural Information Processing Systems*, vol. 32, pp. 9835–9846, 2019.

[91] K. Xu *et al.*, "Automatic perturbation analysis for scalable certified robustness and beyond," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[92] K. Xu *et al.*, "Fast and Complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers," in *International Conference on Learning Representations*, 2021.

[93] S. Wang *et al.*, "Beta-CROWN: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[94] H. Zhang *et al.*, "A branch and bound framework for stronger adversarial attacks of ReLU networks," in *Proceedings of the 39th International Conference on Machine Learning*, vol. 162, 2022, pp. 26 591–26 604.

[95] H. Zhang *et al.*, "General cutting planes for bound-propagation-based neural network verification," *Advances in Neural Information Processing Systems*, 2022.

[96] H. Zhang *et al.*, *α, β-crown: Verified intelligence alpha-beta-crown*, https://github.com/Verified-Intelligence/alpha-beta-CROWN, Team from CMU, UCLA, Drexel University, Columbia University, UIUC, RWTH Aachen University, Sun Yat-sen

University, University of Michigan. Advisors: Kolter, Zico; Hsieh, Cho-Jui; Jana, Suman; Li, Bo; Lin, Xue., 2023.

[97] C. Brix, M. N. Müller, S. Bak, T. T. Johnson, and C. Liu, "First three years of the international verification of neural networks competition (vnn-comp)," *International Journal on Software Tools for Technology Transfer*, pp. 1–11, 2023.

[98] A. Caprara and M. Locatelli, "Global optimization problems and domain reduction strategies," *Mathematical Programming*, vol. 125, pp. 123–137, 2010.

[99] A. M. Gleixner, T. Berthold, B. Müller, and S. Weltge, "Three enhancements for optimization-based bound tightening," *Journal of Global Optimization*, vol. 67, pp. 731–757, 2017.

[100] V. Tjeng, K. Xiao, and R. Tedrake, "Evaluating robustness of neural networks with mixed integer programming," *arXiv preprint arXiv:1711.07356*, 2017.

[101] S. Gowal *et al.*, *On the effectiveness of interval bound propagation for training verifiably robust models*, 2019. arXiv: 1810.12715 [cs.LG].

[102] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proc. ACM Program. Lang.*, vol. 3, no. POPL, Jan. 2019.

[103] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 630–637.

[104] C. Tran, F. Fioretto, and P. Van Hentenryck, "Differentially private and fair deep learning: A lagrangian dual approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 9932–9939.

[105] A. Velloso and P. Van Hentenryck, "Combining deep learning and optimization for preventive security-constrained dc optimal power flow," *IEEE Transactions on Power Systems*, vol. 36, no. 4, pp. 3618–3628, 2021.

[106] X. Pan, T. Zhao, M. Chen, and S. Zhang, "Deepopf: A deep neural network approach for security-constrained dc optimal power flow," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 1725–1735, 2020.

[107] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[108] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[109] W. Song, X. Chen, Q. Li, and Z. Cao, "Flexible job-shop scheduling via graph neural network and deep reinforcement learning," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1600–1610, 2022.

[110] C. K. Joshi, T. Laurent, and X. Bresson, "An efficient graph convolutional network technique for the travelling salesman problem," *arXiv preprint arXiv:1906.01227*, 2019.

[111] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*, PMLR, 2017, pp. 136–145.

[112] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in neural information processing systems*, vol. 32, 2019.

[113] J. Tordesillas, J. P. How, and M. Hutter, "Rayen: Imposition of hard convex constraints on neural networks," *arXiv preprint arXiv:2307.08336*, 2023.

[114] S. Park, W. Chen, T. W. Mak, and P. Van Hentenryck, "Compact optimization learning for ac optimal power flow," *arXiv preprint arXiv:2301.08840*, 2023.

[115] M. Li, S. Kolouri, and J. Mohammadi, "Learning to solve optimization problems with hard linear constraints," *IEEE Access*, 2023.

[116] W. Chen, M. Tanneau, and P. V. Hentenryck, "End-to-end feasible optimization proxies for large-scale economic dispatch," *IEEE Transactions on Power Systems*, pp. 1–12, 2023.

[117] X. Chen, Y. Li, R. Umarov, X. Gao, and L. Song, "Rna secondary structure prediction by learning unrolled algorithms," in *International Conference on Learning Representations*, 2020.

[118] P. L. Donti, D. Rolnick, and J. Z. Kolter, "DC3: A learning method for optimization with hard constraints," in *International Conference on Learning Representations*, 2021.

[119] D. Scieur, G. Gidel, Q. Bertrand, and F. Pedregosa, "The curse of unrolling: Rate of differentiating through optimization," *Advances in Neural Information Processing Systems*, vol. 35, pp. 17 133–17 145, 2022.

[120] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.

[121] H. Sun, Y. Shi, J. Wang, H. D. Tuan, H. V. Poor, and D. Tao, "Alternating differentiation for optimization layers," *arXiv preprint arXiv:2210.01802*, 2022.

[122] R. Nellikkath and S. Chatzivasileiadis, "Physics-informed neural networks for minimising worst-case violations in dc optimal power flow," in *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, IEEE, 2021, pp. 419–424.

[123] M. Bojarski *et al.*, *End to end learning for self-driving cars*, 2016. arXiv: 1604.07316 [cs.CV].

[124] K. D. Julian, J. Lopez, J. S. Brush, M. P. Owen, and M. J. Kochenderfer, "Policy compression for aircraft collision avoidance systems," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, IEEE, 2016, pp. 1–10.

[125] W. Chen, S. Park, M. Tanneau, and P. Van Hentenryck, "Learning optimization proxies for large-scale security-constrained economic dispatch," *Electric Power Systems Research*, vol. 213, p. 108 566, 2022.

[126] C. Szegedy *et al.*, *Intriguing properties of neural networks*, 2014. arXiv: 1312.6199 [cs.CV].

[127] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, *Learning optimal power flow: Worst-case guarantees for neural networks*, 2020. arXiv: 2006.11029 [cs.AI].

[128] S. Chen, E. Wong, J. Z. Kolter, and M. Fazlyab, "Deepsplit: Scalable verification of deep neural networks via operator splitting," *IEEE Open Journal of Control Systems*, vol. 1, pp. 126–140, 2022.

[129] A. Raghunathan, J. Steinhardt, and P. Liang, *Semidefinite relaxations for certifying robustness to adversarial examples*, 2018. arXiv: 1811.01057 [cs.LG].

[130] S. Wang *et al.*, "Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 29 909–29 921.

[131] G. Qiu, M. Tanneau, and P. Van Hentenryck, "Dual Conic Proxies for AC Optimal Power Flow," *2024 Power Systems Computation Conference, Paris*, Jun. 2023.

[132] J. Kotary, F. Fioretto, and P. Van Hentenryck, "Fast approximations for job shop scheduling: A lagrangian dual deep learning method," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 7239–7246.

[133] J. Kotary, F. Fioretto, and P. Van Hentenryck, "Learning hard optimization problems: A data generation perspective," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 981–24 992, 2021.

[134] N. Karalias and A. Loukas, "Erdos goes neural: An unsupervised learning framework for combinatorial optimization on graphs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6659–6672, 2020.

[135] H. P. Wang, N. Wu, H. Yang, C. Hao, and P. Li, "Unsupervised learning for combinatorial optimization with principled objective relaxation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 444–31 458, 2022.

[136] S. Park and P. Van Hentenryck, "Self-supervised primal-dual learning for constrained optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 4052–4060.

[137] A. Caprara, M. Carvalho, A. Lodi, and G. J. Woeginger, "A complexity and approximability study of the bilevel knapsack problem," in *Integer Programming and Combinatorial Optimization: 16th International Conference, IPCO 2013, Valparaíso, Chile, March 18-20, 2013. Proceedings 16*, Springer, 2013, pp. 98–109.

[138] J. T. Moore and J. F. Bard, "The Mixed Integer Linear Bilevel Programming Problem," *Operations Research*, vol. 38, no. 5, pp. 911–921, 1990.

[139] A. Ben-Tal, A. Nemirovski, and L. El Ghaoui, "Robust optimization," 2009.

[140] D. Bertsimas, D. B. Brown, and C. Caramanis, "Theory and applications of robust optimization," *SIAM review*, vol. 53, no. 3, pp. 464–501, 2011.

[141] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[142] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[143] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *International Conference on Machine Learning*, PMLR, 2017, pp. 146–155.

[144] W. Chen, M. Tanneau, and P. Van Hentenryck, "Real-time risk analysis with optimization proxies," *arXiv preprint arXiv:2310.00709*, 2023.

[145] T. Zhao, X. Pan, M. Chen, and S. Low, "Ensuring dnn solution feasibility for optimization problems with linear constraints," in *The Eleventh International Conference on Learning Representations*, 2022.

[146] S. Babaeinejadsarookolaee *et al.*, "The power grid library for benchmarking ac optimal power flow algorithms," *arXiv preprint arXiv:1908.02788*, 2019.

[147] C. Ferrari, M. N. Mueller, N. Jovanović, and M. Vechev, "Complete verification via multi-neuron relaxation guided branch-and-bound," in *International Conference on Learning Representations*, 2022.

[148] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.