High-Performance Computational Methods to Improve the Functioning of Energy Markets

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Timothy Andrew Bjustrom Holt

under the supervision of
Prof. Olaf Schenk and Prof. Igor Pivkin

April 2024

Dissertation Committee

Prof. Ernst Wit	Università della Svizzera italiana, Switzerland
Prof. Stefan Wolf	Università della Svizzera italiana, Switzerland
Dr. Slaven Peles	Oak Ridge National Laboratory, USA
Prof. Simon Scheidegger	Université de Lausanne, Switzerland

Dissertation accepted on 8 April 2024

Research Advisor

Prof. Olaf Schenk

Co-Advisor

Prof. Igor Pivkin

PhD Program Director

Prof. Walter Binder, Prof. Stefan Wolf

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Timothy Andrew Bjustrom Holt
Lugano, 8 April 2024

The computer can be used as a tool
to liberate and protect people,
rather than to control them.

Hal Finney, 1992

Abstract

This thesis delves into the enhancement of energy markets through the application of high-performance computational methods. The overarching objective is to bolster the efficiency, accuracy, and scalability of the computational methods that are used to analyze and operate these markets through both applied industrial and fundamental academic contributions. The work is presented through four interconnected chapters that underscore the vital role of high-performance computing in reshaping the energy landscape. Through: massively parallel deployment of power market optimization models on many-core HPC clusters; modeling techniques to improve performance and accuracy of power market optimization models; a data-driven refinement approach to power generation unit commitment; and computational techniques to scalably identify pricing cycles in retail gasoline markets, this work advances energy market analysis tools in terms of efficiency, scalability, and usefulness.

Acknowledgements

Thank you all those who have mentored me along the way.

Contents

Introduction **1**

Research Motivation 1

Structure 2

I Power Market Models **4**

Forward to Part I 6

1 Massively parallel deployment of power market optimization models on many-core HPC clusters **15**

1.1 Introduction 15

1.1.1 Motivation and Background 15

1.1.2 Research Context 16

1.1.3 Relevant Literature 17

1.1.4 Contributions and Organization 18

1.2 Power Grid Models 19

1.2.1 Zone-Based Market Model 19

1.2.2 Optimal Power Flow Model 20

1.2.3 Unit Commitment 21

1.2.4 Swiss Grid Benchmark Model 22

1.3 Parallelism Treatment 24

1.4 Solution Strategy 26

1.4.1 SLURM Workload Manager 28

1.4.2 Greasy Meta Scheduler 28

1.5 Numerical Experiments 29

1.5.1 Performance Analysis of the Optimization Algorithms . . . 30

1.5.2 Node-Level Parallelism and Memory Bottleneck 32

1.5.3 Analysis of Parallelism Modes 33

1.5.4 Massively Parallel Execution 34

1.5.5 Massively Parallel Execution with Meta Scheduling 35

1.5.6 Verification of the Optimal Parallelism Degree Prediction . 36

1.6 Conclusions 37

2 Modeling techniques to improve performance and accuracy of power market optimization models 39

2.1 Introduction 39
2.2 Model Reformulation 40
2.2.1 Pyomo Model 40
2.2.2 Fusion Model 42
2.3 High-Speed Solution Methods 43
2.4 Multi-Day Modeling 45

II Power Generation Unit Commitment 47

Forward to Part II 49

3 A data-driven refinement approach to power generation unit commitment 56

3.1 Introduction 56
3.1.1 Definitions 58
3.2 Exascale Grid Optimization (ExaGO) Toolkit 60
3.3 Data-Driven Unit Commitment Algorithm 61
3.3.1 Algorithm Objectives 61
3.3.2 Load Shed Recourse Phase 62
3.3.3 Pruning Phase 64
3.4 Numerical Experiments 64
3.4.1 Setup 64
3.4.2 Performance Evaluation Criteria 66
3.4.3 South Carolina Grid Results 68
3.4.4 Texas Grid Results 70
3.5 Discussion 71
3.6 Conclusion 72
3.7 Scientific Software: ExaGO 73
3.7.1 Scalable ACOPF Optimization Techniques 74
3.7.2 Hardware Accelerated Optimization 75
3.7.3 Towards Scalable Stochastic Optimal Power Flow 77
3.8 Appendix: ExaGO ACOPF Forumulation 77
3.8.1 Variables and bounds 78
3.8.2 Objective Function 78
3.8.3 Equality constraints 79
3.8.4 Inequality constraints 80

III Cycle Detection in Gasoline Markets **82**
Forward to Part III . . . 84

4 Computational techniques to scalably identify pricing cycles in retail gasoline markets **86**
4.1 Introduction . . . 86
4.1.1 Related Literature, Contributions, and Replication Package. 90
4.2 Theoretical Background . . . 91
4.2.1 What Are Edgeworth Cycles? . . . 91
4.2.2 Are Edgeworth Cycles Competitive or Collusive? . . . 93
4.3 Data and Manual Classification . . . 94
4.3.1 Data Sources and Preparation . . . 94
4.3.2 Manual-Classification Procedures . . . 95
4.3.3 Rationale for Daily Frequency and Quarterly Window . . . 97
4.4 Models and Methods for Automatic Detection . . . 99
4.4.1 Existing Methods Mostly Focus on Asymmetry . . . 99
4.4.2 Our Proposals to Capture Cyclicality . . . 101
4.4.3 Optimization of Parameter Values ("Training") . . . 106
4.5 Results . . . 107
4.6 How Much Data Do We Need? . . . 110
4.7 Economic and Policy Implications . . . 112
4.7.1 Cycles and Margins . . . 113
4.7.2 Additional Findings . . . 115
4.7.3 Exploratory Data Analysis . . . 116
4.8 Practical Recommendations . . . 120
4.9 Conclusion . . . 121
4.10 Appendix A: Methodological Details and Simulations . . . 123
4.10.1 A.1 Details of the New Methods . . . 123
4.10.2 A.2 Parameter Optimization . . . 127
4.10.3 A.3 Performance on Simulated Cycles . . . 128
4.11 Appendix B: Additional Results . . . 138
4.11.1 B.1 Combining Methods 1–4 . . . 138
4.11.2 B.2 Variants of Methods 5–7 . . . 139

Conclusion **140**

Bibliography **143**

Introduction

Research Motivation

Efficient use of energy resources is one of the key challenges that our civilization must confront in order to continue flourishing in this century. The risks posed by excessive carbon dioxide emissions from energy use are driving significant transition from electricity generation technologies that are dispatchable and centralized to those that are intermittent and decentralized. These intermittent decentralized generation resources qualitatively change power grid forecasting and control problems by introducing significant weather dependent stochastic factors to the power supply and dispatch side, while massively increasing the number of decision variables in optimization problems. While the spotlight is on the push to develop and deploy renewable energy generation, the importance of effective grid planning and control systems to maintain a reliable grid under the increasingly chaotic conditions introduced by distributed stochastic intermittent generation cannot be understated. One of the key hurdles to a sustainable energy future is thus developing market-based control systems that aide in the efficient production and allocation of energy resources under conditions of increasing stochasticity. Developing such systems will be an enormous interdisciplinary effort involving domain scientists and engineers, economists and financial institutions, and computational scientists and software engineers. This thesis presents both applied industrial and fundamental academic contributions to this effort by connecting advanced computational technologies with domain knowledge to develop:

1. Practical methods and software that have been put into production in the power trading industry to aide in electricity price discovery subject to expected weather conditions and other stochastic factors; and

2. More fundamental algorithms and computational techniques that aim to advance the state-of-the-art in power grid planning and control, and the

detection of retail gasoline pricing cycles.

New frontiers in scientific discovery lie at the nexus of domain expertise and computing. It is thus incumbent upon my generation of computational scientists to not only implement scientific methods on computer hardware, but also leverage the expertise of colleagues on multidisciplinary teams to develop computational technology that advances the boundaries of human knowledge. Throughout my doctoral studies I have had the privilege of learning both these skills by working on multidisciplinary teams together with domain experts from economics, finance, electrical engineering, and physics to execute on projects that exploit the state-of-the-art in computing to provide solutions to practical problems facing society today.

Structure

The thesis is presented in three parts, each representing a different research project involving energy systems, data analysis, and computing that I undertook during the course of my doctoral studies. The parts presented are:

Part I: Power Market Models — Research on models to forecast efficient electricity prices under stochastic weather and economic conditions.

Chapter 1: Massively parallel deployment of power market optimization models on many-core HPC clusters — Presenting methods and algorithms to efficiently deploy several power market models on diverse hardware considering multi-level parallelism, memory bottlenecks, and efficient scheduling. Chapter based on papers [81, 80].

Chapter 2: Modeling techniques to improve performance and accuracy of power market optimization models — Presenting theory and numerical experiments on the effectiveness of throughput optimization of power market models using model reformulation, high speed solution methods, and multi-day modeling.

Part II: Power Generation Unit Commitment — Approaches to solve the combinatorial Unit Commitment problem, which involves selecting the optimal subset of power generators to participate in the power market for a given time period.

Chapter 3: A data-driven refinement approach to power generation unit commitment — Presenting a novel approach and algorithms to

scalably solve the power generation unit commitment problem considering a large number of stochastic scenarios. Chapter based on paper [74].

Part III: Cycle Detection in Gasoline Markets — Using advanced data analysis techniques to systematically identify specific patterns in price signals.

Chapter 4: Computational techniques to scalably identify pricing cycles in retail gasoline markets — A scientific evaluation of diverse computational models, both parametric and non-parametric, to identify Edgeworth Cycles in retail gasoline prices in large Australian and German data sets. Chapter based on paper [73].

I hope that the contributions laid out in these chapters will prove valuable to researchers dedicated to advancing the understanding and efficiency of energy systems and markets. It is my aspiration that the outcomes of this thesis will not only enhance the academic discourse within the field, but also play a role in fostering a sustainable future characterized by energy abundance.

Part I

Power Market Models

Forward to Part I: Data Analysis for Power Market Forecasting

The Demand–Generation Balance Megatrend

We are currently undergoing structural changes to power supply infrastructure, particularly in Western Europe and the United States. This is the trend from centralized dispatchable power production to decentralized stochastic power production. Significant adoption of the decentralized stochastic power production provided by solar and wind infrastructure makes maintaining equilibrium in the grid very difficult. This is because the grid must maintain a generation-demand balance at all times. We can represent this with

$$\underbrace{\sum_{g\in\mathcal{G}^c} p_g}_{\substack{\text{dispatchable}\\ \text{generation}}} + \underbrace{\sum_{g\in\mathcal{G}^s} p_g}_{\substack{\text{stochastic}\\ \text{generation}}} = \underbrace{\sum_{d\in\mathcal{D}^c} p_d}_{\substack{\text{dispatchable}\\ \text{demand}}} + \underbrace{\sum_{d\in\mathcal{D}^s} p_d}_{\substack{\text{stochastic}\\ \text{demand}}}, \quad \forall t \in \mathcal{T}, \tag{1}$$

where p is power in megawatts, $\mathcal{G}^c$ and $\mathcal{G}^s$ are the sets of dispatchable (controllable) and stochastic generators, $\mathcal{D}^c$ and $\mathcal{D}^s$ are the sets of dispatchable (controllable) and stochastic loads (demand)[1], and $\mathcal{T}$ is time[2]. Traditionally, the demand side is heavily dominated by stochastic loads, that is to say, consumers demand electricity exogenously from this balance equation, and the capacity of controllable loads is very small, coming from energy storage devices, demand response[3] programs, or smart grid[4] devices [14]. This means that to maintain equilibrium we must depend on the supply side, particularly controllable generation. Traditionally, the quasi-totality of generators have been dispatchable units with some throttling capability – allowing for easy maintenance of equilibrium.

[1] This equation and the associated control problem are significantly complicated by the fact that they are subject to network topology, that is, supply and demand must be equal at all vertices of the network. For this part of the discussion we will ignore network topology, but it should be kept in mind as a major complicating factor.

[2] On very short time scales, the balance of this equation is maintained by the inertia of the large rotating masses in the system. This allows us to focus on the control problem on longer discrete time scales.

[3] "Demand response" refers to programs that provide incentives for power consumers to reduce their demand at times when the grid operator is struggling to maintain the equilibrium of Eq. 1. Such initiatives are growing rapidly in certain markets, but it is still very early on the technology adoption timeline for demand response.

[4] "Smart grid" has become something of a buzz-word, but it often describes automated and decentralized demand response schemes.

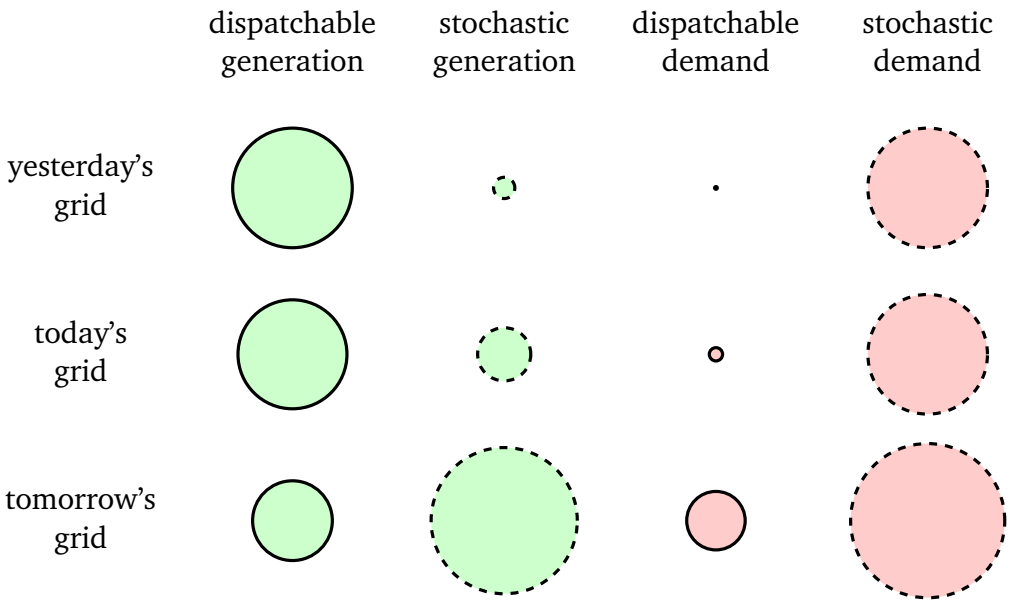


Figure 1: *Conceptual representation of supply and demand trends of electrical infrastructure. Circle size represents typical power output/consumption.*

This is changing, with the recent trend of shutting down dispatchable thermal generators and replacing them with stochastic renewable generators [89]. Figure 1 gives a representation of the trends as they are evolving and planned. The precise sizes shouldn't be taken literally, as this is a conceptual representation, but the trends are as pictured.

Replacing the controllable generation with stochastic generation makes the control problem significantly harder, and also makes forecasting of future grid conditions more important to allow for planning of operations. Being able to plan in advance becomes especially important as responses to grid conditions become more diverse, with more parties to coordinate, and as different types of energy storage devices play a larger role.

Weather is the Major Source of Stochasticity

Electricity demand is relatively predictable, with typical short-term load forecasting accuracy of within 1-3% of actual load [53]. Electricity consumption exhibits strong seasonal trends as well as daily and weekly cyclicality [53]. These trends however can be accurately estimated by models using solely historical load, the calendar, and economic growth independent variables [106]. The major source of stochasticity in load comes from the weather [118, 111], which can be thought

of as a bias term on the relatively consistent load signal. Temperature deviations from room temperature induce large loads for heating or cooling which, together with an ensemble of other weather dependent consumption behaviors, cause weather to be one of the most important factors affecting electricity markets. What we seek is a model that can estimate how weather will impact the grid. Given that weather is a local phenomenon and grids are large, spanning states, countries, and even continents, the model should be capable of mapping local weather conditions to local production and consumption patterns, but also be able to do this for many locations and model the interdependencies between connected locations.

Given the highly uncertain nature of weather, this makes power forecasting a particularly interesting field of study from a computational perspective. Uncertainty in weather can be modeled using monte-carlo style weather forecasts. This type of approach is used by The European Centre for Medium-Range Weather Forecasts (ECMWF) in the ensemble forecasting method. Numerical weather prediction models are nonlinear dynamical systems that provide results based on the evolution of some initial conditions. Each time step in the model inherently contains some forecast error, and these errors compound over time, leading to more significant forecast error for longer time horizons. The ECMWF ensemble forecasting methodology provides some uncertainty quantification of forecast error by providing an set of equally probable forecasts generated from initial condition perturbations [90]. Concretely, these ensembles manifest in 50 equally probable weather forecasts – giving us 50 fully independent probable future scenarios under which to estimate grid conditions.

Power Price Forecasting Workflow

The price of power at a given location and time in the future is a valuable piece of information. Market participants such as power generators and large consumers use this information in planing their operations. The more accurate this figure is, the further ahead of time, the more valuable it is for planning purposes. As such, we desire a model that is capable of estimating this value. Such a model should be capable of translating weather forecasts, the dominant source of stochasticity, into probabilistic price forecasts. The models presented in this work, fit into a specific workflow which is depicted in Figure 2. Part I describes step 4 of the workflow in detail - Run scenarios through the market model, but this is only one part of the process that results in the commercially valuable forecasts. Next we will briefly describe the key elements of each step.

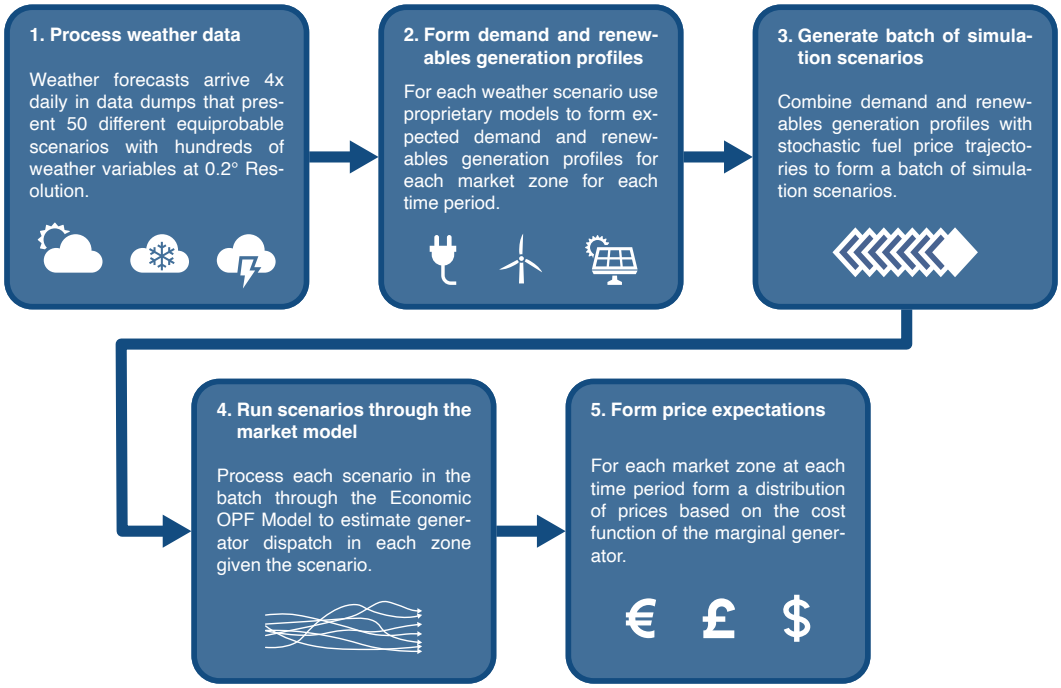


Figure 2: *Modeling process summary diagram from weather scenarios to power price expectations*

Process weather data — The ECMWF ensemble forecasts of 50 equally probable weather scenarios are distributed at 6 hour intervals. The data contains model outputs from the numerical weather models in hourly time steps, at a resolution of 0.2° for a forecast horizon of two weeks. Forecast horizons longer than two weeks are possible with ensembles [25], however the error carried in longer term forecasts limits the usefulness for financial decision making. The data contains variables such as temperature, {u, v, w} components of wind, geopotential, specific humidity, and percipitation, stratified across many different pressure levels[5] [87]. The first step is about identifying which of the myriad of variables are relevant for the workflow, and efficiently extracting those and parsing them into the format that is suitable for the next step in the workflow.

Form demand and renewable generation profiles — Weather is introduced to the grid on the demand side through loads that fluctuate with weather such as heating and cooling, and on the generation side through generators whose output is correlated with weather variables such as wind, solar, and hydro power. Typi-

[5] pressure levels are points or layers of atmospheric pressure used in weather modeling that correlate with altitudes.

cally, these values are estimated with data-driven models with the many weather variables as the independent variables and clever data processing and modeling techniques to account for non-linearities in the relationships between weather variables and forecast variables. Forecasting each of these values is a complex subject which is out of the scope of this thesis. I refer the interested reader to the following summary papers on load forecasting [135], wind power forecasting [66], and solar power forecasting [145], each of which does a nice job of introducing the current modeling paradigms that comprise this second step in the workflow.

Generate batch of simulation scenarios — In this step, the demand and renewables generation profiles for each of the weather scenarios are combined with the other stochastic variables that are important to the power system. The most important variables here are fuel prices, carbon prices, and some economic scenarios such as generator or line outages or geopolitical issues. The combination of all of these scenarios yields a batch of scenarios for a given time horizon that comprises on the order of 10s of thousands of individual simulations.

Run scenarios through the market model — The OPF market model aims to allocate a generation level to each generator in the model such that results in the universal lowest global generation cost. Each of the scenarios can be run independently through the market model so parallelization is trivial. The market model will be covered in detail in the next subsection.

Form price expectations — This is the last step in the workflow that finally yields the desired estimates of future electricity prices. An estimated price can be observed for each hour and in each market zone by taking the dual of the demand-generation balance constraint to determine the marginal generation cost in each of the market zones. The prices for each hour in each market zone can be examined across all of the simulation scenarios to form a distribution of the price random variable.

Optimal Power Flow Market Model

Given that power price is a random variable, more valuable than an estimate of the expected value is an estimate of its distribution. The distribution shows us not only the expected price, but quantifies the amount of uncertainty carried in

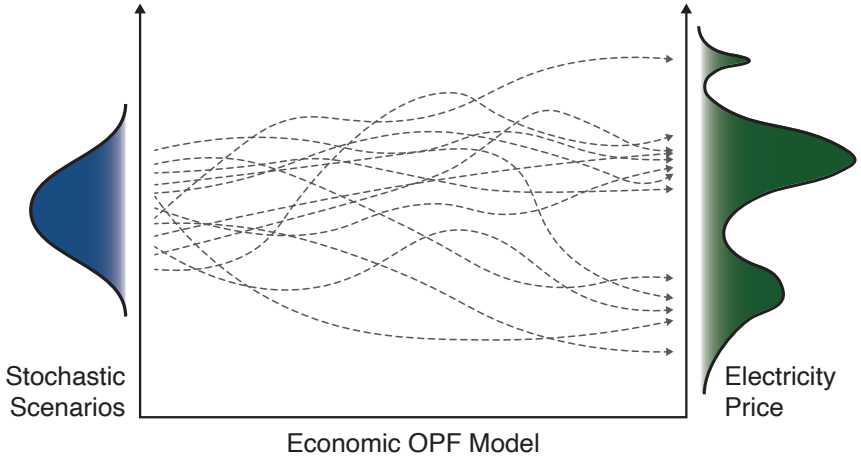


Figure 3: *Economic OPF model allows the translation of uncertain weather forecasts into a distribution of prices that allows modeling of financial risks.*

that estimate as well as the probabilities of extreme outcomes. Such detailed information is extremely valuable from a risk management perspective as it allows market participants to better understand and limit their risk exposure. Figure 3 shows the principle behind such an economic optimal power flow model that takes some set of stochastic scenarios and translates it into a distribution of future power prices.

The market model discussed in Part I is an optimal power flow model that is designed to capture economically relevant effects while avoiding the complexity of an explicitly modeled grid. As such, the model is built to replicate the European power market structure, where demand is aggregated into market zones, which loosely correspond to European nations, and grid infrastructure is abstracted away. The only transmission lines that are explicitly modeled are the market zone interconnects that transfer power between market zones. All generators of material size are modeled. A conceptual representation of this model is depicted in Figure 4. The model spans 43 market zones, 600 generators of 13 different types, and is set up for multi-period simulations between 40 and 256 hours.

The objective of this model is to find the lowest generation cost to meet demand in all market zones. This is done by adjusting the power output decision variables at each of the 600 generators. Since generators have heterogenous production costs, the optimal solution will saturate the generators in order of increasing cost, with the marginal generator in each market zone setting the price for that zone in that time period. As such, the estimated power prices for each

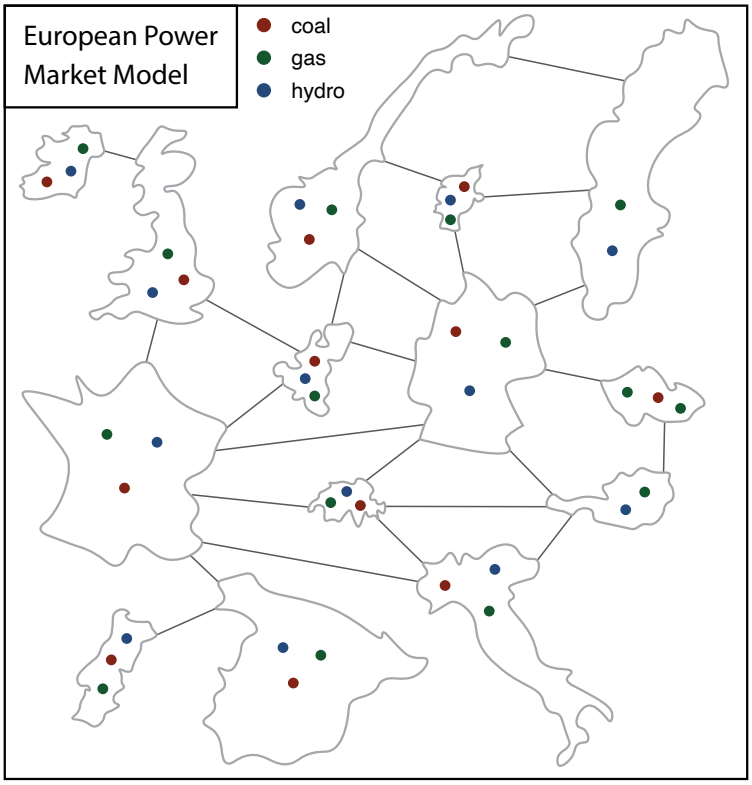


Figure 4: *Conceptual representation of the European Power Market Model. Model optimizes over 43 market zones, 13 generator types, 600 individual generators, and from 40 to 256 time periods.*

zone at each time are the duals of constraints

$$\underbrace{\sum_{g \in \mathcal{G}^z} p_g}_{\substack{\text{market zone} \\ \text{generation}}} + \underbrace{\sum_{b \in \mathcal{B}^z} i_b}_{\substack{\text{market zone} \\ \text{power imports}}} - \underbrace{\sum_{d \in \mathcal{D}^z} p_d}_{\substack{\text{market zone} \\ \text{demand}}} - \underbrace{\sum_{b \in \mathcal{B}^z} e_b}_{\substack{\text{market zone} \\ \text{power exports}}} = 0, \quad \forall t \in \mathcal{T}, \forall z \in \mathcal{Z}, \quad (2)$$

where $\mathcal{T}$ is the set of all time periods, $\mathcal{Z}$ is the set of all market zones, and $\mathcal{B}^z$ is the set of branches connected to a given market zone. This constraint fixes generation in a market zone equal to demand in that market zone, net of power imports / exports for all time (time indices omitted for simplicity). If imports and exports were unconstrained, we would expect a unique price in all zones, however the transmission lines are heavily capacity constrained, so the prices across the continent vary significantly.

The stochastic scenarios in this model manifest in the form of net demand profiles for the various market zones, which is defined as aggregate demand

minus non dispatchable power generation. Different net demand levels in each of the market zones will lead to different generation profiles at the 600 generators modeled. In addition to stochastic demand scenarios, stochastic trajectories for fuel prices are also explored and combined with the demand scenarios. The combinatorial effect from fuel prices and weather / demand scenarios, combined with the constant updates of weather forecasts yields large batches of simulations on the order of hundreds of thousands per day.

Efficient Deployment of the Market Model

Any time we face problem structures where computational tasks can be broken down into fully independent sub-problems, such as we see with these batches of independent scenarios, computational scientists are happy, since these types of problems are ideally suited to exploit the massively parallel computational capabilities of supercomputers. Assuming we have scalable methods and algorithms to parallelize computation of these individual weather scenarios, we can quantify the uncertainty in forecasts of grid conditions, while holding computation wall time[6] constant, simply by increasing computational resources. The following subsection will discuss how we exploit these independent sub-problems to forecast power prices in parallel, outputting rich distributions of power prices expected by the model.

The problem structure of the market model scales linearly in the number of scenarios. That is to say that if solving a scenario takes t seconds of computation time, then solving 100 scenarios takes on the order of $100t$ seconds of computation time. Parallelization of such computations is theoretically trivial, since no advanced decomposition or partitioning algorithms are required, and there is no data exchange needed between computation of scenarios. In practicality however, efficiently deploying these parallel computations on a real-world high-performance compute cluster such that throughput of computation is maximized is a challenging problem.

In trading, market participants who most quickly translate new information into price opinions will be able to react to news before competitors. Minimizing the latency of the power price forecasting workflow is therefore essential. In addition, increasing the throughput of the workflow for a given computational resource allows for more rich modeling by adding extra scenarios. Therefore, the models should be deployed in such a way that maximizes throughput and mini-

[6] Wall time refers to the amount of time that you would measure if you observe a clock on the wall while the computations are being completed.

mizes latency. This gives us a scheduling problem on a many-core cluster for an extremely data-intensive application with heterogenous workloads. Part I of this thesis is dedicated to deployment, scheduling, and performance optimizations that achieve this end of decreasing latency and increasing throughput of power market models.

Chapter 1 deals with deployment of OPF market models on many-core HPC clusters and is based on papers [81, 80]. Chapter 2 deals with modeling paradigms and optimizations that decrease computation time of given model runs.

Chapter 1

Massively parallel deployment of power market optimization models on many-core HPC clusters

1.1 Introduction

Recent trends in the power grid operations and integration of intermittent renewable energy sources (RES) impose great demand on computational resources. The large number of power grid scenarios that need to be analyzed require not only parallel processing algorithms but also efficient execution strategies for a large number of loosely coupled tasks. These can improve utilization of the computational infrastructure required by individual jobs that need to be scheduled on the available computing resources.

1.1.1 Motivation and Background

In order to operate the power grid in a secure and reliable way, a plethora of parameters need to be considered. These are parameters such as weather, fuel prices, and available transmission capacity between market zones [56]. The difficultly of accurately modeling these systems is constantly increasing given the volatility induced by the rapid integration of intermittent RES into grids. Additional challenges are associated with uncertainty quantification of the model parameters and their sensitivity analysis. All these factors contribute to a large number of power grid scenarios that need to be rapidly solved in order to provide power grid operators tools required to control the complex power grid systems and manage the associated uncertainty.

Most of these problems can benefit from parallel processing, often built into the simulation frameworks or available in the off the shelf solvers such as HiOP [32], PIPS [8], or Beltistos [79]. Nonetheless, the question of the optimal level of parallelism arises and is left up to the end user to decide. This decision, however, requires the knowledge of the underlying architecture, since the excessive level of parallelism might introduce many bottlenecks on the hardware level and result in a significant slowdown of the overall processing time. This work provides an analysis of the computational setup that could guide users of such parallel tools and help to achieve high-throughput data analytics.

1.1.2 Research Context

Power grid dynamics are typically modeled as multistage stochastic unit commitment (UC) problems [132, 113, 59], optimal power flow (OPF) [58, 62] problems, or economic dispatch (ED) [9, 10]. Uncertainty is incorporated using stochastic programming techniques based on scenario trees in which the uncertainty is known at each node. After applying various scenario generation schemes based either on expert knowledge, artificial intelligence, or Monte Carlo simulation, the stochastic UC becomes a large optimization problem. Due to the large-scale nature of the problem, the computational complexity is addressed by decomposing the problem into smaller subproblems and utilizing parallel processing. Benders decomposition, Lagrangian relaxation, augmented Lagrangian methods, or progressive hedging are usually among the methods of choice [63, 117], since parallelization of these solution algorithms is straightforward.

Stochastic UC is a risk-neutral model that is concerned with the optimization of expected payoff. In order to enable risk modeling, individual stochastic trajectories, represented by a single UC problem, are analyzed independently. Similarly as before, a large number of model evaluations are required. With this method one can obtain all possible price trajectories resulting from the selected scenarios. The resulting price distributions inform the risk management processes that are crucial to applications such as energy trading [80].

Additionally, many model parameters are determined experimentally, using the expert knowledge or based on historical data, with the exact values not available. In order to properly evaluate the effects of the uncertain parameters, one needs to perform an uncertainty quantification [54, 108, 127]. It quantifies the confidence of the model output given the uncertainty in the model parameters. Sensitivity analysis is tightly linked to uncertainty quantification and is the process of quantifying the fraction of the output uncertainty that can be explained

by individual parameters. However, global sensitivity analysis presents computational challenges due to the large number of input-output samples needed to estimate the uncertainty contributions. In order to perform the sensitivity analysis, parallel and high-throughput processing techniques are essential.

Similarly to the continuous counterparts, the mixed integer linear programming (MIP) models are used in the stochastic frameworks. Many aspects of real-life problems are modeled using discrete variables, including on-line status of generator units [113, 59], transformer tap ratios [75], models of the storage devices [59], or demand flexibility models [88, 86]. The resulting MIP problems are solved using algorithms usually based on the dual simplex (DS) or interior point (IP) method extended by heuristics to deal with the integrality constraints, including branch and cut, cutting planes, or many advanced presolving techniques. As such, these algorithms have different memory access patterns and might behave differently compared to the underlying algorithm for the continuous variables.

High-throughput processing is usually supported by an appropriate software tool, either classical job schedulers such as SLURM [147] or the workload meta schedulers. These include FireWorks [77], supporting also dynamic workflows, failure-detection routines, and built-in tools and execution modes for running high-throughput computations at large computing centers. Launcher [141] is a utility for performing simple, data parallel, high-throughput computing workflows on clusters, massively parallel processor systems, workgroups of computers, and personal machines. Greasy [24] meta scheduler is used to manage high-throughput simulations and to simplify the execution of embarrassingly parallel simulations in any environment. It was primarily designed to run serial applications. Greasy is used in this work due to its architectural simplicity, ease of use, and the fact that it is a tool already used at multiple supercomputing centers.

1.1.3 Relevant Literature

Massively parallel simulations that exploit modern multi-core central processing units (CPUs) introduce pressure on various subsystems, particularly main memory. Existing literature on efficient utilization of the available computational resources and avoidance of the bottlenecks is very sparse and scattered across various scientific domains. Sources of bottlenecks for parallel protocol processing and high-speed data transfers are identified, e.g., in [65, 142]. The work in [65] studies the impact of different process affinity strategies, considering affinity using cores within the same or different sockets. The authors conclude that affinitization has a significant impact on parallel protocol processing

efficiency, and that the performance bottleneck changes significantly with different affinitization strategies. The focus is put on the communication patterns as opposed to computationally heavy tasks which do not require any interprocess communication, which is the main focus of this work.

The study [142] quantifies cache memory limitations on nonuniform memory access multicore platforms arising during parallel optimization of simulation models governed by partial differential equation (PDE). Many parallel tasks are generated within the PDE model calibration problems which seek to find the model parameters that minimize the error between the PDE model and observed reality. Typically, many simulations are run in parallel, each on its own core. Affinity scheduling strategy for parallel computation is proposed, such that the computational efficiency improves due to improved utilization of the memory hierarchy. It is acknowledged that utilizing excessive parallelism does deteriorate the cache utilization, especially when the processes can migrate across the cores. However, the study does not provide any discussion on how to determine the level of parallelism minimizing the memory bottlenecks on top of enforcing the CPU affinity. Additionally, it doesn't consider the scheduling of the individual jobs on large computational clusters.

1.1.4 Contributions and Organization

This work analyzes high-throughput scheduling techniques and addresses the computational challenges of the massive parallelism associated with stochastic models, uncertainty analysis, or similar applications that rely on a large number of computational tasks which commonly arise in smart grid operations. The main contributions of this paper can be summarized as follows:

- Introduce a technique that mitigates the bottlenecks of embarrassingly parallel simulations by maximizing the utilization of available computational resources and thus reducing the processing time.
- Investigate the proposed technique on various mathematical programs including market based ED, OPF, and UC and experimentally validate the predictions.
- Perform the benchmarks using the ED models of the continental Europe and OPF models of the Swiss transmission network.

This work is based on a previous study of LP problems [80], extending the proposed concepts to additional problem types typically encountered in power grid analysis.

1.2 Power Grid Models

Energy markets and operations of the power grid devices are modeled on different levels of abstraction, capturing different aspects of the underlying physical equipment. Some of the most commonly used mathematical models are (i) an ED problem modeled as LP, (ii) direct current (DC) OPF formulated as QP, and (iii) the UC considering also the discrete aspects of the problem modeled as MIP.

1.2.1 Zone-Based Market Model

The European electricity market is based on bidding zones, which are modeled as one node. Lossless ED considers a problem where the objective is to find the set of generator dispatch points $\mathbf{p}^{\mathrm{G}}$ that minimize the total cost of meeting a specified demand $\mathbf{p}^{\mathrm{D}}$, without modeling any network infrastructure. The problem consists of several zones, where each zone contains several generators $\boldsymbol{G}$ and energy that can be imported or exported. Similarly, the problem is defined over a multiperiod time horizon $\boldsymbol{T}$. The LP model in this work represents a simplified UC problem formulated as a continuous problem in order to have certain guarantees about the convergence and optimality of the solution, as well as reducing the computation time. The LP model reads

$$\underset{\mathbf{p}^{\mathrm{G}}}{\text{minimize}} \quad \sum_{t\in\boldsymbol{T}}\sum_{g\in\boldsymbol{G}} \boldsymbol{c}^{f}_{g,t}\mathbf{p}^{\mathrm{G}}_{g,t} \tag{1.1a}$$

$$\text{subject to } \forall t \in \{1,2,\ldots,N\}:$$

$$\mathbf{p}^{\mathrm{gen}}_{g,t} = \mathbf{p}^{\mathrm{gen}}_{g,t-1} + \mathbf{p}^{\mathrm{G,on}}_{g,t} - \mathbf{p}^{\mathrm{G,off}}_{g,t}, \tag{1.1b}$$

$$\sum_{g\in\boldsymbol{G}} \mathbf{p}^{\mathrm{gen}}{}_{g,t} + \mathbf{n}^{v}_{t} + \mathbf{p}^{\mathrm{Sd}}_{t} = \mathbf{p}^{\mathrm{D}}_{t} + \mathbf{p}^{\mathrm{Sc}}_{t}, \tag{1.1c}$$

$$\mathbf{p}^{\mathrm{G}}_{\min} \le \mathbf{p}^{\mathrm{G}}_{t} \le \mathbf{p}^{\mathrm{G}}_{\max}, \tag{1.1d}$$

$$\boldsymbol{\epsilon}^{\min}_{\mathrm{S}} \le \boldsymbol{\epsilon}_{t} \le \boldsymbol{\epsilon}^{\max}_{\mathrm{S}}, \tag{1.1e}$$

$$\mathbf{p}^{\mathrm{G,off}}_{g,t} \le \boldsymbol{\Delta}^{\mathrm{lo}}, \quad \mathbf{p}^{\mathrm{G,on}}_{g,t} \le \boldsymbol{\Delta}^{\mathrm{up}}. \tag{1.1f}$$

The objective (1.1a) is to minimize the energy cost, where the cost coefficients $\boldsymbol{c}^{f}$ represent marginal cost and approximate start-up costs of each conventional generation unit $g \in \boldsymbol{G}$. The power injection variables $\mathbf{p}^{\mathrm{G}} = \left[\mathbf{p}^{\mathrm{gen}}, \mathbf{p}^{\mathrm{S}}\right]$ consists not only of the conventional generator outputs $\mathbf{p}^{\mathrm{gen}}$ but also includes the injections $\mathbf{p}^{\mathrm{S}}$ incurred by the storage devices. The conventional power output is represented recursively with respect to the previous time instance and the power increment $\mathbf{p}^{\mathrm{G,on}}_{g,t}$ and decrement $\mathbf{p}^{\mathrm{G,off}}_{g,t}$ in the current time period, as expressed in (1.1b).

Considering this representation, minimum up-/down-time and generation ramp constraints can be easily approximated by additional linear constraints. The demand balance constraint (1.1c) states that the sum of all generation components (power plants, net import, and storage discharge) should be equal to the sum of all load components (demand and storage charging) for all time instances $t \in \boldsymbol{T}$. The net import $\mathbf{n}_t^y$ is simply a sum of the power imports and exports for the given zone of interest. Additional constraints are imposed for the links between the zones, such as maximum capacity or flow-based constraints [85].

Energy storage devices are modeled using charging and discharging efficiencies and technical limitations of the state of charge, similar to the model in [84, 23, 98]. N_S energy storage units are considered, where the vector of the storage power injections consists of discharging and charging injections,

$$\mathbf{p}^\mathrm{S} = \left[\mathbf{p}_1^\mathrm{Sd}, \ldots, \mathbf{p}_{N_\mathrm{S}}^\mathrm{Sd}, \mathbf{p}_1^\mathrm{Sc}, \ldots, \mathbf{p}_{N_\mathrm{S}}^\mathrm{Sc}\right]. \tag{1.2}$$

The evolution of the state of charge levels $\boldsymbol{\epsilon}_t \in \{R\}^{N_\mathrm{S}}$ follows the update equation

$$\boldsymbol{\epsilon}_t = \boldsymbol{\epsilon}_{t-1} + \mathbf{B}^\mathrm{S}\, \mathbf{p}^{\mathrm{S},t} \quad t = 1, \ldots, N, \tag{1.3}$$

and introduces a coupling between the individual time periods. The energy level in each period needs to honor the storage capacity, as expressed by the constraint (1.1e). The initial storage level is denoted $\boldsymbol{\epsilon}_0$ and the constant matrix $\mathbf{B}^\mathrm{S} \in \{R\}^{N_\mathrm{S} \times 2N_\mathrm{S}}$ models discharging and charging efficiencies of the storage devices,

$$\mathbf{B}^\mathrm{S} = -\delta t \begin{pmatrix} \eta_{\mathrm{d},1}^{-1} & & & \eta_{\mathrm{c},1} & & \\ & \ddots & & & \ddots & \\ & & \eta_{\mathrm{d},N_\mathrm{S}}^{-1} & & & \eta_{\mathrm{c},N_\mathrm{S}} \end{pmatrix} \tag{1.4}$$

with the discharging and charging efficiencies $\eta_{\mathrm{d},i}$ and $\eta_{\mathrm{c},i}$, $i = 1, 2, \ldots, N_\mathrm{S}$.

1.2.2 Optimal Power Flow Model

An extension of the ED, considering also the transmission network and DC power flow equations $\boldsymbol{c}_\varepsilon$ as a function of bus voltage angle variables $\boldsymbol{\theta}$, along with limits on the branch power flows $\boldsymbol{c}_I$, becomes the DC OPF problem. The DC OPF is

formulated as

$$\underset{\boldsymbol{\theta},\, \mathbf{p}^{\mathrm{G}}}{\text{minimize}} \sum_{t\in\boldsymbol{T}}\sum_{g\in\boldsymbol{G}} f_g(\mathbf{p}^{G}_{g,t}) \tag{1.5a}$$

$$\text{subject to } \forall t \in \{1,2,\ldots,N\}:$$

$$\boldsymbol{c}^t_{\varepsilon}(\boldsymbol{\theta}_t, \mathbf{p}^{\mathrm{G}}_{g,t}) = \mathbf{0}, \tag{1.5b}$$

$$\boldsymbol{c}^t_{I}(\boldsymbol{\theta}_t) \leq \mathbf{p}^{\mathrm{S}^{\max}_{\mathrm{L}}}, \tag{1.5c}$$

$$\mathbf{p}^{\mathrm{G}}_{\min} \leq \mathbf{p}^{\mathrm{G}}_{g,t} \leq \mathbf{p}^{\mathrm{G}}_{\max}, \tag{1.5d}$$

$$\boldsymbol{\epsilon}^{\min}_{\mathrm{S}} \leq \boldsymbol{\epsilon}_t \leq \boldsymbol{\epsilon}^{\max}_{\mathrm{S}}, \tag{1.5e}$$

$$-\boldsymbol{\Delta}^{\mathrm{lo}} \leq \mathbf{p}^{\mathrm{G}}_{g,t} - \mathbf{p}^{\mathrm{G}}_{g,t-1} \leq \boldsymbol{\Delta}^{\mathrm{up}}. \tag{1.5f}$$

The objective function f_g is a quadratic cost defined for each generation unit $g \in \boldsymbol{G}$. Other cost components might also include the wear and tear of load-following ramping and value of the initial and expected leftover stored energy in the storage devices. At each network bus, the external power injections must equal the injections from the connected generators, storages, and load components, resulting in the power balance constraint (1.5b)

$$\boldsymbol{c}^t_{\varepsilon} := C^{\mathrm{G}}\mathbf{p}^{\mathrm{gen}}_t + C^{\mathrm{S}}\mathbf{p}^{\mathrm{S}}_t - \mathbf{p}^{\mathrm{D}}_t - \mathbf{p}^{\mathrm{B}}_t(\boldsymbol{\theta}_t), \tag{1.6}$$

where $C^{\mathrm{G}}, C^{\mathrm{S}}$ are the generator and storage connectivity matrices, respectively. The power flow in the transmission lines is limited, as expressed by the constraint (1.5c). The intertemporal coupling is introduced by energy storage devices (1.5e) and generator ramp limits (1.5f). Additional modeling aspects are described in more detail in [98, 152, 151].

1.2.3 Unit Commitment

The LP and QP problems in the previous sections have been restricted to continuous optimization variables. The real-life decision problems also consist of discrete UC decisions, modeled by integral variables. The problems include additional startup and shutdown costs associated with changes in on-line status from a prior commitment state. In multiperiod problems, these states are coupled through time, not only by the startup and shutdown costs, but also by minimum up and down time constraints.

The MIP problem formulation is an extension of the problem from section 1.2.2. Additional sets of binary variables $\boldsymbol{u}, \mathbf{v}, \boldsymbol{w} \in \{0,1\}$ are introduced, where $\boldsymbol{u}_{g,t}$ represents the on-line status of the generation unit g in time period t, while the

binary startup and shutdown states are represented by $\mathbf{v}_{g,t}$ and $\boldsymbol{w}_{g,t}$ variables, respectively.

The constraints are either extended by the new binary variables, e.g., the injection limits (1.5d) are replaced by

$$\boldsymbol{u}_{g,t}\mathbf{p}^{\mathrm{G}}_{\min} \leq \mathbf{p}^{\mathrm{G}}_{g,t} \leq \boldsymbol{u}_{g,t}\mathbf{p}^{\mathrm{G}}_{\max}, \tag{1.7}$$

or the new constraints are added, such as the minimum up and down times of the dispatchable units

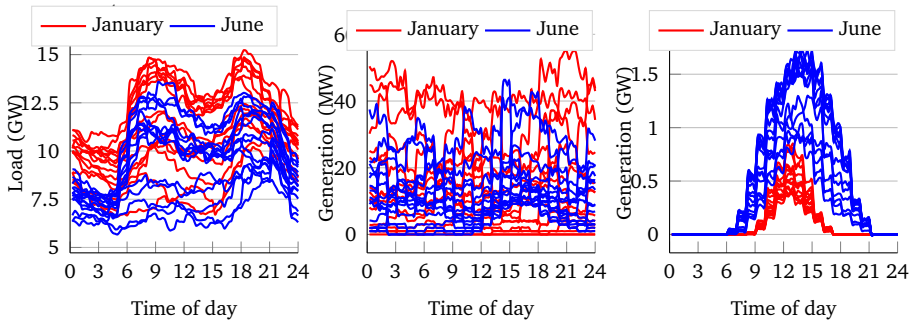
$$\sum_{n=t-\tau^u+1} \mathbf{v}_{g,n} \leq \boldsymbol{u}_{g,t}, \tag{1.8}$$

$$\sum_{n=t-\tau^d+1} \boldsymbol{w}_{g,n} \leq 1-\boldsymbol{u}_{g,t}, \tag{1.9}$$

and a set of the constraints modeling startup and shutdown events

$$\boldsymbol{u}_{g,t} - \boldsymbol{u}_{g,t-1} = \mathbf{v}_{g,t} - \boldsymbol{w}_{g,t}. \tag{1.10}$$

The full UC model formulation is available in the MOST framework [98, 152, 151].



(a) *Load profiles.* ***(b)*** *Wind energy profiles.* ***(c)*** *Solar energy profiles.*

Figure 1.1: *Daily profiles of the Swiss grid benchmark example for selected months [51].*

1.2.4 Swiss Grid Benchmark Model

For the purpose of benchmarking the models introduced in the previous sections, two power models were set up. The LP ED problem was run using the proprietary model of the continental Europe, while the QP and MIP problems were applied

Figure 1.2: *Topology of the Swiss grid with the external nodes in the neighboring countries.*

to the model of the Swiss transmission grid introduced next. The topology of the grid, including the external nodes abroad, is illustrated in Fig. 1.2, consisting of 231 nodes and 439 transmission lines. The external nodes are used in order to model the imports and exports of the power. The load is evenly distributed across the nodes inside Switzerland up to a small random perturbation up to 100 MW with the net zero sum. The load data are 15 minute samples collected by the national transmission system operator SwissGrid [129]. Fig. 1.1a illustrates the overall load during the first 15 days of January and June of 2020. The grid model also includes RES, namely, wind and solar energy, with the historical data shown in Figs. 1.1b and 1.1c. The overall RES input is evenly distributed across the selected nodes in the network.

On top of the RES, the Swiss grid example also contains energy storage devices. For all LP models the length of the time period δt is set to 1 hour, while in the QP and MIP simulations, the length of the time period δt is set to 15 minutes. The energy storage devices are located at the first N_S buses sorted according to the largest positive active load demand specified in the case file. The storage size $\epsilon_S^{\max}$ is chosen to contain up to 10 MWh. The initial state of charge is 70%, which represents $\epsilon_0 = 0.7\epsilon_S^{\max}$. The storage device power ratings are limited to

allow a complete discharging and charging within three hours and two hours, respectively. Therefore, $\mathbf{p}^{\mathrm{Sd,max}} = \frac{1}{3}\boldsymbol{\epsilon}_{\mathrm{S}}^{\max}$ and $\mathbf{p}^{\mathrm{Sc,min}} = -\frac{1}{2}\boldsymbol{\epsilon}_{\mathrm{S}}^{\max}$. All storage device discharging and charging efficiencies are chosen as $\eta_{\mathrm{d}} = 0.95$ and $\eta_{\mathrm{c}} = 0.93$.

1.3 Parallelism Treatment

The stochastic scenarios representing different market developments are processed by running individual LP, QP, and MIP model instances in parallel. Modern many-core CPUs allow multiple solves to be run simultaneously on a given compute node, however, this imposes greater congestion on the node's memory controller as the available bandwidth is shared across cores. Greater congestion in turn adversely impacts solution times.

Considering the increase in runtime of individual solves as the level of parallelism increases, it might be beneficial to reduce the amount of parallelism in favor of reducing the memory bottleneck. It is not obvious, however, how much the parallelism should be reduced in order to achieve the optimal hardware utilization. An empirical procedure proposed in this section may be used to determine the optimal level of parallelism. By following the procedure prior to the execution of subproblems on a given architecture, power grid practitioners can improve utilization of computational resources by avoiding the memory bottlenecks inherent in modern many-core CPUs. Consequently, significantly faster execution can thus be achieved for models comprising many independent subproblems.

In order to determine the optimal level of parallelism for a given model, it is important to find the average run time t_{avg}^{c} at each level of parallelism c. The model of interest is simply run multiple times, each time with a different number of instances running simultaneously, utilizing a different number of cores of the multicore CPU on a given compute node, as demonstrated in Fig. 1.3.

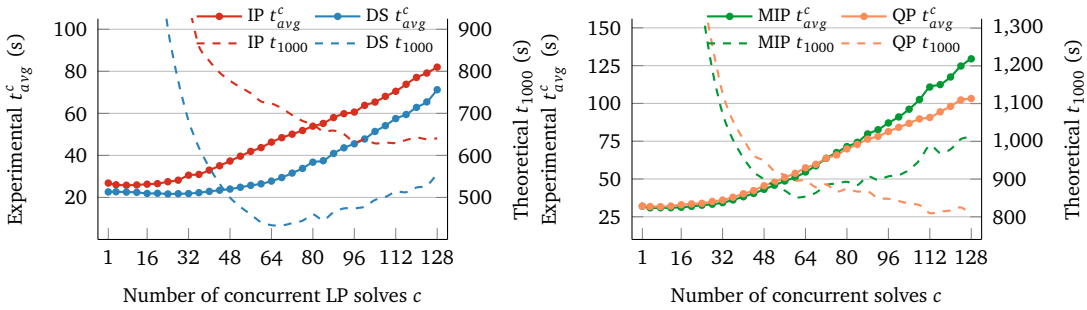
The performance of two different solution algorithms (DS and IP) on instances of the LP model are shown in Fig. 1.3a. Given that the DS algorithm outperforms the IP algorithm across the entire range of solve concurrency, the choice between algorithms clearly favors the DS algorithm. This relationship between the number of concurrent solves and t_{avg}^{c} carries information not only about which algorithm will solve the problem faster, but also about the optimal level of parallelism with which to execute the given solution method. This difference can be mostly attributed to the difference in dynamic random access memory (DRAM) read volume. With $c = 128$ LPs running concurrently on the compute node, the IP executions will require a combined 7.2 TB of data from

memory, while the DS executions will require only 1.7 TB. A similar measure is shown in Fig. 1.3b for the MIP and QP models.

The optimal level of parallelism, i.e. the number of concurrent model solves c, can be determined by finding the minimum expected processing time t_n of n model solves with respect to c. An approximation of this measure is given by

$$t_n \approx \frac{n}{c} t^c_{avg}. \tag{1.11}$$

The expected processing time t_n represents a lower bound for processing n model solves since it assumes no scheduling overhead or idle CPU time. However, since it depends on the experimental quantity t^c_{avg}, its numerical value bears some inherent error.



(a) *LP model with IP and DS algorithms* ***(b)*** *MIP and QP models*

Figure 1.3: *Average runtime t^c_{avg} as a function of c and expected processing time t_{1000} for a batch of $n = 1000$ problems.*

The theoretical processing times t_{1000} for a batch of $n = 1000$ jobs are shown in Fig. 1.3a. The batch size n was set such that it reflects a realistic batch size encountered in practice. The point at which the minimum t_n is attained is relatively unaffected by changing batch size n except for very small batch sizes ($n < 320$) which do not arise in practice.

A substantial difference in the behavior of the algorithms can be observed with respect to parallelism, exhibiting two modes of model behavior. While the overall processing time t_{1000} using the IP algorithm reaches a minimum by exploiting the maximum level of parallelism, that is $c = 128$, with the DS algorithm the models are expected to be processed in the most efficient way by decreasing the level of parallelism to $c = 64$, utilizing only half of the available cores. A similar pattern is observed for the MIP and QP models in Fig. 1.3b. While the QP model can efficiently utilize the maximum level of parallelism, the MIP model's performance improves by reducing the parallelism to $c = 64$.

Equation (1.11) is an approximation of t_n, since it implies that partial batches, i.e., batches of solves that are less than the chosen level of parallelism, c, will be executed at a fraction of t^c_{avg} corresponding to the batch fraction. This would imply that a batch of 1 job, with $c = 128$, would be processed at $\frac{1}{128} * t^{128}_{avg}$ which is not the case. To eliminate this effect an alternative estimation of t_n is proposed:

$$t_n \approx \left\lceil \frac{n}{c} \right\rceil t^c_{avg}. \tag{1.12}$$

The ceiling function $\lceil \cdot \rceil$ implies that the compute node processes the jobs in batches of size c, and that the last batch must be processed at a cost of t^c_{avg}, irrespective of how many solves are remaining. This however represents a simplification on two accounts: (i) first it implicitly assumes that solve time is deterministic, so that each process in each batch of size c starts and ends at the same time. In reality, the solve time is a random variable, with a variance that increases as c increases. This suggests that as c increases, some processes will finish before others, and the cores will immediately start on a new job in the queue. (ii) Second, there is the observation that t^c_{avg} decreases as c decreases. Thus, when there are no remaining jobs in the queue, the effective c will rapidly decrease as it approaches 1, causing the remaining processes to speed up until all jobs finish. Given this, (1.11) gives a reasonable approximation for t_n from a theoretical perspective. The comparison of both predictions are evaluated and compared with actual measurements in the following section.

1.4 Solution Strategy

The resulting LP models are solved by both the DS and IP algorithms, while the QP and MIP models are solved by the IP algorithm. Historically, the DS was considered superior but this proposition has been challenged by advancements in IP methods, which outperform the DS, especially for large-scale problems. The computational complexity of the DS algorithm lies in the combinatorial nature of the search space defined by vertices of the feasible region, which grows very quickly for large LP problems. On the other hand, the computational bottleneck of the IP algorithm is the solution of a large sparse linear system in each IP iteration, which can be effectively mitigated by efficient direct sparse linear algebra routines [20, 84]. When it comes to memory requirements, DS solves asymmetric linear systems of the size of the basis, which is much smaller compared to linear systems in the IP algorithms. Additionally, only a single column changes in every iteration, thus updating the factors is usually done rather than refac-

Table 1.1: *Computational requirements of forecasting electricity prices on a typical day.*

		Short-term forecasts	Long-term forecasts
a)	Forecast horizon (days)	45	700
b)	# of runs per day	15	1
c)	# of weather scenarios	52	45
d)	# of fuel & econ shocks	25	25
	# of optimizations per day*	68 850	66 500
	CPU time per optimization**	164	164
	CPU hours per day	3 137	3 029

* $a \times b \times (c + 2d)$

** *average time in seconds using the DS algorithm with the naive dispatch strategy*

toring the whole matrix, which is recomputed only occasionally for numerical stability reasons. In the IP algorithm, the factorization is computed in every iteration and storing the factors requires a significant amount of memory [78]. The performance and memory requirements of both algorithms are compared in section 1.5.

Additional challenges lie in processing the large number of scenarios associated with the highly dynamic nature of electricity markets. The LP models must be solved for a large number of equiprobable scenarios in order to get a reasonable estimation of the electricity prices distribution. This challenge is compounded by the fact that expectations of input variables are always changing. In the real-world trading environment, this means optimization of the entire problem set is done 15 times per day with the latest available input values. Table 1.1 shows the number of LP solves required per day in order to achieve acceptable prediction accuracy, which is 135 350 LPs at a cost of 6 166 hours of CPU time.

Given the large quantity of optimizations that must be completed on a continual basis, effective parallelization strategies are critical. Parallelization on the level of individual LPs is not considered, given its relatively small size. Instead, parallelization across the set of LPs offers far greater benefit. Thus, each LP solve is executed serially using a single CPU core. The main objectives of an effective computational strategy are optimizing the data pipeline such that the required data remain as close to the CPU registers as possible, and ensuring that when a core finishes processing its LP, there is another job assigned to it with minimal delay.

1.4.1 SLURM Workload Manager

SLURM Workload Manager is an open-source Linux utility that provides access to available computational resources for some duration of time required to perform computation in the context of heterogeneous multiuser, multinode clusters. SLURM is designed for scheduling massively parallel jobs, which usually take significant time to complete. If the program running time is small, on the order of less than one minute, and the number of scheduled tasks is very large, the SLURM scheduler will incur noticeable overhead. Such tasks should not be submitted as individual allocations, but rather packed into a single allocation containing multiple job steps. The disadvantage of this strategy is that it becomes difficult to keep all cores saturated with work when individual job steps finish. This difficulty stems from inadequate tools available in shell scripts and SLURM to detect job step completion, idle resources within an allocation, or specific dependencies between the individual job steps.

1.4.2 Greasy Meta Scheduler

Greasy is an open-source meta scheduler that works on top of SLURM to maximize resource utilization and minimize accounting overhead in embarrassingly parallel applications. Greasy is launched with a list of tasks to run, which are executed using the resources within a SLURM allocation. For each task, Greasy dispatches a job step if resources are available, otherwise it forms a queue and dispatches tasks as soon as resources become available. Additionally, users can control the compute resource utilization by adjusting the number of "workers" to adapt the scheduler to the character of various applications, whether memory or compute bound. A worker is an abstraction that Greasy uses to control the execution flow of job steps within the task list. Conceptually, a worker takes a task off the queue and runs it on the compute node such that each worker is always busy. The user may specify the number of workers in order to control how many processes will run concurrently on the allocated resources. As detailed in section 1.5, undersubscribing nodes and executing on maximally scattered cores can substantially ease bottlenecks in memory bound applications.

The vanilla Greasy implementation dispatches job steps with minimal SLURM specifications, leveraging the process control of either SLURM or the underlying Unix kernel. The schedulers on the level of the operating system may shift processes to different cores during their lifetimes. This can improve throughput for CPU-bound applications, because while the process is waiting for input/output it gets evicted from the core so that another process can use the otherwise lost

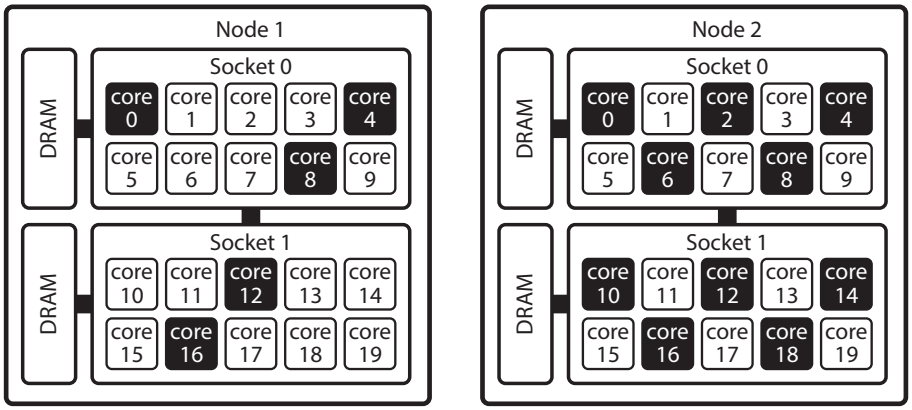


Figure 1.4: *Greasy CPU binding strategy for maximal scattering, Node 1 uses 5 workers while Node 2 uses 10.*

CPU cycles. Once the process is ready to continue, it is assigned to the next available core. For memory bound applications, however, this has dire performance consequences because data in cache memory is completely lost when a process is assigned to a different core. Scheduling processes such that they are executed on the same processor during their lifetime can dramatically improve performance by reducing the number of time-consuming cache misses.

We have extended Greasy such that it provides CPU affinity control [2], a feature that is not available in the original source code. An additional benefit of CPU affinity is that it also allows control over the load balance between the two CPU sockets. To achieve load balance, workers are spread out across maximally dispersed CPU cores based on CPU number as shown in Fig. 1.4.

1.5 Numerical Experiments

In this section, the computational nature of the power grid models is analyzed, focusing on the hardware-software interaction resulting from the application of optimization methods to solve the models. Multiple aspects of the solution process pipeline are analyzed, including (i) a choice of the solution algorithm based on its performance and impact on memory resources, and (ii) scheduling techniques responsible for allocation of computational resources to the individual model instances in massively parallel settings. The experiments were carried out on two Linux based compute clusters with different architectures: the 4-node "DXT Cluster," and the 41-node "ICS Cluster[1]."

[1]The DXT Cluster uses nodes with two 64-core AMD EPYC 7702 1.5 GHz CPUs and 640 GB of memory, SLURM version 18.08.8, GREASY version 2.2.2, and MOSEK version 9.2.21. The ICS

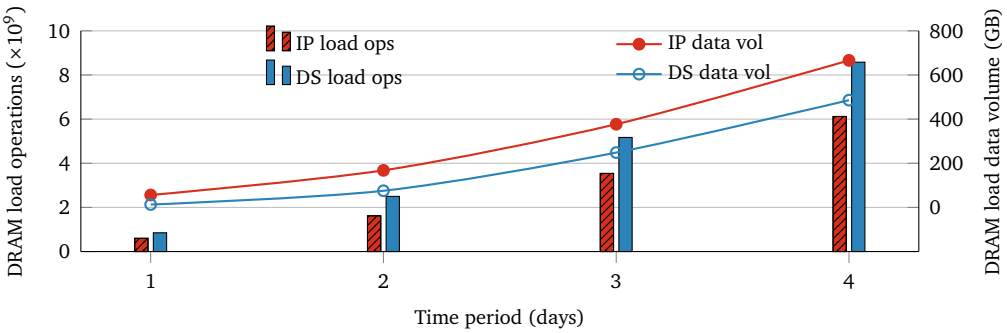


Figure 1.5: *IP and DS memory characteristics.*

In the numerical experiments, the performance of the solution methods is compared using the metric "solve time," which is defined as the wall time of the optimize function of the Mosek solver called for a single LP, QP, or MIP problem, which excludes the problem assembly and setup.

1.5.1 Performance Analysis of the Optimization Algorithms

Performance and memory requirements of the DS and the IP algorithms are studied considering serial execution for LP models of increasing size, as shown in Table 1.2. First, the memory footprint of each algorithm is analyzed, since this imposes the main bottleneck in massively parallel simulations. Memory performance measurements are made using the LIKWID framework [130], accessing the performance counters on the Intel architecture.

From a memory perspective, Table 1.2 shows that the IP algorithm dispatches fewer data load requests (32 billion) compared to the DS algorithm (38 billion), and exhibits slightly better cache locality (1.9% miss rate) compared to DS (2.3%). As seen in Fig. 1.5, however, this economy of data load requests does not translate to economy of data transfers. Although fewer data load requests combined with a better overall cache hit rate results in a significant advantage in the number of DRAM load operations, these operations have far greater average data volume, resulting in a larger DRAM data transfer volume for the IP algorithm. This difference is most significant for the small problem size, where the IP algorithm reads 4.3× more data from DRAM (56 GB) than the DS algorithm (13 GB). These two examples serve as a baseline to show how the memory requirements can change significantly depending on the solution algorithm. Similarly, the memory footprint will be different for each model given differences in

Cluster uses nodes with two 10-core Intel Xeon E5-2650 v3 CPUs and 64 GB of memory, SLURM version 20.02.4, GREASY version 2.2.2, and MOSEK version 9.2.29.

Table 1.2: *Properties of the LP instances and data access characteristics of the solution algorithms.*

Time period (days)	1	2	3	4
# of variables	71 120	113 792	156 464	199 136
# of constraints	104 480	167 376	229 721	292 116
Dual simplex algorithm				
Data load ops ($\times 10^9$)	38	118	246	422
L1 cache hit rate (%)	95.5	95.9	96.0	96.1
Cache miss rate (%)	2.3	2.1	2.1	2.0
Interior point algorithm				
Data load ops ($\times 10^9$)	32	87	190	346
L1 cache hit rate (%)	96.5	96.5	96.6	96.8
Cache miss rate (%)	1.9	1.9	1.9	1.8

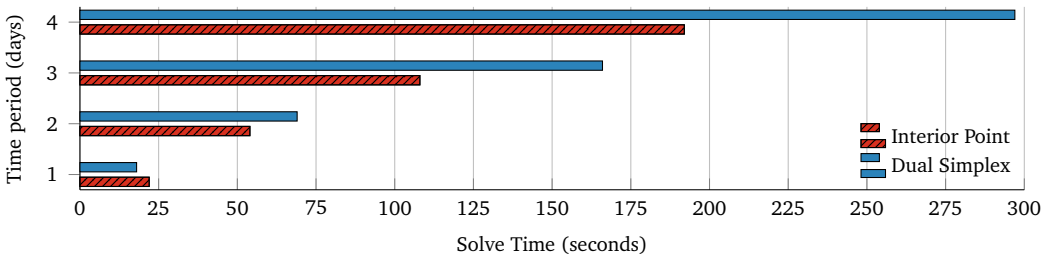


Figure 1.6: *IP and DS scaling for increasing problem size.*

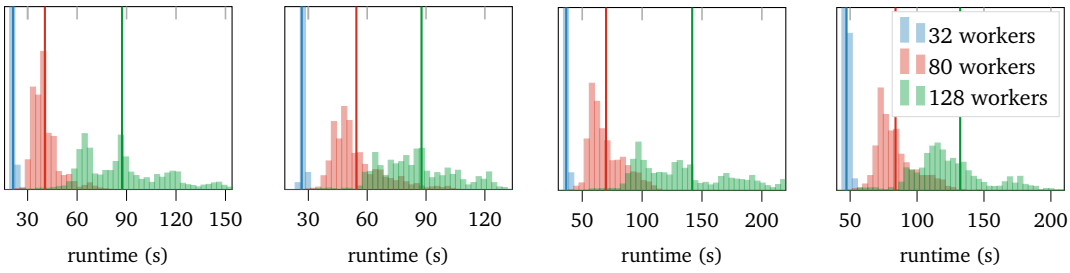
the problem structure.

To establish a baseline, solve times are measured with only a single model instance running in single-core mode on an otherwise idle compute node. Under these conditions, the solve time for a typical single-day LP is on the order of 20 seconds on either the DXT Cluster or the ICS Cluster. With these relatively small problem sizes and nonconcurrent execution, the speed difference between the algorithms is minimal. However, the IP algorithm exhibits superior scaling as the problem size grows in the multiday simulations, as seen in Fig. 1.6. The increase from 1 to 4 days results in a 7.8× increase in constraint matrix size. This causes an 8.7× increase in the solve time using the IP algorithm, compared to 16.5× for the DS algorithm. While the IP algorithm is clearly the better performer from a solve time perspective with large problem sizes, the advantage is less clear with the small problem size. With the introduction of parallelism in the following section, the superior memory performance of the DS algorithm will clearly shift

Table 1.3: *Optimization problem sizes in terms of the number of variables and constraints. For MIP the continuous and integer variables are listed separately.*

Grid model	Variables	Constraints
LP	71 120	104 480
MIP	57 429/3 744	74 181
QP	180 117	213 717

the balance.



(a) *LP model - DS* ***(b)*** *LP model - IP* ***(c)*** *MIP model* ***(d)*** *QP model*

Figure 1.7: *Histograms of individual problem run times with CPU affinity ($n = 3000$), y-axis cropped at 1000. Vertical lines represent mean runtime.*

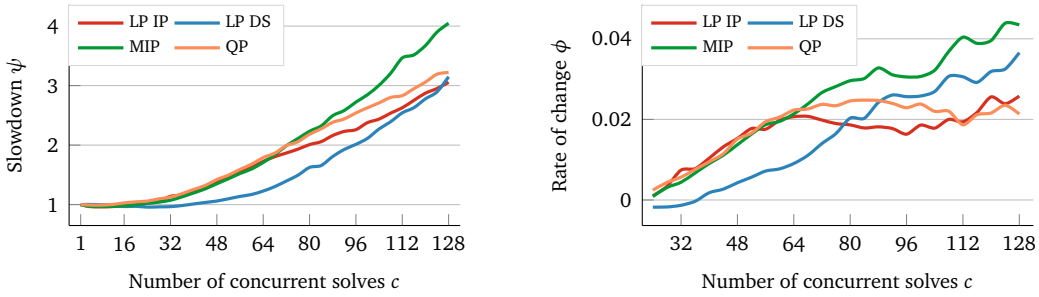
1.5.2 Node-Level Parallelism and Memory Bottleneck

The many-core CPUs of the DXT Cluster are exploited in order to run multiple model solves simultaneously on the given compute node. The optimization problems associated with the power grid models are summarized in Table 1.3. The memory bandwidth on the node becomes becomes saturated and eventually congested as the degree of parallelism increases, since the available bandwidth is shared across the cores. This congestion adversely impacts the solution times, as shown in Fig. 1.7. All models and solution methods experience significant slowdown as the number of solves running concurrently increases (controlled by the number of Greasy workers). The mean slowdown when increasing from 32 to 128 concurrent solves ranges from 2.80× for the QP model, to 4.12× for the LP with the DS algorithm.

Another memory congestion indicator on the node is the runtime variance of individual processes. Figure 1.7 shows how this variance increases substantially as the number of solves running concurrently increases. When increasing from 32 to 128 concurrent solves the standard deviation of runtime increases

from 24.13× for the LP IP, to 46.31× for the MIP. This variance is caused by the memory controller scheduling the processes competing for the bandwidth as it becomes constrained. The substantial increase in variance indicates that memory bandwidth is relatively unconstrained for the lower level of concurrency, and severely constrained for the higher level of concurrency.

1.5.3 Analysis of Parallelism Modes



(a) *Slowdown of t^c_{avg} relative to single worker t^1*

(b) *Slowdown rate of change (average over 6 periods)*

Figure 1.8: *Derived metrics for the average run time t^c_{avg} for all models.*

In order to better analyze different modes of the optimal parallelism determined by a procedure introduced in section 1.3, derived metrics for the measured t^c_{avg} are provided in Fig. 1.8. In Fig. 1.8a the slowdown of t^c_{avg}, ψ is defined as

$$\psi(c) = \frac{t^c_{avg}}{t^1_{avg}}, \tag{1.13}$$

while in Fig. 1.8b, the smoothed rate of change of t^c_{avg}, ϕ is defined as

$$\begin{gathered}\phi(c) = \frac{1}{6}\sum_{j\in K}\frac{t^j_{avg} - t^{j-4}_{avg}}{4}, \\ K = \{c - 4x \mid x \in \{0, \ldots, 5\}\}.\end{gathered} \tag{1.14}$$

The processing time was observed to attain a minimum by either utilizing the full compute node with $c = 128$ or utilizing only half of the cores, $c = 64$, depending on the solution algorithm or the power grid model examined. The slowdown of t^c_{avg} with increasing level of parallelism relative to the single process execution t^1_{avg} is shown in Fig. 1.8a. The two groups are visible by analyzing the

rate of the slowdown, i.e., the slope of the slowdown curves, shown in Fig. 1.8b. The LP with the DS algorithm and the MIP models both have generally increasing slopes throughout the entire range of c, while the slopes of the other two models stabilize around $c = 64$. This suggests that the slope is a more important factor than the total amount of slowdown since the LP with the IP algorithm and the QP both slow down a similar amount as the LP with the DS algorithm. The next subsection demonstrates how the predicted t_n translates to experimental results in massively parallel execution setup.

1.5.4 Massively Parallel Execution

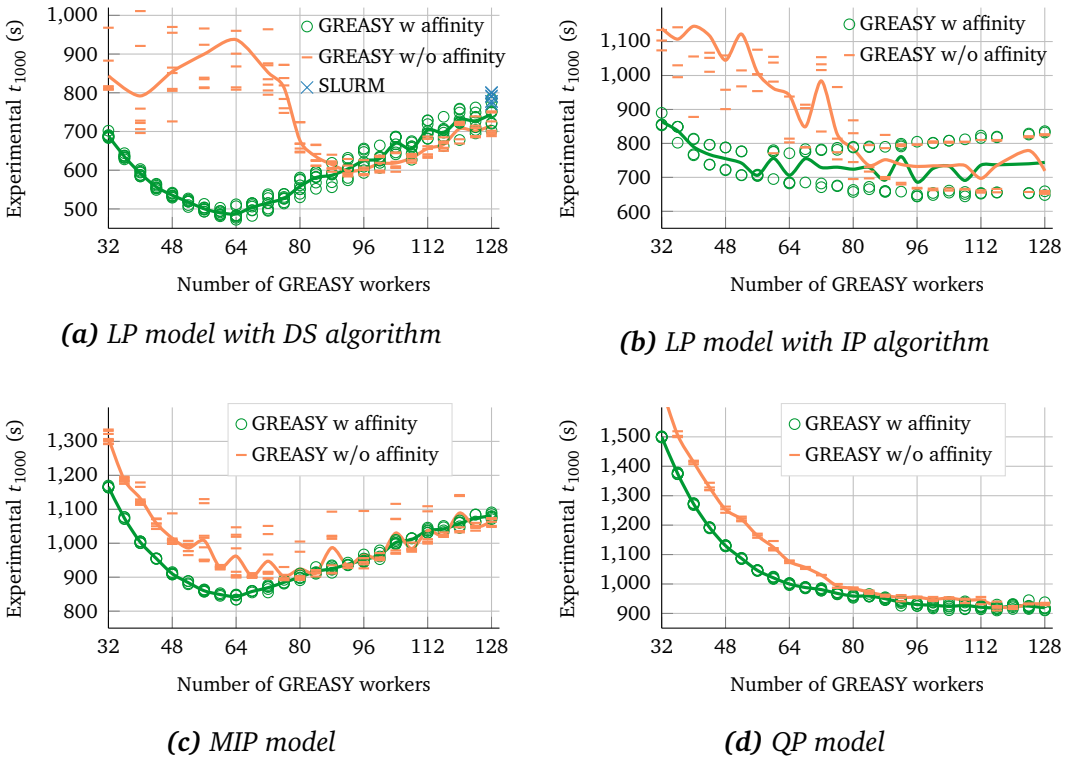
Average runtime of c model solves running concurrently, t^c_{avg}, has been determined experimentally, as shown in section 1.3. In the large-scale experiment considering $n = 1000$ model solves, both the LP model with DS and the MIP model are expected to achieve a minimum runtime t_{1000} at running 64 concurrent solves. On the other hand, the other two models are not expected to reach no decisive minimum before 128 concurrent solves, as shown in Fig. 1.3. Running the LP model with the DS algorithm with 64 concurrent processes results in an average solve time of 26 seconds and an average setup time of 5 seconds. Thus, expected processing time is

$$t^*_{1\,000} = \frac{1\,000}{64}(26+5)\,\mathrm{s} = 484\,\mathrm{s} \tag{1.15}$$

on a single compute node. To achieve such processing time, the assumption that all cores are fully utilized 100% of the time with no delays between individual LP solves running on a given core would have to be met.

Keeping all cores occupied is one responsibility of the workload manager such as SLURM. However, centralized schedulers such as SLURM are fundamentally designed for the traditional paradigm where there are a few large, long-running jobs, rather than ensembles of small, short-running tasks [7]. The strategy of submitting each individual model solve as a job batch to SLURM thus incurs extra overhead as the scheduler identifies and matches jobs to idle resources, accounts for user priority, prepares the environment, creates temporary directories, performs some sanity or health checks, etc. Considering the example with $n = 1000$ for the LP model with the DS algorithm, SLURM needs significantly more time to finish processing all LP solves, compared to the established expected processing time t^*_{1000}. The average processing time across 7 trials for SLURM is $t^{\mathrm{slurm}}_{1000} = 787\,\mathrm{s}$ seconds, as shown in Fig. 1.9a. These experimental results were obtained using a strategy that submits each LP solve to SLURM as a separate job batch, which max-

imizes the administrative overhead mentioned above. Multiple LP solves could be grouped into job batches, however, while such a strategy reduces administrative overhead, it becomes difficult to balance the batches to keep available CPUs fully saturated with jobs. Experimentally it was found that the administrative overhead of SLURM is much less than the idle CPU overhead that such job grouping strategies incur. To eliminate SLURM overhead while minimizing idle time across CPU cores, the meta scheduler Greasy is used.



(a) *LP model with DS algorithm*

(b) *LP model with IP algorithm*

(c) *MIP model*

(d) *QP model*

Figure 1.9: *Performance of GREASY for various numbers of workers with and without CPU affinity control*

1.5.5 Massively Parallel Execution with Meta Scheduling

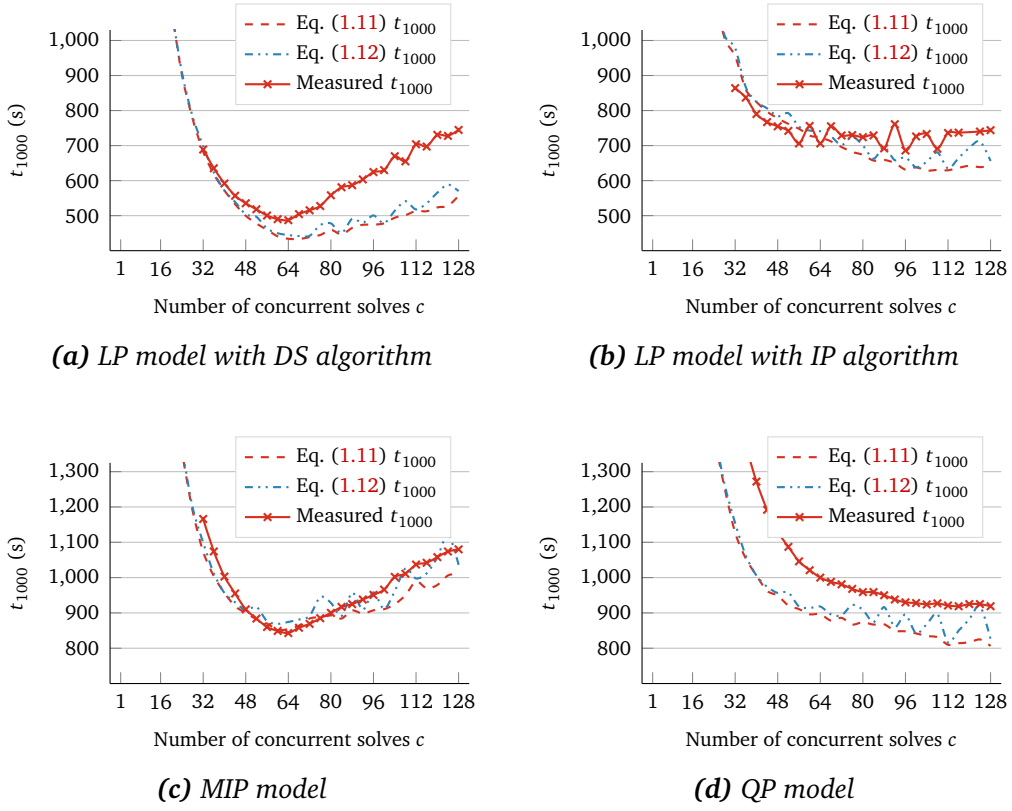
The primary goal of the meta scheduler is to eliminate the overhead incurred by SLURM. Considering a Greasy configuration using all 128 cores (128 workers), the execution of Greasy and SLURM are equivalent from a perspective of the number of solves that are running concurrently and the utilized resources. Figure 1.9a illustrates that for the LP with the DS algorithm Greasy was able

to achieve an average execution time $t_{1000}^{\text{greasy}} = 715\,\text{s}$ without CPU affinity control and $t_{1000}^{\text{greasy}} = 745\,\text{s}$ with CPU affinity control, which represents decrease of 9% and 5%, respectively, compared to t_{1000}^{slurm} on a single compute node of the DXT Cluster. The main benefit from using Greasy for data-intensive applications, however, arises from the ability to control the level of compute resources saturation. Since the memory bottleneck is exacerbated as the number of concurrent jobs increases, dispatching fewer concurrent solves to the compute node should benefit the overall processing time as established by (1.11). To control this parameter, Greasy uses the "worker" abstraction introduced in section 1.4.2. Figure 1.9 illustrates the effect of undersubscribing the DXT Cluster node consisting of 128 cores on a batch of 1 000 model solves. These experimental measurements confirm the predictions of (1.11) that the optimal processing time $t_{1000}^{*} = 484\,\text{s}$ and thus optimal throughput should be achieved at 64 concurrent processes. The experimental measurement $t_{1000} = 487\,\text{s}$ differs from the predicted result by less than 1%, and represents a time reduction of 38% compared to using the plain SLURM strategy, demonstrating how effective the reduction of parallelism is at reducing memory bottlenecks.

For models which are optimally executed with an undersubscribed node, controlling the CPU affinity is essential to achieving the optimal processing time, although the magnitude of the impact depends on the specific model. The comparison of experiments with and without such control is illustrated in Fig. 1.9. For a node employing at least 75% of the available cores, the difference is modest, but a gap of almost a factor of two occurs for the undersubscribed node of the LP DS, especially for the point at which the optimal throughput was established. This is the effect of the Unix kernel scheduler intervening with the affinity of the processes, effectively eliminating the benefit of data locality and thus losing the advantage brought by the cache hierarchy.

1.5.6 Verification of the Optimal Parallelism Degree Prediction

The correspondence between the experimental measurements of t_{1000} using Greasy, and the t_n predicted by (1.11) and (1.12) is discussed in this section. The two equations are compared against the experimental results obtained by running batches of 1 000 jobs using Greasy with CPU affinity. The predicted values, introduced in Fig. 1.3, are qualitatively similar to the experimental values, as can be seen in Fig. 1.10. This demonstrates the fact that the predictions correspond well with experimental results in terms of determining the optimal level of parallelism that needs to be used in order to minimize the impact of memory bottlenecks. In this way, the processing throughput of the computational tasks is



(a) *LP model with DS algorithm*

(b) *LP model with IP algorithm*

(c) *MIP model*

(d) *QP model*

Figure 1.10: *Predicted and experimentally measured processing time t_{1000} of 1 000 LPs.*

maximized.

The gap between the predicted and the measured t_{1000} may be attributed to further memory congestion introduced by the introduction of massive parallelism, as well as Greasy overhead. It can be also seen that the ceiling function in (1.12) introduces some nonlinearities to the predicted t_{1000} that do not appear consistently in the measured t_{1000}. However, the predictions do not change qualitatively using both formulations of the t_n prediction, thus the optimal level of parallelism can be found using either predictor.

1.6 Conclusions

For many time critical applications that consist of a large number of subproblems the usual approach to decrease computation time is to improve the solver. This, however, is often very expensive in terms of development time or license fees.

Significant improvements can be achieved by optimizing the hardware utilization, either by selecting a method which reduces the bottlenecks or by adopting scheduling techniques better suited for the computational nature of the problem at hand. In the real-world trading environment, the careful management of the resources during the execution of embarrassingly parallel LP simulations improved the throughput of computations by 38%. This type of speedup is significant considering the computational demands of 135 350 optimizations per day, reducing daily computation time from 6 166 CPU hours to about 2 280 CPU hours.

This work proposed a simple procedure that can be used to determine the optimal level of node-level parallelism by power grid practitioners. The proposed procedure for parallel processing of a large number of simulations is based on a simple benchmark of empirical measurements. As such, it doesn't require any preliminary knowledge about the hardware architecture (e.g., the memory hierarchy properties) or characteristics of the solution algorithm (such as its memory access patterns). The limitation of this approach is such that the conclusions from one hardware architecture are not, in general, transferable to other architectures. The same applies to the particular problem at hand, where the conclusions are not transferable across the different problem types. Also the solution method needs to be considered, where various solution strategies might behave differently. Additionally, it is assumed that the problems included in the job pool are homogeneous from the computational perspective, i.e., having similar memory and time complexity, as well as the data access patterns. On top of this, the jobs are assumed to be independent, i.e., no interaction occurs between them.

Future work directions could focus on applying the proposed parallel computation schemes to additional problem instances in the smart grid analysis, e.g., nonconvex nonlinear AC OPF. The methodology is production ready, and can be integrated in the tools such as GREASY, or other pilot job mechanisms as a preprocessing or analysis step suggesting to the user runtime parameters aiming to improve the hardware utilization. Additionally, before applying the procedure to the parallel decomposition schemes such as Benders, one should study how does the process synchronization (communication between the processes) impact the prediction.

Chapter 2

Modeling techniques to improve performance and accuracy of power market optimization models

2.1 Introduction

The content detailed in this chapter served as the subject of a 6-month full-time internship that I completed at DXT Commodities from January to June 2022.[1] The overarching goal of the internship was to improve the performance and accuracy of the power forecasting models employed by the DXT power trading desk. For performance, the critical metrics for DXT are both simulation throughput and latency, as it is imperative to reduce the time between receiving new weather forecasts and having high fidelity model results. For accuracy, the critical metric is the prediction error of the model on a back-fit test which simulates how the model would have performed in the past given the data available at the time. Specifically the objectives were as follows:

1. **Model Reformulation:** Reformulate the Stack European power market model using high-level algebraic modeling language. Such a model would enable rapid prototyping of new model features and would facilitate updates to accommodate changing market structures. This would also allow

[1] The internship was focused on computational engineering of models whose details are sensitive business information. Given that no work was published from this internship, this chapter will focus mostly on high-level concepts and performance findings of the developments implemented during the internship, while avoiding implementation details that could reveal sensitive information.

for performance improvements by enabling performance testing of individual model components.

2. **High-Speed Solution Methods:** Investigate solution methods for the reformulated model including hot-start optimization, parallel interior point methods, and re-calibration of the dispatch protocol.

3. **Multi-Day Modeling:** Extend the capabilities of the Stack model to be able to tractably handle simulations with extended time periods of up to a week, or 184 periods. The legacy model was a multi-period LP model, but the scaling of both problem setup and model solution limited the ability to run multi-day simulations with more than 40 periods. The advantage of multi-day simulations is that they are able to capture the economic reality of electricity markets which exhibit natural weekly cycles.

In this chapter I will detail the various findings of the extensive model development and deployment, and performance benchmarking that was completed for this project.

2.2 Model Reformulation

The European market model that was implemented by DXT was written natively in Python using the Scipy and Numpy libraries to build the respective sparse matrices for the LP and then pass these to the solver using the Python interface. This modeling setup was limited in its usability and maintainability, as well as scaling for longer simulation periods greater than 4 days. The model reformulation efforts were about designing a more efficient data interface as well as an implementation of the model that would relax the limitations of the existing implementation.

2.2.1 Pyomo Model

The first objective of the was to build out a functional European market model using the Pyomo algebraic modeling language [28, 69]. Pyomo is a python package developed at Sandia National Labs that constructs LP models using notation that is very similar to the common optimization algebraic notation. Pyomo also provides a universal interface to allow for the model to be solved by a number of different optimization software packages. A diagram of the Pyomo structure is shown in Fig. 2.1.

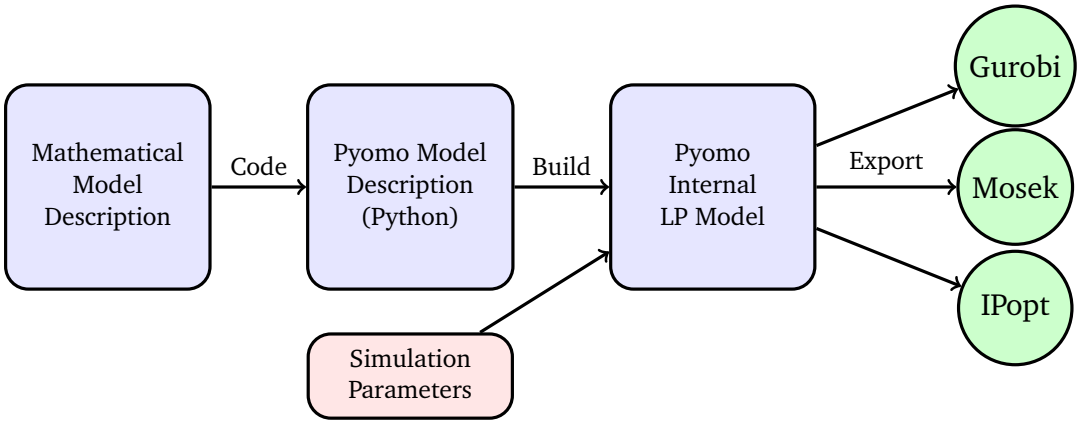


Figure 2.1: Pyomo model structure *- Pyomo simplifies the "Code" step since the model definitions are written in a syntax similar to the mathematical description. Pyomo also automatically completes the "Build" and "Export" steps at runtime, providing interfaces to major optimization packages.*

Table 2.1: *Single-core time performance (seconds) for new models on Intel i9 Coffee Lake 2.3/4.8 GHz*

	Pyomo model	Fusion model
Build Model	7.9	2.9
Export Model	6.7	N/A
Solve Model (Mosek)	9.4	9.0
Total	23.9	11.9

The Pyomo model was built from scratch based on the mathematical definition of the European market model, was fully constructed to the point where it included all constraints in the European market model. The model had excellent performance in terms of solve time, improving upon the current production model by approximately 3×. These gains in solve time, however were largely offset by overhead inherent in the design of Pyomo. The fact that the Pyomo internal LP model is constructed using individual parameter indexes and for-loops makes building the model inherently slow. Additionally, since Pyomo is designed to be independant of the optimization software, an additional step is needed to translate the Pyomo LP representation into a form suitable for the optimization software. The Pyomo model timings for a sample simulation of 8 March 2022 on an Intel i9 laptop computer are shown in Tab. 2.1.

2.2.2 Fusion Model

Given the promising results of the Pyomo model in terms of optimization speed and ease of use, it was decided to develop a model using this modeling paradigm, but with a software package that does not have the same performance limitations as Pyomo. The Mosek Fusion library [12] was identified as a possible solution since it provides an algebraic-like modeling interface, but is entirely matrix-based and builds the models directly into the Mosek internal representation, thereby eliminating the "Export" step.

The Mosek Fusion package is an object-oriented C++ library that provides an interface allowing users to build optimization models using algebraic-like expressions. The Fusion library is designed such that the programmer uses matrix operations, and low-level computations are executed using efficient linear algebra subroutines. The library is designed to provide a compromise between an expressive high-level interface, and fast low-level performance suitable for production optimization models. A diagram of the Fusion structure is shown in Fig. 2.2.

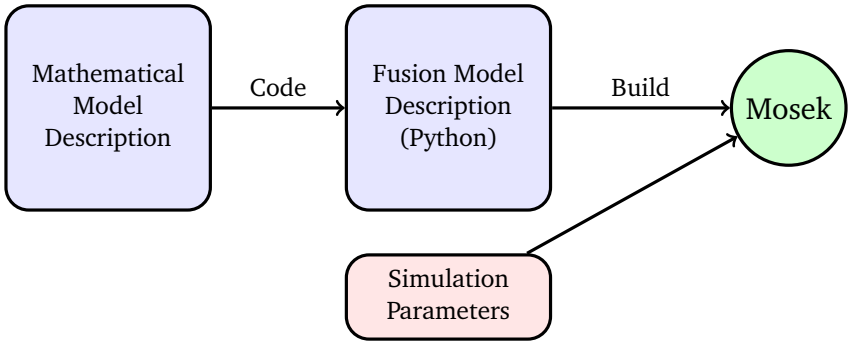


Figure 2.2: Fusion model structure *- Fusion provides an efficient direct interface to the Mosek solver while sacrificing some ease-of-use found in Pyomo.*

The performance of the Fusion model is excellent, showing significant improvements in all steps of the optimization process over both the Pyomo model and the legacy production model. The Fusion model timings for a sample simulation on an Intel i9 laptop computer are shown in Tab. 2.1. In a production setting, run times of individual simulations should be measured in a fully parallel setting to account for node congestion and other factors that are experienced in the production environment. Fig. 2.3 shows the comparison of timings in a large fully parallelized run such as occurs in production. In this scenario, we see an average speedup of 6.5× for the new Fusion model over the legacy production model.

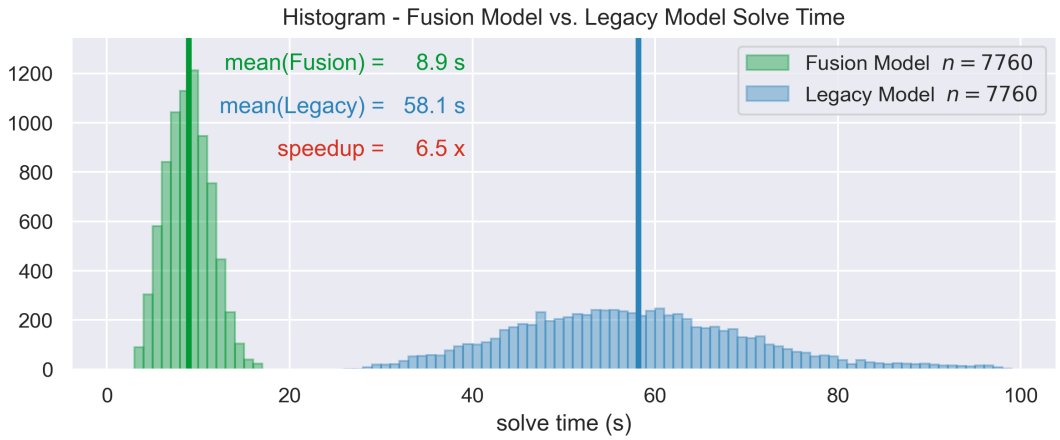


Figure 2.3: *Solve time comparison between new Fusion model and legacy production model in fully parallel production environment of the Stackmaster HPC cluster. New fusion model exhibits an average speedup of 6.5×.*

Another important metric for these models is latency, which is the time from when the job is dispatched until all of the results are in. This metric is affected especially by the right tail simulations since we must wait for the slowest of the simulations to complete. In such a case, the legacy production model had some simulations which took upwards of 200 seconds and so by this metric, we see an order of magnitude improvement.

2.3 High-Speed Solution Methods

Batches of scenarios are frequently processed where the scenarios in the batch are somewhat related to each other, specifically, they have partially overlapping sets of parameter values. An example of this would be a base case scenario, and a scenario where demand is shocked by a some amount. Since demand enters into the model in a single constraint, the demand-generation balance, the two scenarios would have almost identical parameter value sets, and therefore constraint matrices.

Given that the simplex method requires a feasible starting point, and finding such a starting point is non-trivial,[2] some time can be saved by providing directly

[2]We need a basic feasible starting point x and corresponding initial basis $\mathcal{B} \subset \{1, 2, \ldots, n\}$ with $|\mathcal{B}| = m$ where the basis matrix B is nonsingular, $x_{\mathrm{B}} = B^{-1}b \geq 0$, and $x_{\mathrm{N}} = 0$. Finding the initial point and basis may be nontrivial, with equivalent difficulty as solving the linear program [101].

a starting point for the optimization. With scenarios where we have two closely related problems with minimal entries changed in the constraint matrix or in the right-hand side, we expect the optimal solutions to be close to each other geometrically speaking. We can exploit this closeness by solving the base-case scenario to optimality using the Simplex algorithm, then using that solution as an initial feasible point of the next scenario. This is referred to as a "hot-start" in optimization. The basic idea of the hot-start is depicted in Fig. 2.4.

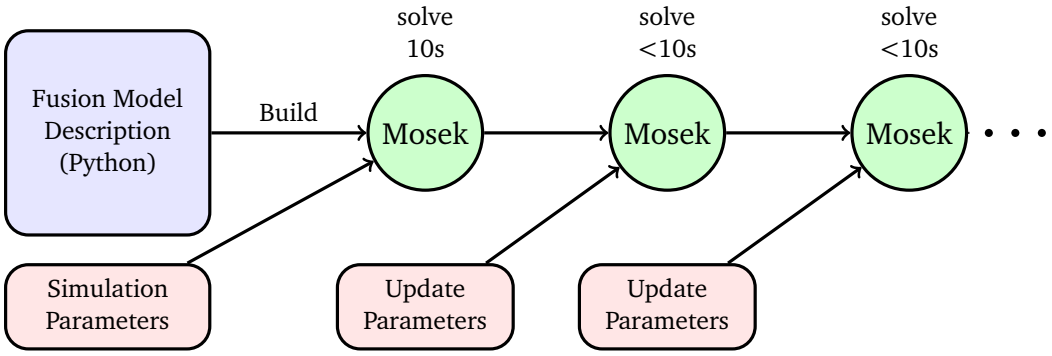


Figure 2.4: Hot-Start Optimization *- Related simulations may exploit the previous Simplex optimal solution to achieve much faster subsequent solve times.*

There are 3 different situations that should be considered when analyzing the potential benefits of the hot-start:

1. Minor update: cases such as a demand shock or a change in fuel prices, where only a small number of parameters change.

2. Moderate update: cases where multiple parameters change, however the number of parameters does not. In such cases, the optimal solutions may be far apart, but the initial guess may still be of value, and the time required to update the parameters is limited.

3. Major update: cases where many parameters change, and the number of parameters also changes. In this case the optimal solutions will be far apart and the time required to update the parameters is significant.

The time performance for each of these three types of updates is detailed in Tab. 2.2. We see that there can be significant benefit of nearly 8× speedup for scenarios if they fall in the minor update group.

Table 2.2: *Single-core hot-start time performance (seconds) on Intel i9 Coffee Lake 2.3/4.8 GHz*

	minor update	moderate update	major update
Build Base Model	3.1	2.9	3.1
Solve Base Model	11.5	11.4	11.9
Update Parameters	0.1	2.4	2.8
Solve Scenario Model	1.8	3.9	9.0
Time Saved per Scenario	12.7	8.0	3.2
Scenario Speedup	7.7×	2.3×	1.3×

Table 2.3: *Single-core multi-day time performance (seconds) of Fusion model on Intel i9 Coffee Lake 2.3/4.8 GHz*

DAYS	1	2	3	4	5	6	7	8	9
# of Constraints ($\times 10^5$)	2.0	3.2	4.3	5.5	6.6	7.7	8.8	9.9	11.0
# of Variables ($\times 10^5$)	1.0	1.7	2.3	2.9	3.5	4.2	4.8	5.4	6.1
Timings (seconds)									
Init Data Container	0.4	0.6	0.7	0.8	1.0	1.3	1.5	1.5	1.8
Build LP	3	5	6	8	10	12	16	18	21
Optimize - IP Parallel	10	17	28	38	49	72	77	85	123
Optimize - IP Serial	12	23	36	49	64	93	97	121	175
Optimize - Dual Simplex	7	25	39	67	85	134	186	249	304
Total time - Best Algo	11	22	35	46	60	85	94	105	146

2.4 Multi-Day Modeling

Multi-day models better capture the economics of power markets, that exhibit weekly cyclicality and seasonality. In addition, results are less dependent upon the starting conditions of the simulation, which carry uncertainty. Multi-day simulations are challenging, since the size of the constraint matrix grows quickly and subsequent operations can quickly become impractically slow if inefficient algorithms are used. This said, since the problem structure is extremely sparse, very large problems should be manageable even on computers with modest memory, provided that the problem is built efficiently.

When using the Fusion library, sparse matrices are built internally by the optimized library designed specifically for this purpose. As such, the Fusion model is

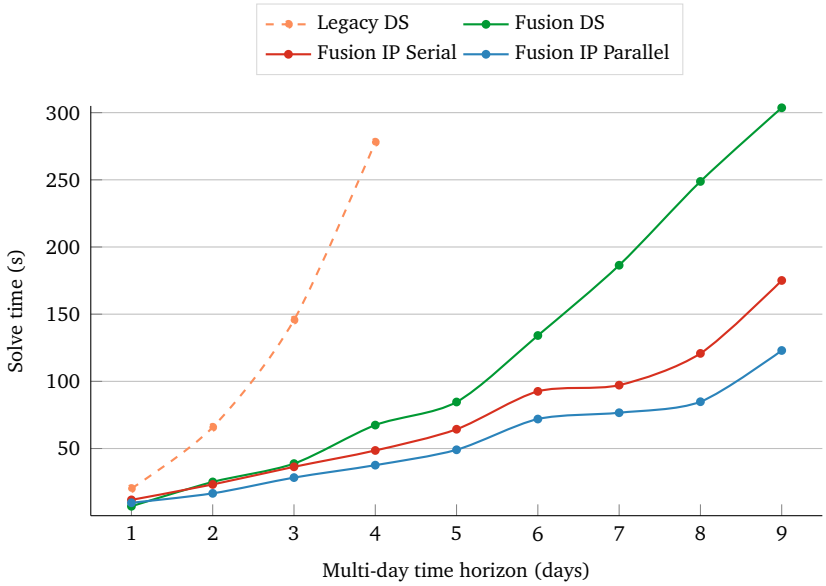


Figure 2.5: Multi-Day Performance *- Legacy production and Fusion models solved with the Dual Simplex and Interior Point solution algorithms in parallel and serial versions.*

scalable for large simulations. The Fusion model has been able to solve problem sizes up to 9 days, which is currently the maximum size of data set that can be prepared by the DXT database. Time performance of the Fusion model for multi-day problems from 1-9 days can be seen in Tab. 2.3. Each of the solve (optimize) time rows is with a different solution algorithm. Either the parallelized Interior Point algorithm, the serial version, or the Dual Simplex algorithm. Graphical comparison between the optimization time of the Fusion model and the legacy production model on multi-day solves can be seen in Fig. 2.5.

Part II

Power Generation Unit Commitment

Forward to Part II: A brief overview of the AC Optimal Power Flow and Unit Commitment Problems

A basic understanding of the AC (alternating current) Optimal Power Flow and Unit Commitment problems is essential to grasp the content of Part II. This forward will provide this background for readers who are unfamiliar with the domain, while providing references for readers who want to delve deeper.

Brief Summary of AC Optimal Power Flow[3]

Electric power systems lend themselves to a graphical representation where the vertices are electrical buses connected by branches that represent transmission lines. Buses are connection points for various types of electrical infrastructure, most notably generators and loads. The fundamental purpose of the power grid is to transfer electric energy from generators to loads, while maintaining all operating parameters within the specified constraints. The AC Optimal Power Flow (ACOPF) problem is an optimization problem tasked with determining these optimal operating parameters of an electric power network under a given set of conditions.

This optimization needs to be frequently completed during the operation of a grid as load and supply conditions evolve, or else in advance using load and supply forecasts in order to aide in planning of grid operation. By determining the optimal operating conditions of the network, ACOPF analysis plays a vital role in ensuring the smooth operation and effective planning of power systems, allowing them to achieve their overarching goal of delivering reliable and affordable power to consumers. ACOPF includes variables and constraints that represent the phenomena present with alternating current power systems such as reactive power and voltage constraints. These unique elements are highly relevant from the grid operator's perspective as they are essential to understanding and planning the operation of a given grid on the engineering level. This level of detail however may not be necessary to understand the economics corresponding to a given load condition. This is why the market model presented in Part I did not model any AC-related phenomena and aggregated demand onto the market zone

[3]This summary is inspired by the excellent introductory article on optimal power flow presented by Frank and Rubennack in *IIE Transactions* in 2016 [57]. For a detailed introduction to the subject that is both quantitative and qualitative, I highly recommend this article.

level while ignoring intra-zone transmission.

Mathematical Formulation [57]. The ACOPF problem is cast as a nonlinear optimization problem, with the primary objective of minimizing the total generation cost while satisfying the power flow equations and system constraints. The mathematical representation includes variables such as real and reactive power generation, voltage magnitudes, and power flow parameters. The specification of the optimization problem can vary significantly according to the network features, chosen representation of voltage and admittance, and the level of detail represented. The classical formulation of the optimization problem from Carpentier [33] and Dommel and Tinney [44] models a set of buses **N** connected by a set of branches **L**, with generators located at buses $\mathbf{G} \subseteq \mathbf{N}$. The generation cost functions vary from generator to generator, but usually take the form of a quadratic function of real power output: $C_i(P_i^G)$. Minimizing the cost across all generators is the basic objective.

$$\begin{aligned}
\min \quad & \sum_{i \in \mathbf{G}} C_i\left(P_i^{\mathrm{G}}\right), && && (2.1)\\
\text{s.t.} \quad & P_i(V,\delta) = P_i^{\mathrm{G}} - P_i^{\mathrm{L}}, && \forall i \in \mathbf{N}, && (2.2)\\
& Q_i(V,\delta) = Q_i^{\mathrm{G}} - Q_i^{\mathrm{L}}, && \forall i \in \mathbf{N}, && (2.3)\\
& P_i^{\mathrm{G,min}} \le P_i^{\mathrm{G}} \le P_i^{\mathrm{G,max}}, && \forall i \in \mathbf{G}, && (2.4)\\
& Q_i^{\mathrm{G,min}} \le Q_i^{\mathrm{G}} \le Q_i^{\mathrm{G,max}}, && \forall i \in \mathbf{G}, && (2.5)\\
& V_i^{\min} \le V_i \le V_i^{\max}, && \forall i \in \mathbf{N}, && (2.6)\\
& \delta_i^{\min} \le \delta_i \le \delta_i^{\max}, && \forall i \in \mathbf{N}. && (2.7)
\end{aligned}$$

Constraints 2.2 and 2.3 are the real and reactive power flow equations in polar form. The remaining constraints represent bounds on the system voltages and powers. In the general form, real and reactive loads are fixed, while the optimization control variables are the real and reactive power output of each generator.

The decision variables of the optimization can be partitioned into a set of control variables u (real and reactive bus power injections),

$$u = \left(P^{\mathrm{G}}_{i:i\in\mathbf{G}}, Q^{\mathrm{G}}_{i:i\in\mathbf{G}}\right),$$

and a set of state variables x (voltage magnitudes and angles) [44, 27],

$$x = (\delta_2, \ldots, \delta_N, V_2, \ldots, V_N).$$

a frequent addition to the above formulation is to add min and max flow constraints onto the branches such that the solution respects the engineered limits of the respective power transmission lines. In following work, such constraints are always considered for the systems.

The power balance constraints associated with alternating current increase the difficulty of optimal power flow analysis given their non-linear nature and coupling of variables. The power balance equation involves squared terms and trigonometric functions due to the presence of real and reactive power magnitudes and phase angle, making the overall OPF problem non-linear. A common relaxation in power system optimization is to thus relax the voltage constraints constraints and eliminate the reactive power and voltage variables. This relaxation results in the DCOPF problem, since this relaxation causes the grid to behave as though it were based on direct current.

Challenges in ACOPF. The optimization of ACOPF problems poses significant computational challenges due to:

- **Non-linearity:** quadratic terms and trigonometric functions in the power balance equations, make the optimization problem inherently non-linear,
- **Convexity:** convexity is often lacking in ACOPF problems, primarily due to the presence of non-convex constraints related to voltage magnitude limits and branch angle differences,
- **Numerical Instability:** numerical challenges arise from solution sensitivity to parameter variations and potential existence of local optima,
- **Binary or Integer Constraints:** binary or integer constraints are often used to represent discrete control variables like switching devices, resulting in a combinatorial search space,
- **Stochasticity:** renewable energy generation and demand forecasts are inherently stochastic,
- **Size and Complexity of Networks:** power networks characterized by many buses, branches, and generators amplify the computational burden significantly given the elevated algorithmic complexity of solution methods.

Solution Methods. The solution methods for the ACOPF problem primarily involve advanced optimization algorithms coupled with robust numerical solvers to address the challenges posed by its non-linearity and complexity. Gradient-based optimization methods, such as interior-point methods and sequential quadratic programming, are widely used for solving ACOPF problems. These algorithms iteratively improve the solution by considering the gradient of the objective function and constraints. Additionally, metaheuristic approaches like genetic algorithms, particle swarm optimization, and simulated annealing are frequently proposed alternatives for handling non-convexities and finding solutions in complex, multimodal solution spaces. Robust numerical solvers play a critical role in enhancing the stability and efficiency of the optimization process. Efficient linear programming solvers, non-linear solvers, and numerical libraries contribute to the accuracy and usability of the optimization results.

Brief Summary of the Unit Commitment problem

The power generation unit commitment (UC) problem focuses on determining the optimal commitment and dispatch schedule for a set of power generation units. Given that grids must be designed to meet peak load conditions that happen only sporadically for brief periods of time, there is necessarily an oversupply of generators available to the network during the majority of normal operating periods. Typically, a power generator has a fixed cost associated with being active and ready to supply a specified amount of power to the grid. These "startup costs" result in a discontinuous cost function for generators, and are usually modeled using binary variables that result in a mixed-integer optimization problem. Naturally, UC decisions have multi-period impacts since starting and shuttering a generator is a significant process in the real world.

Adding the power flow equations to the UC problem results in the security-constrained unit commitment (SCUC). This security constrained formulation ensures that the Unit Commitment produced is actually feasible for the physical power system, not violating voltage and flow limits [57]. This formulation, however, results in a significantly harder optimization problem. As such, a common workflow is to find a relaxed Unit Commitment, and then test it afterwards for feasibility on the security constrained system.

Another aspect of UC is to plan generator dispatch decisions so that the grid is resistant to unplanned outages – referred to as contingencies – which are a stochastic phenomenon. Adding in consideration of contingencies results in a non-linear, non-convex, multi-period, combinatorial, stochastic optimization problem. There are no known algorithms to natively solve such a problem, and

so relaxations are frequently used to solve an approximation of the problem.

Mathematical Formulation. The Generation UC problem is formulated as a mixed-integer non-linear optimization problem, with the primary objective of minimizing the total operational cost while satisfying operational and technical constraints. The cost function thus encompasses binary decision variables for unit commitment and continuous variables for power generation. A very basic representation of the optimization problem can be expressed as follows:

$$\text{Minimize} \quad \sum_{t=1}^{T}\sum_{i=1}^{N}(C_i^{\text{startup}}U_{it} + C_i^{\text{operation}}P_{it}) \tag{2.8}$$

Subject to:

$$\begin{aligned}
&U_{it} \in \{0,1\} \quad \forall i \in \text{generators}, \forall t \in \text{time periods}\\
&P_{it} \leq P_i^{\max} \cdot U_{it} \quad \forall i \in \text{generators}, \forall t \in \text{time periods}\\
&P_i^{\min} \cdot U_{it} \leq P_{it} \quad \forall i \in \text{generators}, \forall t \in \text{time periods}\\
&P_{it} - P_{i(t-1)} \leq R_i \quad \forall i \in \text{generators}, \forall t \in \text{time periods}\\
&\sum_{j=1}^{N} D_{jt} = \sum_{i=1}^{N'} P_{it} \quad \forall t \in \text{time periods}
\end{aligned}$$

Where:

T is the total number of time periods,
N is the number of buses,
N' is the number of generation units,
U_{it} is the commitment status of unit i at time t,
P_{it} is the power output of unit i at time t,
C_i^{startup} and $C_i^{\text{operation}}$ are the startup and operating costs of unit i,
$P_i^{\min}$ and $P_i^{\max}$ are the power output limits of unit i,
R_i is the ramping limit of unit i,
D_{jt} is the demand at bus j at time t.

This basic formulation captures the essence of the problem, while practical formulations often also address generator startup and shutdown time and costs, limits on generator cycling, ramp rate limits, reserve margin requirements, and other scheduling constraints as well as power balance equations [57]. Zhu [149], and Bai and Wei [16] are good references for details of the SCUC problem also providing detailed formulations.

Challenges in Generation UC. The Generation UC problem faces the same challenges as ACOPF, with a particular emphasis on combinatoriality because the subset selection problem is intrinsically combinatorial. The non-linear, non-convex, multi-period, combinatorial, stochastic features of the fully constrained problem formulation make it intractable to solve in its natural form. For this reason, solutions rely on relaxations and clever shortcuts to explore the space and find credible candidate solutions that converge on optimality in a reliable manner.

Solution Methodologies. Addressing the challenges posed by the UC problem has spurred the development of diverse solution methodologies which can be sorted on a scale in order of practical vs. academic relevance. On the practical side, frequently we talk about relaxing the problem and using traditional well-established methods to solve the relaxation [110]. A common relaxation is to remove all non-linear elements of the problem such as reactive power, voltage constraints and non-linear costs. This can result in a mixed-integer linear program for which reasonable algorithms exist, and for which small systems can even be solved to optimality in reasonable time. The weakness of this approach is that these relaxations significantly change the fundamental problem such that what we are optimizing is only an approximation of the actual problem. This approach can be characterized as "optimal solutions to approximate problems". The approach presented in Chapter 3 takes a different tack. We instead keep the fully constrained representation of the grid, assume a set of active generators by arbitrarily fixing binary unit commitment variables, use massively parallel computing to solve the fully constrained ACOPF problems with the arbitrary unit commitment, and then iteratively refine that selection. This approach can be characterized as "suboptimal solutions to exact problems" and poses an interesting twist on traditional methods.

On the academic side, there are many papers that explore different meta-heuristic algorithms, such as genetic algorithms, particle swarm optimization, and evolutionary algorithms, that provide alternative strategies for exploring

the complex solution space [68, 42, 52, 133]. These approaches can be supplemented with advanced optimization techniques such as stochastic programming aim to enhance the resilience of solutions against uncertainties in renewable generation and demand forecasts [63, 38, 139]. There is also some research into the integration of machine learning techniques, such as neural networks and reinforcement learning [146, 61]. Such methods may be used as a shortcut prediction for parts or sub-problems of the optimization process, as a starting guess for optimization, or as a complete alternative to the traditional approaches.

Future Directions and Conclusion. As the power generation landscape continues to evolve with the increasing penetration of renewable energy sources and the deployment of advanced grid technologies, the Generation Unit Commitment problem remains a focal point for research and innovation. Future avenues of exploration may include the development of hybrid optimization techniques, the incorporation of real-time data and adaptive learning strategies, and the exploration of decentralized decision-making approaches to address scalability concerns. As the relevance and importance of solving the UC problem increases alongside the ever-increasing stochasticity and complexity of the grid, there will no doubt continue to be significant research and improvements into both existing and novel techniques for advancing the field of power system optimization.

Chapter 3 introduces a novel data-driven method to scalably improve unit commitments under stochastic conditions by exploiting the data produced by ACOPF economic dispatch computations. It is based on the paper [74].

Chapter 3

A data-driven refinement approach to power generation unit commitment

3.1 Introduction

Transformational decarbonization of the electricity generation fleet is needed to achieve emissions reductions targets [107]. This will require a transformation from centralized dispatchable generation towards distributed renewable generation. Inherent variability in generation coupled with impacts of stochastic weather scenarios requires a new class of scalable engineering tools to maintain the reliability requirements of the grid while maximizing utilization of installed renewable generation. In particular, developing grid planning and management approaches that include stochastic factors and contingencies into generation dispatch decisions, is a key requirement for successful renewables integration [125]. These factors are normally considered downstream of the unit commitment (UC) computations in the economic dispatch process for computational reasons [63]. The standard procedure of conducting economic dispatch computations over a set of contingencies and weather scenarios generates data that is imbued with information about contingencies and weather. We propose a method to efficiently "recycle" this data to refine UC, thus including contingencies and weather information into the UC. This data-driven UC refinement is a minimal intervention approach, simply adding an extra step in standard workflows to update UC after economic dispatch.

Our novel data-driven unit commitment approach is enabled by ExaGO [5], a recent development in security constrained alternating current optimal power flow (SC-ACOPF) software that introduces massively parallel computation of contingency and scenario analysis, greatly lowering the time-to-solution of such

analysis. This means we can generate massive amounts of data in the economic dispatch step at low-cost that can be used to refine UC in an iterative manner. This integrated UC and economic dispatch approach leverages today's vastly expanded computational capabilities in a scalable way, showing strong potential to improve both the security and cost of power grids.

UC is a challenging computational problem [148]. Its general formulation – a constrained non-linear mixed-integer optimization problem – is an NP-hard problem that is difficult to solve efficiently even when using parallel computations [52]. Typically, model approximations are made to make the UC problem easier to solve. Most commonly, the model is linearized so that mixed-integer *linear* programming can be used, which can be implemented more efficiently [83]. The weaknesses of this approach are: 1) it is difficult to integrate consideration of contingencies or stochastic scenarios into these solutions [133, 6, 38]; 2) the linear approximations of the UC model do not capture voltage or reactive power constraints, leading to solutions that are only approximations of the fully constrained problem [119]; and 3) the combinatorial nature of the problem formulation limits the scalability and solution accuracy achievable in a reasonable time-frame for large grids [13]. This standard UC approach could thus be improved by injecting information on contingencies and non-linear security constraints in a way that is scalable so that it can be applied to large grids.

Contingency analysis including all security constraints is typically done after UC through a series of alternating current power flow (ACPF) forward simulations [52]. Methods to include contingencies and stochastic information into UC have been studied and proposed, with [6, 133, 63] providing summaries of this research. This is a fundamentally challenging problem since it is large-scale, non-linear, non-convex, and combinatorial, while solutions are expected in short time-frames. Stochastic non-linear mixed-integer programming, that would provide a native way to solve such problems, is still an open research problem requiring more investigation.

On the other hand, there has been significant progress in the development of computational methods for security constrained and stochastic economic dispatch [114, 126, 80, 81]. Robust, parallelizable methods for stochastic SC-ACOPF have been developed and successfully tested on different grid models [3, 110, 137]. These methods are stable with well understood complexity and convergence properties; allowing for scheduling analyses that run within strict time constraints. Furthermore, these analyses can model and *strictly enforce* all security constraints without making approximations. Finally, these methods run efficiently on inexpensive hardware, process a large number of contingencies in a relatively short time, and generate large amounts of high-quality data.

Successful development of methods for security constrained economic dispatch creates new opportunities to exploit the big data contained in the economic dispatch solutions to improve UC. In this work, we propose an integrated security constrained unit commitment and economic dispatch approach that examines data generated in SC-ACOPF analysis to suggest refinements to UC. Simulations show that these refinements have potential to not only increase grid security across large sets of contingencies, but also reduce total generation cost. We call this approach Data-Driven Unit Commitment (DDUC). To keep the presentation streamlined, we consider only day-ahead reliability UC in a deregulated region.

The main contribution of this work is the introduction of a scalable grid planning technique that exploits the data generated by economic dispatch analysis with contingencies to improve UC. The approach is formalized in an algorithm and presented alongside a statistically rigorous set of simulations that give a promising proof-of-concept. The advantages of this approach are:

1. Incorporates contingency analysis and stochastic weather scenarios into computation of UC.

2. Does not use mixed-integer programming for the refinements, instead relying on data analysis techniques that are more scalable for large complex grids.

3. Exploits the data generated by economic dispatch computations that must be done in any case.

4. Additional computations imposed by DDUC utilize graph-based algorithms of linear complexity, meaning they are extremely fast and efficient.

3.1.1 Definitions

Unit Commitment - Given a specific grid topology, a Unit Commitment is a subset of generators in that grid that are active in a given period, therefore able to produce power. Generators have a positive "startup cost" or a cost for being active in a given period. By excluding some generators from the UC, the sum of these startup costs is reduced. In very general terms, a good UC is one that excludes generators whose startup costs are high proportional to the benefit that they contribute to the grid. Identifying these generators is an NP-hard problem that is typically solved using mixed-integer programming techniques that are computationally very costly and difficult to parallelize. In this work, we present an approach to UC that can be conducted in linear time, provided solutions to

the Economic Dispatch problem. While this approach carries no guarantees, empirically it is very effective at improving given UCs.

Grid Cost - The total cost of operating the grid for a given period. This is comprised of the sum of generating costs (mostly comprising fuel, operating, and startup costs), loadshed costs (a penalty assigned for each MW of load that is shed from the grid), and penalties incurred for violating soft constraints such as branch limits or power balances.

Contingencies - Refer to unplanned or unexpected events that can affect the operation of the power system. Contingencies can include equipment failures, such as transmission line or generator outages, or changes in the power demand. Contingencies can lead to violations of operational limits and outages of the power system. In modern power grids that include significant generation from stochastic renewable sources, the importance of contingency analysis is greatly elevated since effective generator outages occur frequently from events such as sudden changes in cloud-cover or wind conditions, or rainfall. Analyzing realistic sets of contingencies on such grids results in huge combinatorial search spaces that are intractable to exhaustively analyze. For this reason, it is attractive to have computational methods that can respond quickly to contingencies as they occur in real time. In this work, we analyze two types of contingencies: generator outages, and transmission line outages.

Scenarios - Refers to a set of possible configurations and operating conditions and environments of the power grid. Scenarios is a broader concept than contingencies, potentially incorporating different sets of weather, generation, demand, economic, and political conditions. In this work, contingencies is a subset of of scenarios and the terms are used interchangeably.

Base-case - The default scenario under consideration. In this work we consider base-cases which are primed with a given UC, then a set of contingencies of that base-case is analyzed. The base-case is that scenario in which there are no unplanned outages.

Security Constraints - Ensure that the power system operates in a safe and stable manner under different operating conditions, such as variations in demand or changes in the network topology. Security constraints prevent violations of operational limits and maintain the reliability of the power system. The ExaGO software package allows for strict enforcement of Security Constraints, with penalized load shedding used to prevent violation of constraints.

Economic Dispatch - Refers to the optimization of power generation in a power system to meet the power demand at minimum cost, while satisfying operational constraints, both linear and non-linear, such as thermal, voltage, and reactive power limits, for a given UC. The objective function in the economic

dispatch problem is typically a quadratic cost function that reflects the variable and fixed costs of generation.

Load shedding - Given a specific grid topology, load shedding is the act of isolating loads from the grid, typically to prevent a violation of security constraints. Load shedding is required to maintain security constraints when the demand for power at certain buses of the grid exceeds the ability of the grid to deliver that power. This is usually due to some combination of high load conditions, insufficient active generation capacity, power transmission constraints, and equipment outages. Concretely, this means that some consumers have their electricity turned off, a highly undesirable outcome. A key assumption of the DDUC approach is that the amount of load shedding that is required to maintain security constraints can be reduced by adjusting the UC, specifically, activating a subset of currently inactive generators.

3.2 Exascale Grid Optimization (ExaGO) Toolkit

The Exascale Grid Optimization Toolkit (ExaGO) [3, 5] is a package for solving large-scale AC optimal power flow problems with stochastic (wind generation, load), security (generation and network contingencies), and scheduling (generator ramping) constraints. It implements scalable algorithms that allow it to run on hardware ranging form a laptop to a supercomputer. ExaGO is portable and can be deployed on traditional CPU and/or heterogeneous GPU-based architectures. It has interfaces to state-of-the-art optimization libraries HiOp [41] and Ipopt [136].

In this work, we use ExaGO's `scopflow` application to solve SC ACOPF application formulated as

$$\min \sum_{c \in \mathcal{C}} f(x_c) \tag{3.1}$$

$$\text{s.t. } g(x_c) = 0, \tag{3.2}$$

$$h(x_c) \leq 0, \tag{3.3}$$

$$x^- \leq x_c \leq x^+, \tag{3.4}$$

$$-\Delta_c x \leq x_c - x_0 \leq \Delta_c x,\ c \neq 0 \tag{3.5}$$

where, $\mathcal{C}$ represents the set of contingencies, including the base-case denoted by subscript 0. `scopflow` aims to minimize the objective $\sum_{c \in \mathcal{C}} f(x_c)$, while adhering to the equality $g(x_c)$, inequality $h(x_c)$, and the lower/upper bound (x^-,x^+) constraints. For notational ease we include the base-case in set $\mathcal{C}$, i.e., $\mathcal{C} \equiv \mathcal{C} \cup c_0$.

Each subproblem c has the detailed formulation of an AC optimal power flow problem. Equation (3.5) represents the coupling between the base-case and each of the contingency states c_i. Equation (3.5) is the most typical form of coupling that limits the deviation of the contingency variables x_c from the base x_0 to within $\delta_c x$. An example of this constraint could be the allowed real power output deviation for the generators constrained by their ramp limit.

For the purpose of demonstrating and testing our DDUC approach, we relax the coupling constraints (3.5) between the base and contingency subproblems. This results in decoupling of AC optimal power flow subproblems for each contingency. Each AC optimal power subproblem is solved in parallel. In essence, this is similar to a parallel AC contingency analysis with the difference that instead of solving power flow for each contingency, we solve an AC optimal power flow. ExaGO's `scopflow` application has an in-built solver called *EMPAR* (short for "embarrassingly parallel") that can be used for such decoupled contingency analysis.

We also model load shedding for each AC optimal power subproblem where each load i can shed an up to γ_i % of its load at a given cost C_i. This load loss formulation allows setting priority or importance to loads (by setting higher costs C_i) and making provision for load that should not be curtailed $(1-\gamma_i)$, for example in the case of critical loads.

3.3 Data-Driven Unit Commitment Algorithm

3.3.1 Algorithm Objectives

Given an initial UC, the DDUC algorithm proposes updated UCs after observing the optimal results of the economic dispatch problem as solved by ExaGO. The algorithm has two objectives: 1) efficiently find a UC that reduces or eliminates the necessity for load shedding over a set of contingencies cases; and 2) exploit the solutions of economic dispatch over time, to evolve the UC to reduce the overall cost of the day-ahead UC. A UC that simultaneously realizes both of these goals would be an unambiguous improvement to grid operation. The DDUC algorithm achieves this by adding "important" generators to the UC and removing "unimportant" ones over a series of iterations.

The identification of "unimportant" generators is done using a data-driven approach that examines the solutions of the SC ACOPF problem finding those generators that have low capacity factors, that is generators whose available capacity is left mostly idle. The hypothesis of this heuristic is that if a generator

Algorithm 1: Data-Driven Unit Commitment

Data:
- $\mathcal{G}$: power grid model,
- $\mathcal{U}$: unit commitment for that grid,
- $\mathcal{C}$: set of contingencies for that grid,
- n : number of iterations

Result: $\mathcal{U}'$: new unit commitment

```
Function DDUC(G, U, C, n):
    U' ← recourse(G,U,C)
    for n do
        U' ← prune(G,U',C)
        U' ← recourse(G,U', C)
    end
return U'
```

provided either low-cost or critical power to the grid, it should have a high capacity factor in the optimal (lowest-cost) SC ACOPF solution. The identification of "important" generators is more nuanced. The basic idea is to find contingency cases that require load shedding to maintain security constraints, then find currently deactivated generators that are "close" to the load that was preferentially shed by the fully-constrained SC ACOPF optimization algorithm for that contingency case. The hypothesis of this heuristic is that those generators that are proximate to the areas of the grid that require load shedding will be most able to provide the missing power. Details on how proximity is measured and how proximate generators are algorithmically identified follow in Section 3.3.2.

The DDUC algorithm works in two phases: a load shed recourse phase where "important" generators are identified and added to the UC, and a pruning phase where "unimportant" generators are identified and removed from the UC. Algorithm 1 defines how these phases are combined to produce the new UC.

3.3.2 Load Shed Recourse Phase

In the load shed recourse phase, the ACOPF solutions for the base-case and each contingency are analyzed to find situations in which load shedding is required to keep from violating security constraints. In each of the load shedding contingency cases, the network is analyzed to find generators that are currently inactive that would be good candidates to add to the unit commitment. A good candidate is a generator that: 1) is close, in the graphical sense, to the bus that shed the most load; and 2) has available transmission capacity on the shortest path

Algorithm 2: Load Shed Recourse

Data:
- $\{\mathcal{G},\mathcal{U},\mathcal{C}\}$,
- α : activation parameter,
- k : number of generators to return from BFS

Result: $\mathcal{U}'$: new unit commitment

Function `recourse(`$\mathcal{G}$, $\mathcal{U}$, $\mathcal{C}$, α, k`)`**:**
  if $\mathcal{C} \supset \{c_0\}$ **then**
    $\mathcal{U} =$ `recourse(`$\mathcal{G}$, $\mathcal{U}$, $\mathcal{C} = \{c_0\}$, α, k`)`
  end
  Run ExaGO `scopflow` on $\mathcal{G},\mathcal{U},\mathcal{C}$
  $\mathcal{S} \leftarrow$ set of cases with load shedding
  $\mathcal{U}' \leftarrow \mathcal{U}$
  for $s \in \mathcal{S}$ **do**
    $\mathcal{U}_s \leftarrow \mathcal{U}$
    $\mathcal{B} \leftarrow$ buses in s with load shedding
    $\pi \leftarrow$ total load shed in s
    $r \leftarrow 0$
    while $r < \alpha\pi$ **and** $\mathcal{B} \neq \varnothing$ **do**
      $b \leftarrow$ pop highest load shed in $\mathcal{B}$
      `breadthFirstSearch(`b,k`)`
      $\mathcal{N} \leftarrow k$ nearest deactivated gens to b
      for $g \in \mathcal{N}$ **do**
        $\rho \leftarrow$ path capacity from g to b
        $\rho' \leftarrow$ generating capacity of g
        $p_g \leftarrow \min(\rho, \rho')$
      end
      $g \leftarrow \arg\max(p)$
      $\mathcal{U}_s \leftarrow \mathcal{U}_s \cup g$
      $r \leftarrow r + \max(p)$
    end
    $\mathcal{U}' \leftarrow \mathcal{U}' \cup \mathcal{U}_s$
  end
return $\mathcal{U}'$

connecting it with that bus.

Such generators are found using a breadth first search (BFS) from the bus that sheds the most load. The search terminates when the number of inactive generators found reaches k, a tuning parameter that effectively controls the relative importance of the graphical proximity and available capacity measures. A low k will emphasize graphical closeness, while a high k will emphasize avail-

able capacity. For each generator in this set, the available capacity on the shortest transmission path between that generator and the bus with load shedding is then measured. The generator with the greatest ability to provide generation to the bus, based on both generator capacity and transmission path capacity is then added to the set of generators to activate. This process repeats for the bus with the next highest amount of load shedding until the generation capacity activated multiplied by parameter α is greater than the total amount of load shedding in the scenario. Once this process has been repeated for each contingency that had load shedding, the union of all sets of generators identified for each case is added to the UC. Full details of this load shed recourse phase are outlined in Algorithm 2.

3.3.3 Pruning Phase

The sole purpose of the pruning phase is to further optimize UC. The ACOPF solution for the base-case and each of the contingency cases is analyzed to find the generators with the lowest capacity factors in the network, across all cases, and then remove some of those generators from the UC. The prune algorithm proceeds by taking a sum of the capacity factor for each generator in the UC over all of the cases / scenarios and then removing the z generators with the lowest total capacity factor across all scenarios (Algorithm 3).

Tuning the parameter z, which defines how many generators to prune at each pass of the DDUC algorithm, is critical to the effectiveness of the algorithm at reducing the cost of the UC. Too low a z, and inefficient generators will remain in the UC. Too high a z, and "important" generators may be mistakenly pruned. Methods for exploring the space of z values and algorithms for reaching optimal z values is something we would like to explore in further research. For this paper we used a percentage of active generators in the range of 2-8% that decreases with every iteration.

3.4 Numerical Experiments

3.4.1 Setup

Numerical experiments to test the effectiveness of the algorithm were conducted on two test grids from the ACTIVSg series [17, 143, 18] – the synthetic SC and Texas grids. At 500 and 2000 buses respectively, these grids are sufficiently complex to provide challenging problems, while being small enough to provide SC

Algorithm 3: Prune Generators

Data:
- $\{\mathcal{G},\mathcal{U},\mathcal{C}\}$,
- z : number of generators to prune

Result: $\mathcal{U}'$: new unit commitment

Function `prune`$(\mathcal{G}, \mathcal{U}, \mathcal{C}, z)$**:**
- Run ExaGO `scopflow` on $\mathcal{G},\mathcal{U},\mathcal{C}$
- $\mathcal{S} \leftarrow$ set of all cases
- $f_g \leftarrow 0,\ \forall\ g \in \mathcal{U}$
- **for** $s \in \mathcal{S}$ **do**
  - **for** $g \in \mathcal{U}$ **do**
    - $f_g \leftarrow f_g +$ capacity factor of g in s
  - **end**
- **end**
- $\mathcal{U}' \leftarrow \mathcal{U}$
- **for** z **do**
  - $g \leftarrow$ pop $\arg\min(f)$
  - $\mathcal{U}' \leftarrow \mathcal{U}' \setminus g$
- **end**

return $\mathcal{U}'$

ACOPF solutions with ExaGO in seconds rather than minutes. For each grid, a single scenario with static load conditions was analyzed, with variability on the supply and transmission side from the contingencies considered. Basic properties of these test grids are shown in Table 3.1.

The experiments were conducted on a cluster with a 64-core AMD 3rd Gen EPYC CPU on each node. The experiments were run using the ExaGO `scopflow` EMPAR formulation using Ipopt [136] as the optimization engine and Pardiso [8, 21, 19] as the solver for linear systems. The `scopflow` EMPAR solver provides massively parallel ACOPF solutions of all contingencies in a fully de-coupled formulation. As such, one compute core is required for each contingency plus one for the base-case to achieve maximal parallel throughput. For the SC grid 590 contingencies were considered in all experiments, representing the full set of $n-1$ contingencies for generators and one branch outage per bus. For the Texas grid a random sample of 500 $n-1$ contingencies were considered including a mix of both branch and generator contingencies. The SC and Texas grids thus required 591 and 501 CPU cores, respectively, to solve fully parallel.

These are relatively small examples in terms of the number of contingencies and scenarios, however, this method is also applicable to extreme-scale analy-

Table 3.1: *Properties of tested grids.*

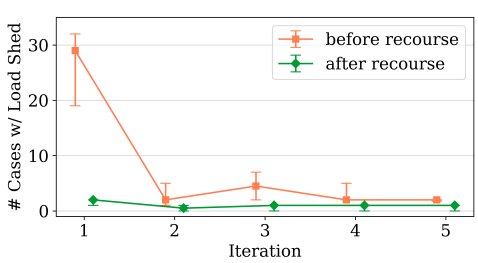
	South Carolina grid	Texas grid
Buses	500	2,000
Generators	90	544
Branches	597	3,206
Installed generation (MW)	12,189	96,292
Total load (MW)	7,751	67,109
Contingencies considered	590	500

ses. ExaGO has been successfully run on Frontier, the world's most powerful and first exascale supercomputer [123], to perform analysis on the 10,000-bus synthetic Western Interconnection model, using 9,999 contingencies and 10 stochastic scenarios, resulting in nearly 100,000 sub-problems. This problem was run on 9,000 compute nodes with 72,000 MPI ranks and successfully completed in 16 minutes. To integrate such runs with the DDUC algorithm would require further research and development, but the linear algorithmic complexity of the data analysis methods make integration with extreme-scale problems feasible.

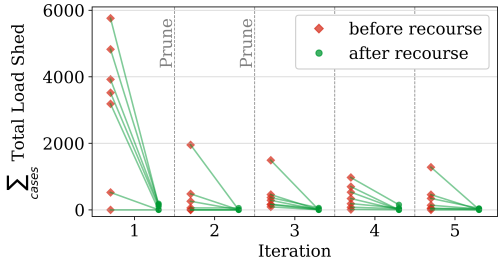
A statistical approach was used to evaluate the performance of the DDUC algorithm given many distinct starting UCs. Specifically, performance was evaluated for 100 stochastically generated starting UCs for each grid. The starting UCs were made by randomly deactivating x percent of the generators in the grid, then adding back generators until the base-case scenario was feasible without load-shedding; $x = \{40\%$ for SC, 30% for Texas$\}$. This statistical approach provides assurance that the positive results are representative of the performance of the algorithm on a broad range of starting UCs and not a circumstance of a specific case. The DDUC algorithm was evaluated using 5 iterations (5 cycles of load shed recourse and pruning), as it was empirically most effective in the first 5 iterations.

3.4.2 Performance Evaluation Criteria

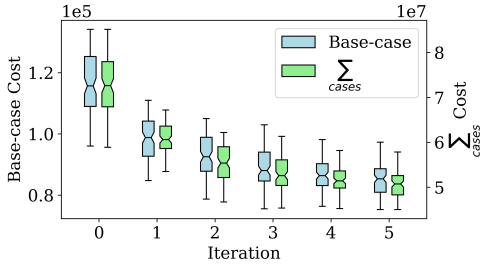
We evaluate the DDUC algorithm for its performance on two metrics: 1) reduction of the amount of load shedding from contingencies, and 2) improvement of the overall cost of running the grid with the suggested UC. The reduction of load shedding given a set of contingencies is the more important of these two metrics, since loss of power for consumers is a highly undesirable event. Any



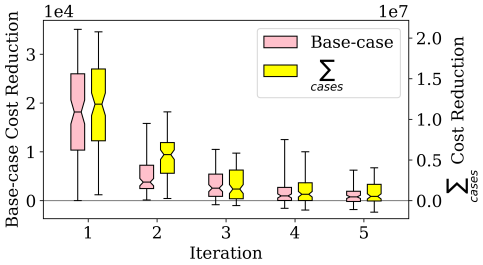
(a) *SC grid number of scenarios with load shedding (n=100)*



(b) *SC grid total load shed reduction from recourse (n=7)*



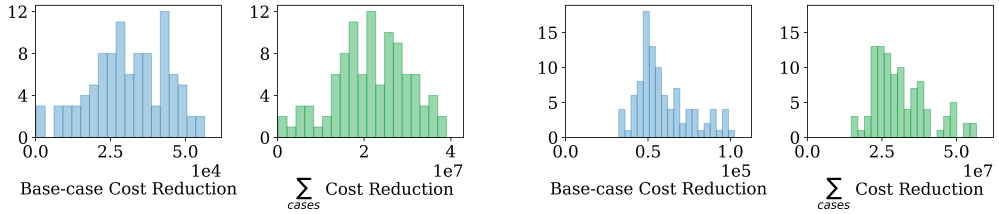
(c) *SC grid absolute cost function value (n=100)*



(d) *SC grid iteration-over-iteration cost reduction (n=100)*

Figure 3.1: *DDUC algorithm performance on South Carolina grid. Figure 3.1a - recourse algorithm reduction in number of load shed cases across all contingencies; error bars show middle quartiles. Figure 3.1b - recourse algorithm reduction of load shed (MW) for a random sample of 7 runs. Figure 3.1c - value of the cost function ($) of the base-case and the sum of all cost function values across all contingency cases. Figure 3.1d - cost reduction ($) achieved on an iteration-over-iteration basis for each of 5 iterations. Box plot notches represent median, box represents middle quartiles, whiskers represent 5th to 95th percentiles.*

approach to find a UC to reduce load shedding under contingencies, however, should be evaluated not only for its benefits of reducing load shedding, but also for its impact on the cost of operating the grid. A UC that reduces load shedding in the contingencies may be undesirable if it significantly increases costs. The following results show that the DDUC algorithm tends to find UCs that reduce both load shedding and cost, giving us a win-win. However, the cost savings results are intrinsically sensitive to the particular cost model employed. Our model includes a cost attached to load shedding, so there is a benefit in the cost metric for preventing load shedding.



(a) *South Carolina grid* ***(b)*** *Texas grid*

Figure 3.2: *Histogram - cost function reduction from initial UC to UC assigned after 5 iterations of DDUC algorithm for n=100 runs, each seeded with different stochastic starting UC.*

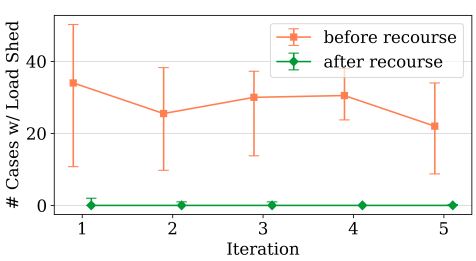
3.4.3 South Carolina Grid Results

Upon testing the DDUC algorithm, we are pleased to see impressive performance on both metrics. Figure 3.1 visualizes the results for testing on the SC grid. The top panel shows performance against the objective of eliminating load shedding, while the lower panel shows performance at lowering overall cost.

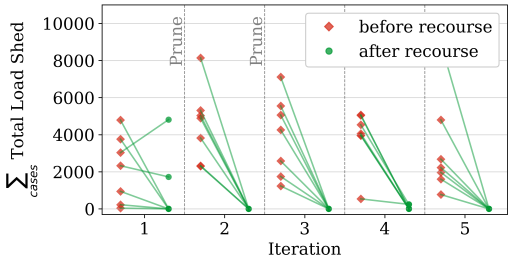
Figure 3.1a shows the consistent ability of the recourse algorithm to find UCs that eliminate the necessity of load shedding from contingency scenarios. This figure shows that arbitrary starting UCs typically must shed load in 20-30 different contingency cases. After the recourse algorithm is run on this starting UC, however, the number of contingency cases requiring load shedding drops considerably to the range of 2-3. Subsequent iterations of the DDUC algorithm do not dramatically increase load shedding, and it is interesting to note that for this case, the recourse algorithm has difficulty eliminating the necessity for load shedding from all contingency scenarios, typically having 1 contingency that still requires it. This suggests that the resilience of SC grid (as described in the model) could be improved with suitable upgrades.

Figure 3.1b shows a slightly different view of the effectiveness of the recourse algorithm. On the y-axis, the sum of load shedding across all contingency cases ($\sum_{cases}$) for a particular UC is indicated by red diamonds before the recourse algorithm and green dots after the recourse algorithm. Only 7 randomly selected starting UCs are shown so that the plot is not too crowded. This figure shows that while there may be significant load shedding required to maintain security constraints across the set of contingency scenarios before the recourse algorithm, after the algorithm, the amount of load shedding required across all contingencies is most often 0 or nearly 0.

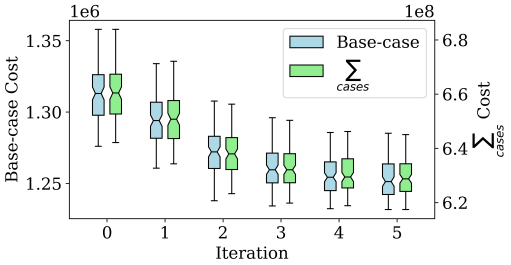
Figure 3.1c shows performance of the DDUC algorithm at reducing the overall



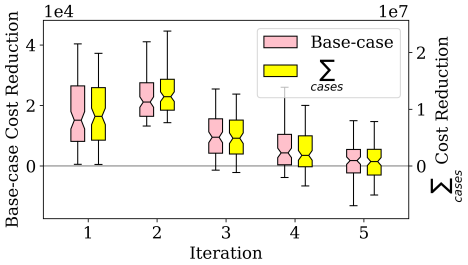
(a) *Texas grid number of scenarios with load shedding (n=100)*



(b) *Texas grid total load shed reduction from recourse (n=7)*



(c) *Texas grid absolute cost function value (n=100)*



(d) *Texas grid iteration-over-iteration cost reduction (n=100)*

Figure 3.3: *DDUC algorithm performance on Texas grid. See Figure 3.1 for detailed description of plots.*

cost of the network. The different colors of boxes represent the cost function value for the base-case, and the sum of the cost functions of all contingency cases including the base-case. An impressive improvement in cost is shown, with the median cost of a starting UC at $131.3k compared to the median cost of $85.3k after 5 iterations of the UC algorithm; an impressive improvement of 26.2%.

Figure 3.1d shows The cost reduction on an iteration-over-iteration basis of the given UCs. What this means is that the metric measures the improvement for each individual UC over the five iterations of the algorithm, yielding a positive value if the cost decreased during the iteration and a negative value if the cost increased during the iteration. These measures are then aggregated in the box plot. We see that for each of the five iterations, the algorithm resulted in a cost reduction to the system in the vast majority of instances. Notably, in both the first and second iteration, at least 95% of the time the algorithm resulted in reductions for both the base-case cost and the sum of contingency cases cost.

Figure 3.2a shows a histogram of the reduction in the value of the cost function for 100 runs of the DDUC algorithm. This shows us the distribution of the cost reduction to be expected from an arbitrary starting UC to a corrected UC for

the SC grid. Considering the base-case cost, over the 100 run sample: the minimum cost reduction observed was $313, the mean cost reduction observed was $30,986, and the maximum cost reduction observed was $56,432. Importantly, in none of the 100 runs was a cost increase observed.

3.4.4 Texas Grid Results

Repeating the same tests on the Texas grid, we see similarly positive results. Figures 3.2b and 3.3 visualize these results on the Texas grid test case.

In Figure 3.2b, we see that considering the base-case cost, over the 100 run sample: the minimum cost reduction observed was $32,341, the mean cost reduction observed was $59,587, and the maximum cost reduction observed was $102,508. While theses reductions are lower in relative terms than they were for the SC grid, they are significant in absolute terms. The distribution is also more impressive on the Texas grid in that even the minimum observed cost reductions are significantly positive. Again, in none of the 100 runs, was a cost increase observed.

Figure 3.3a shows the impressive results of the DDUC algorithm for eliminating load shed also persist on the Texas grid case. This figure shows that arbitrary starting UCs typically must shed load in 15-50 different contingency cases. After the recourse algorithm is run on this starting UC, however, the number of cases requiring load shedding drops considerably to the range of 0-2. Subsequent iterations of the DDUC algorithm perform similarly well, eventually eliminating load shedding altogether seeing no load shedding cases for iterations 4 and 5 in the middle quartiles of data. This is a better result than on the SC grid, and is perhaps attributable to the greater size and diversity of the Texas grid making it less vulnerable to a particular outage.

In Figure 3.3b, we see that during the first iteration of the DDUC algorithm there is one run where total load shedding over all cases increases after the recourse algorithm. This is not persistent however, as the recourse algorithm consistently eliminates load shedding on subsequent iterations. The high level of load shedding after the pruning phase in contrast to the SC grid is attributable to the fact that the larger Texas grid undergoes more aggressive pruning since the parameter of how many generators to eliminate is set as a percentage of the active generators, and was set higher in the Texas grid.

Figure 3.3c shows an impressive improvement in cost, with the median cost of a starting UC at $1.31m compared to the median cost of $1.25m after 5 iterations of the UC algorithm; an improvement of 4.7%.

In Figure 3.3d, again we see that for each of the five iterations, the algorithm

resulted in a cost reduction to the system in the majority of instances. Notably, in both the first and second iteration, at least 95% of the time the algorithm resulted in reductions for both the base case cost and the sum of contingency scenario costs. Interestingly, in the Texas grid, the DDUC algorithm does a better job of reducing cost in the second iteration than the first one.

3.5 Discussion

The data-driven approach to finding UCs that are both secure and low-cost has some fundamental merits that make it worthwhile exploring in more depth. Perhaps the most important of these is that the input data for the approach is coming from a stochastic SC ACOPF process that considers both contingencies and weather scenarios. Using this data as a feed-stock for UC refinement therefore provides the possibility to find UCs that are best with respect to not only some base-case scenario, but also to an aggregation of possible contingencies and weather conditions. The ability to include contingencies and stochastic factors into the UC optimization process is going to be critical to increasing the penetration of renewables without sacrificing grid reliability. In addition, inducting knowledge of contingencies and weather into the UC optimization process could give grid planners confidence to tighten the safety margins used in UC decisions. This would allow for more cost-effective grid operation in the same way that superior calculation techniques in civil engineering allowed for building designs using fewer materials.

Another important benefit of this DDUC approach is that the incremental computations to find the UC refinements are extremely efficient and low-cost. The underlying algorithms are linear complexity with respect to both the grid topology, and the number of contingencies and scenario cases considered. In contrast to the standard mixed-integer programming approach to UC, this means that the method is highly scalable as network complexity increases.

A potential deployment of the DDUC approach could be: an initial seed UC is found using the traditional mixed-integer approach. Economic dispatch with respect to contingencies and weather scenarios is then run using this seed UC. The data from the economic dispatch is then recycled into the DDUC algorithm to refine the seed UC with the knowledge of contingencies and weather scenarios that is contained in the economic dispatch solution, and this refinement process is repeated as new information on the state of the grid and the weather becomes available.

Another interesting possible application of the DDUC approach is for black-

start operations where it is known a priori which units to start and due to the weak grid conditions, the chances of load shedding due to improper unit commitment is significant. In such a case, the DDUC approach could provide feasible and optimal grid operational solutions with no or least amount of load shed.

The primary goal of the DDUC algorithm is to reduce load shedding across a range of contingencies by finding UCs that are less vulnerable to the *ensemble* of contingencies. This is a critical consideration since more than 13 million people in the United States were affected by power outages each year from 2008 to 2015, and the annual number of outages grew from 2,169 to 3,571 in this same time period [64]. Another interesting area of research to explore would be "demand-response" which amounts to voluntary load shedding. Demand-response programs are common and rapidly growing in the US, and usually involve some incentive that is provided by the grid operator to induce large consumers to lower their consumption when conditions are tight [14]. The DDUC approach would allow for demand-response to be explicitly modeled using the cost of the incentive as the cost for load reduction variables in ExaGO. This is an exciting possibility that could allow operators to better understand the relationship between UCs and demand-response requirements.

While the results of the numerical experiments presented herein are preliminary, they give reason to justify optimism in the possibilities of the DDUC approach as it is refined through further research. These experiments have shown a validation of the heuristics, showing their ability to identify both the "important" and "unimportant" generators in the grid. While benefits in terms of total generation cost will likely diminish with more optimal seed UCs generated from a mixed-integer programming approach, the DDUC method should be recognized for its ability to efficiently include contingency and weather information into the UC. While the experiments were only run with contingencies in this work, the inclusion of stochastic weather scenarios would be a simple extension since this functionality is already fully supported by ExaGO.

3.6 Conclusion

To achieve decarbonization goals [107], we need to develop tools to efficiently manage highly dynamic and stochastic power grids. Stochasticity of weather and operations and their impact on the grid can be modeled effectively in the economic dispatch process using standard models and proven approaches. New computational tools such as ExaGO allow us to vastly scale-up stochasticity modeling efforts by leveraging parallel computing and modern computational re-

sources to give high-resolution information in short time-frames. The outcome is a tool set and data to support the decision-making process on resilience investments, improving operational efficiency and grid reliability. These economic dispatch computations with contingencies and stochastic scenarios generate huge amounts of high-quality data that can be analyzed and effectively "recycled" to imbue contingency and stochastic information into unit commitment through heuristic refinements. This process is a minimal intervention approach that is intrinsically scalable given the linear complexity of the graph-based algorithms used to analyze the economic dispatch output data. The preliminary results presented in this work show that this data-driven unit commitment approach has merit. It performed effectively on non-trivial grids of 500 and 2000 buses, consistently providing unit commitments that strictly enforced all security constraints (including non-linear) and minimized or fully eliminated the need for load shedding across contingencies with power imbalance.

With further research, this data-driven unit commitment refinement approach could be tested across a wide array of grids, with larger contingency sets, with stochastic weather scenarios, and with different seed unit commitments. With further development and a robust implementation, this could prove to be a computationally efficient and practical method to include contingency and stochastic weather information into the unit commitment optimization process; providing more reliable and efficient grids with high penetrations of renewable energy.

3.7 Scientific Software: ExaGO

ExaGO: the exascale grid optimization toolkit, is a set of software tools designed to enable scalable power grid optimization such that AC optimal power flow problems can be extended to extreme scales, exploiting exascale supercomputers such as the Frontier supercomputer at Oak Ridge National Laboratory [123]. ExaGO was developed as part of the Exascale Computing Project (ECP), a $1.8 billion U.S. Department of Energy investment in development of new computational technology. ExaGO represents the state-of-the-art in scalable AC optimal power flow for fully constrained problems, scalably modeling fully constrained grids including stochastic (wind generation, load), security (generation and network contingencies), and scheduling (generator ramping) constraints [4].

To enable exascale grid optimization, two key breakthroughs were needed. The first being optimization techniques and algorithms that allow for grid optimization problems to scale close to linearly in the number of stochastic scenarios. The second being optimization software that utilizes hardware accelerators such

as GPUs. The following subsections describe these two challenges in more detail.

3.7.1 Scalable ACOPF Optimization Techniques

The need for scalable algorithms for grid simulation arises from the necessity of grid operators to plan for contingencies such as equipment outages and ensure that such outages will not cause catastrophic or cascading failures in the grid. Simulation of contingencies typically involves the simulation of an ACOPF model for each contingency, which can number in the tens of thousands, to find a secure state of the grid [50]. The large number of contingencies is a function of "$N-1$" grid operation paradigm in which grid security must be assured if any one piece of equipment fails. As such, there must be a contingency that simulates the failure of a given piece of grid infrastructure for all grid infrastructure [105]. As the sensitivity of the grid to stochastic phenomena such as weather increases, simulations of "$N-2$" or even "$N-3$" contingencies may be desireable. Given that these are combinatorial formulations, $\binom{N}{2}$, and $\binom{N}{3}$, they result in extremely large sets of scenarios. Since the global minimum subject to the ensemble of all contingencies is desired, all contingencies can be bundled into a single optimization problem, however this results in extremely large systems that quickly become intractable. A novel solution to this problem is to break the scenarios into separate problems, that are coupled using a coupling variable in a two-stage stochastic optimization scheme. ExaGO exploits this with the primal decomposition technique [139, 137] as implemented in the HiOP optimization software [41]. The following description from [139] describes the key idea of the primal decomposition method. The primal decomposition solver breaks the problem down into a two-stage stochastic optimization problem with recourse of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x)+\mathcal{R}(x) \\ \text{s.t.} \quad & c(x)=c_E \\ & d^l \le d(x) \le d^u \\ & x^l \le x \le x^u, \end{aligned}$$

where the recourse function $\mathcal{R}(x)=\mathbb{E}_\Omega[r(x,\omega)]$ is defined as the expectation (integral) of the optimal value function $r(x,\omega)$ of the second-stage problem parameterized by a random vector ω over a probability space Ω. More specifically,

the second-stage optimal value function has the following mathematical form:

$$\begin{aligned} r(x,\omega) = \min_{y\in\mathbb{R}^m} \quad & p(y,x,\omega) \\ \text{s.t.} \quad & c(y,x,\omega) = c_E(\omega) \\ & d^l(\omega) \le d(y,x,\omega) \le d^u(\omega), \\ & y^l(\omega) \le y \le y^u(\omega), \end{aligned}$$

The second-stage problem is dependent on the first-stage through the coupled variable x. For further details on the formulation, the interested reader should consult the full paper by Wang et al. [139]. Essentially, this optimization technique breaks the problem down into a series of sub-problems overseen by a coordination problem. In the research presented in Chapter 3 the sub-problems were solved using Ipopt [136].

3.7.2 Hardware Accelerated Optimization

The interior point algorithm is the typical solution method of choice for these large-scale non-linear, non-convex optimization problems given its robust properties and scalability and efficiency for handling large systems [13]. The majority of computation in the interior point algorithm is the solution of a sparse system of equations [81]. This is a major advantage of the interior point algorithm, since this heavy computational task can be off-loaded to highly efficient and optimized parallel numerical libraries such as Pardiso [8]. The disadvantage of such libraries is that they currently do not use hardware accelerators such as GPUs. This lack of support for GPUs is an inherent barrier to exascale computations, since the exascale barrier has only been broken thanks to a heavy reliance on the extremely high arithmetic performance of GPUs[1].

Solving sparse linear systems on GPUs poses challenges due to the inherent nature of sparse matrices and the complexities of parallelizing certain solution methods. Sparse matrices contain mostly zero elements, leading to irregular memory access patterns and potentially inefficient GPU memory utilization. These irregular access patterns and data layout make exploiting data level parallelism – the specialty of GPUs – inherently difficult [109].

Direct solving methods, such as Gaussian elimination or LU decomposition, involve factorizing the matrix into lower and upper triangular matrices. These methods can benefit from parallelism in certain steps, such as partial pivoting or triangular solve. While the factorization itself can be parallelized using pipeline

[1]The world's first certified exascale supercomputer, Frontier, at Oak Ridge National Laboratory derives over 99% of its arithmetic performance from GPUs [95].

parallelism such as in the left-right looking strategy [122], translation of these algorithms to GPUs are inherently difficult due to dependencies between elements, synchronization events, and irregular access patterns. Additionally, the memory requirements for storing the factorized matrices might be prohibitive for large sparse systems [121].

Iterative methods, such as conjugate gradient (CG) or preconditioned iterative methods (e.g., GMRES), are often favored for sparse linear systems on GPUs due to their ability to exploit sparsity and their potential for parallelism [153]. These methods involve iterative updates to approximate the solution, where each iteration typically consists of matrix-vector multiplication and vector operations. While these operations can be parallelized efficiently on GPUs, achieving convergence may require a large number of iterations, which can limit the performance benefits, especially if the convergence rate is slow or if the preconditioning step is computationally expensive [115].

Overall, the difficulty in parallelizing sparse linear system solution methods on GPUs lies in balancing memory access patterns, exploiting parallelism in matrix operations, managing dependencies between matrix elements, and optimizing convergence criteria. Direct methods face challenges in efficiently factorizing sparse matrices, while iterative methods require careful optimization to balance computation and memory access while ensuring rapid convergence.

Progress is being made, with significant research in the field of GPU enabled sparse linear solvers, such as next-generation linear algebra libraries for heterogeneous computing presented in Matrix Algebra on GPU and Multi-core Architectures (MAGMA) [11, 144, 45], the NVIDIA cuSPARSE library [1, 131], hybrid direct-iterative methods such as HyKKT [116], or the gpu-based solver implemented in HiOP [41] as part of the Exascale Computing Project. For a comprehensive review of the current state-of-the-art, I refer the reader to [153]. ExaGO can be configured to use HiOP GPU based solver for the sub-problems, resulting in truly exascale computations for the stochastic ACOPF problems. ExaGO has been used for Stochastic security constrained alternating current optimal power flow (SC ACOPF) analysis for 10,000-bus model of Western U.S. grid with 100,000 stochastic scenarios performed on 72,000 MPI ranks and using 9,000 nodes on Frontier exascale computer (72,000 GPUs). The analysis successfully converged to the solution which was validated against commercial tools [128]. That said, this is still a field under development, and currently proposed solutions for GPUs are not at the same level of maturity as the CPU counterparts.

3.7.3 Towards Scalable Stochastic Optimal Power Flow

With the breakthroughs of allowing coupled ACOPF optimizations such as those with contingencies to scale roughly linearly in the number of scenarios, and recent developments of GPU enabled sparse linear solvers, we are entering the era where large stochastic grid optimizations are becoming tractable given sufficient computational resources. While further development is required to make these techniques stable, robust, and trusted — ExaGO has served as a breakthrough proof of concept and development platform for the next generation of stochastic optimal power flow technology.

3.8 Appendix: ExaGO ACOPF Forumulation

The ExaGO base program is OPFLOW – a fully constrained AC optimal power flow model that is then called by higher level programs that model contingencies, stochastic scenarios, and multi-period problems. The OPFLOW formulation includes:

- An objective function including generation cost, deviation cost, load loss costs, and power imbalance costs
- Nodal power balance
- Shunt power
- Generator real and reactive power output
- Branch flows
- Automatic generation control (optional)
- Generator bus voltage control

The following details on the formulation are provided by the ExaGO User Manual [4].

Optimal power flow is a general nonlinear programming problem of the form

$$\begin{aligned} \min.\ & f(x) && (3.6)\\ & \text{s.t.} \\ & g(x) = 0 && (3.7)\\ & h(x) \leq 0 && (3.8)\\ & x^{\min} \leq x \leq x^{\max} && (3.9) \end{aligned}$$

With decision variables x, bounded to $x^{\min}$ and $x^{\max}$, objective function $f(x)$, and equality and inequality constraints $g(x)$ and $h(x)$.

3.8.1 Variables and bounds

Table 3.2 describes the optimization variables that are modeled by OPFLOW. The power imbalance variables are slack variables that measure the violation

Table 3.2: *Optimal power flow (OPFLOW) variables*

Symbol	**Variable**	**Bounds**
p_j^g	Generator real power dispatch	$p_j^{gmin} \le p_j^g \le p_j^{gmax}$
q_j^g	Generator reactive power dispatch	$q_j^{gmin} \le q_j^g \le q_j^{gmax}$
Δp_j^g	Generator real power deviation	$-p_j^r \le \Delta p_j^g \le p_j^r$
p_j^{gset}	Generator real power set-point	$p_j^{gmin} \le p_j^{gset} \le p_j^{gmax}$
ΔP	System power excess/deficit	Unbounded
θ_i	Bus voltage angle	-$\pi \le \theta_i \le \pi$
v_i	Bus voltage magnitude	$v_i^{min} \le v_i \le v_i^{max}$
$\Delta p_i^+, \Delta p_i^-$	Bus real power mismatch variables	$0 \le \Delta p_i^+, \Delta p_i^- \le \infty$
$\Delta q_i^+, \Delta q_i^-$	Bus reactive power mismatch variables	$0 \le \Delta q_i^+, \Delta q_i^- \le \infty$
Δp_j^l	Real power load loss	$0 \le \Delta p_j^l \le p_j^l$
Δq_j^l	Reactive power load loss	$0 \le \Delta q_j^l \le q_j^l$

of power balance at buses. Inclusion of these variables ensures feasibility of the bus power balance constraints.

3.8.2 Objective Function

$$\text{min. } C_{gen}(p^g) + C_{dev}(\Delta p^g) + C_{loss}(\Delta p^l, \Delta q^l) + C_{imb}(\Delta p^+, \Delta p^-, \Delta q^+, \Delta q^-) \quad (3.10)$$

Total generation cost $C_{gen}(p^g)$

$$C_{gen}(p^g) = \sum_{j \in J^{gen}} C_j^g(p_j^g) \quad (3.11)$$

Where C_j^g is a quadratic function of the form $C_j^g = a_j^g p_j^{g2} + b_j^g p_j^g + c_j^g$.

Total generation setpoint deviation $C(\Delta p^{g})$

$$C_{dev}(\Delta p^{g}) = \sum_{j \in J^{\text{gen}}} (\Delta p_j^{g2}) \tag{3.12}$$

Load loss $C(\Delta p^{l}, \Delta q^{l})$

$$C_{loss}(\Delta p^{l}, \Delta q^{l}) = \sum_{j \in J^{\text{ld}}} \sigma_j^{l} (\Delta p_j^{l} + \Delta q_j^{l}) \tag{3.13}$$

With default load loss penalty σ_j^{l} is \$1000/MW for all loads.

Power imbalance $C_{imb}(\Delta p^{+}, \Delta p^{-}, \Delta q^{+}, \Delta q^{-})$

$$C_{imb}(\Delta p^{+}, \Delta p^{-}, \Delta q^{+}, \Delta q^{-}) = \sum_{i \in J^{bus}} \sigma_i (\Delta p^{+} + \Delta p^{-} + \Delta q^{+} + \Delta q^{-}) \tag{3.14}$$

Imbalance variables $\Delta p^{+}, \Delta p^{-}, \Delta q^{+}, \Delta q^{-}$ are slack or non-physical variables that ensure feasibility of the bus power balance constraints and can provide a measure of infeasibility.

3.8.3 Equality constraints

Nodal power balance

The nodal power balance equations for each bus i are given by

$$\sum_{\substack{j \in J^{\text{gen}} \\ A_{ij}^{g} \neq 0}} p_j^{g} = p_i^{\text{sh}} + \Delta p_i^{+} - \Delta p_i^{-} + \sum_{\substack{j \in J^{\text{ld}} \\ A_{ij}^{l} \neq 0}} (p_j^{l} - \Delta p_j^{l}) + \sum_{\substack{j \in J^{\text{br}} \\ A_{oi}^{\text{br}} \neq 0}} p_{j_{od}}^{\text{br}} + \sum_{\substack{j \in J^{\text{br}} \\ A_{id}^{\text{br}} \neq 0}} p_{j_{do}}^{\text{br}} \tag{3.15}$$

$$\sum_{\substack{j \in J^{\text{gen}} \\ A_{ij}^{g} \neq 0}} q_j^{g} = q_i^{\text{sh}} + \Delta q_i^{+} - \Delta q_i^{-} + \sum_{\substack{j \in J^{\text{ld}} \\ A_{ij}^{l} \neq 0}} (q_j^{l} - \Delta q_j^{l}) + \sum_{\substack{j \in J^{\text{br}} \\ A_{oi}^{\text{br}} \neq 0}} q_{j_{od}}^{\text{br}} + \sum_{\substack{j \in J^{\text{br}} \\ A_{id}^{\text{br}} \neq 0}} q_{j_{do}}^{\text{br}} \tag{3.16}$$

(3.17)

where, the real and reactive power shunt consumption is given by (3.18) and (3.19)

The real and reactive power flow $p_{j_{od}}^{\text{br}}$, $q_{j_{od}}^{\text{br}}$ for line j from the origin bus o to destination bus d is given by (3.24) – (3.25) and from destination bus d to origin bus o is given by (3.26) – (3.27)

Shunt power

$$p_i^{\text{sh}} = g_i^{\text{sh}} {v_i}^2 \tag{3.18}$$
$$q_i^{\text{sh}} = -b_i^{\text{sh}} {v_i}^2 \tag{3.19}$$

Generator real power output

Generator real power output p_j^{g} relates to power deviation Δp_j^{g} by the following relations

$$p_j^{\text{gset}} + \Delta p_j^{\text{g}} - p_j^{\text{g}} = 0 \tag{3.20}$$
$$p_j^{\text{gset}} - p_j^{\text{g}^*} = 0 \tag{3.21}$$

The second equation sets generator set-point p_j^{gset} to a fixed value $p_j^{\text{g}^*}$. Here, $p_j^{\text{g}^*}$ is the set-point for the generator real power output.

3.8.4 Inequality constraints

MVA flow on branches

MVA flow limits at origin and destination buses for each line.

$$ {p_{j_{od}}^{\text{br}}}^2 + {q_{j_{od}}^{\text{br}}}^2 \leq {s_j^{\text{rateA}}}^2, \quad j \in J^{\text{br}} \tag{3.22}$$
$$ {p_{j_{do}}^{\text{br}}}^2 + {q_{j_{do}}^{\text{br}}}^2 \leq {s_j^{\text{rateA}}}^2, \quad j \in J^{\text{br}} \tag{3.23}$$

Branch flows

In polar coordinates, the real and reactive power flow $p_{j_{od}}^{\text{br}}$, $q_{j_{od}}^{\text{br}}$ from bus o to bus d on line j is given by (3.24) – (3.25)

$$p_{j_{od}}^{\text{br}} = g_{oo} v_o^2 + v_o v_d (g_{od} \cos(\theta_o - \theta_d) + b_{od} \sin(\theta_o - \theta_d)) \tag{3.24}$$
$$q_{j_{od}}^{\text{br}} = -b_{oo} v_o^2 + v_o v_d (-b_{od} \cos(\theta_o - \theta_d) + g_{od} \sin(\theta_o - \theta_d)) \tag{3.25}$$

and from bus d to bus o is given by (3.26) – (3.27)

$$p^{\text{br}}_{j_{do}} = g_{dd} v_d^2 + v_d v_o (g_{do} \cos(\theta_d - \theta_o) + b_{do} \sin(\theta_d - \theta_o)) \tag{3.26}$$

$$q^{\text{br}}_{j_{do}} = -b_{dd} v_d^2 + v_d v_o (-b_{do} \cos(\theta_d - \theta_o) + g_{do} \sin(\theta_d - \theta_o)) \tag{3.27}$$

Automatic generation control (AGC)

ACG adds two additional constraints for each participating generator to enforce the proportional generator redispatch participation as done in automatic generation control (AGC). These two equations are

$$\begin{aligned} \left(\alpha^{\text{g}}_j \Delta P - \Delta p^{\text{g}}_j\right)\left(p^{\text{g}}_j - p^{\text{gmax}}_j\right) \geq 0 \\ \left(\Delta p^{\text{g}}_j - \alpha^{\text{g}}_j \Delta P\right)\left(p^{\text{gmin}}_j - p^{\text{g}}_j\right) \geq 0 \end{aligned} \tag{3.28}$$

Eq. 3.28 forces the generator set-point deviation to be equal to the generation participation when the generator has head-room available $p^{\text{gmin}}_j \leq p^{\text{g}}_j \leq p^{\text{gmax}}_j$. Here, α^{g}_j is the generator participation factor which is the proportion of the power deficit/excess ΔP that the generator provides.

Generator bus voltage control

The generator bus voltage can be fixed when the total reactive power generation available at the bus is within bounds. When it reaches its bounds, the voltage varies with the generator reactive power fixed at its bound. To implement this behavior, two inequality constraints are added for each generator bus

$$\begin{aligned} (v^{\text{set}}_i - v_i)(q_i - q^{\text{max}_i}) \geq 0 \\ (v_i - v^{\text{set}}_i)(q^{\text{min}_i} - q_i) \geq 0 \end{aligned} \tag{3.29}$$

Here, q_i, q^{max_i}, and q^{min_i} are the generated, maximum, and minimum reactive power at the bus, respectively.

Part III

Cycle Detection in Gasoline Markets

Forward to Part III: Applying Computational and Data Science Methods to Advance the Boundaries in Economics Research

This following chapter changes focus from electrical power systems to gasoline markets, while at the same time shifting from more classical numerical analysis to large scale data science and machine learning. The goal of the project was straightforward: identify or construct models that can scalably and reliably detect cyclical patterns in gasoline price data. The cyclical patterns in question are Edgeworth Cycles, which are covered in detail in the following chapter. These patterns are particularly interesting to economists because they are underpinned by an elegant game theory model, and we see them emerging in consumer markets. Such emergence may have important economic and policy implications, as it could indicate price collusion at the expense of the public.

Research into this phenomena, and it's impact on the consumers, has been impeded by the absence of a systematic definition of these price cycles as they occur in empirical data. The research was relying on a patchwork of simple heuristics that were parameterized by the age-old technique of "eyeballing" the data. As we show in the paper, by using different empirical definitions of these cycles you can in fact come to completely different economic conclusions about the consequences of this phenomenon.

The research was conducted for the benefit of economists studying price cycling phenomena, and thus the following chapter, based on our paper published in *The Journal of Law and Economics* [73], is written for an audience of economics researchers. Nonetheless, the research presented in the following chapter is underpinned by a significant computational infrastructure that is capable of processing large data sets, (on the order of 80GB) in an efficient manner into custom data structures with flexible interfaces to interact with a range of 16 different models ranging from simple parametric models to tree-based machine learning models, to deep neural networks. The results presented in this paper flow from 1000s of node hours on a state-of-the-art GPU cluster in order to develop the models and conduct a thorough uncertainty quantification process for model accuracy. Doing so also involved implementing a custom optimization engine into the framework that is able to handle the specific interfacing needs of this project and the poorly behaved non-convex discrete functions. We were happy with the research outcomes, as the proposed computational models can scalably identify price cycles with an accuracy of 80 to 99% depending on the data set.

The code-base for this project grew to over 30,000 lines of code, before being trimmed back into a compact and easy to use package that is now available on Github to empower researchers from economics to leverage computation to achieve their research goals. While the presentation style of the following chapter is different from that which we are familiar with in computational science, I present it here as a model example of how interdisciplinary collaboration between computational scientists and domain experts can significantly push the frontiers of research in other disciplines.

Chapter 4

Computational techniques to scalably identify pricing cycles in retail gasoline markets

4.1 Introduction

Retail gasoline prices are known to follow cyclical patterns in many countries [30]. The patterns persist even after controlling for wholesale and crude-oil prices. Because these cycles are so regular and conspicuous, and because price increases tend to be larger than decreases, observers suspect anti-competitive business practices. The occasional discovery of price-fixing cases supports this view [40, 55, 138].[1]

These asymmetric movements are called Edgeworth cycles and have been studied extensively.[2] In particular, scholars and antitrust practitioners have investigated whether the presence of cycles is associated with higher prices and markups. [43, 40, 29] find that asymmetry is correlated with higher margins, price-fixing collusion, and concentrated market structure, respectively. However, [92, 150, 103] show prices and margins are *lower* in markets with asymmetric price cycles. Given the diversity of countries and regions in these studies (Australia, Canada, the US, and several countries in Europe), the cycle-competition

[1] Recent studies on algorithmic collusion suggest interactions between self-learning algorithms could lead to collusive equilibria with such cycles [82]; the use of "repricing algorithms" by many sellers on Amazon has made these phenomena prevalent in e-commerce as well [100].

[2] Maskin and Tirole (1988) [97] coined the term after Edgeworth's (1925) hypothetical example [49]. It became a popular topic for empirical research since Castanias and Johnson (1993) [35]. We explain its theoretical background in section 2.

relationship could be intrinsically heterogeneous across markets.

But another, perhaps more fundamental, problem is measurement: the lack of a formal definition or a reliable method to detect cycles in large datasets. Because theory provides only a loose characterization of Edgeworth cycles, empirical researchers have to rely on visual inspections and summary statistics based on a single quantifiable characteristic: asymmetry. Meanwhile, the phenomena's most basic property, cyclicality, is almost completely absent from the existing operational definitions. Even though asymmetry may be the most salient feature of—and hence a necessary condition for—Edgeworth cycles, it is not a sufficient condition. Empirical findings are only as good as the measures they employ; the incompleteness of detection methods could affect the reliability of "facts" about competition and price cycles. Now that the governments of many countries and regions are making large-scale price data publicly available,[3] developing scalable detection methods represents an important practical challenge for economists and policymakers.[4]

This work proposes a systematic approach to detecting Edgeworth cycles using scalable computational techniques and machine learning models. We formalize four existing methods as simple parametric models: (1) the "positive runs vs. negative runs" method of [35], (2) the "mean increase vs. mean decrease" method of [47], (3) the "negative median change" method of [92], and (4) the "many big price increases" method of [30]. We then propose six new methods based on spectral analysis and nonparametric/machine-learning techniques: (5) Fourier transform, (6) the Lomb-Scargle periodogram, (7) cubic splines, (8) long short-term memory (LSTM), (9) an "ensemble" (aggregation) of Methods 1-7 within a random-forests framework, and (10) an ensemble of Methods 1-8 within an extended LSTM.

To evaluate the performance of each method, we collect data on retail and wholesale gasoline prices in two regions of Australia, Western Australia (WA) and New South Wales (NSW), as well as the entirety of Germany. These datasets

[3]The governments of Australia, Germany, and other countries have made detailed price data publicly available to inform consumers and encourage further scrutiny. The Australian Consumer and Competition Commission has a team dedicated to monitoring gasoline prices and regularly publishes reports. See https://www.accc.gov.au/consumers/petrol-diesel-lpg/about-fuel-prices. The Bundeskartellamt does the same in Germany.

[4]Systematic methods to detect price cycles are useful for researchers who do *not* want to study cycles as well. Chandra and Tappata (2011) [36] examine the role of consumer search in generating temporal dispersion in the US retail gasoline prices. However, they could not completely reject Edgeworth cycles as an alternative explanation (see their page 697 and footnote 46) because they did not have a scalable method to prove the absence of cycles in their large dataset of more than 25,000 stations. Our procedure would have allowed them to provide more concrete evidence.

cover the universe of gasoline stations in these regions/countries, record each station's retail price at a daily (or higher) frequency, and are made publicly available by legal mandates.[5] Given the lack of a clear theoretical definition, we construct a benchmark "ground truth" based on human recognition of price cycles as follows. We reorganize the raw data as panel data of the daily margins (= retail minus wholesale prices) of gasoline stations and group them into calendar quarters, so that a station-quarter (i.e., a set of 90 consecutive days of retail-margin observations for each station) becomes the effective unit of observation. We employ eight research assistants (RAs) to manually classify each station-quarter as either "cycling", "maybe cycling", or "not cycling". We then define a binary indicator variable that equals 1 if an observation is labeled as "cycling" by all of the RAs (the majority of observations are labeled by three RAs), and 0 otherwise, thereby preparing a conservative target for automatic cycle detection. Note that we look only for cyclicality and do not impose asymmetry or other criteria in the manual-classification stage. The reason is that asymmetry is—unlike cyclicality—amenable to clear mathematical definitions and can easily be checked at a later stage. Hence, we prioritize the detection of cyclicality, thereby alleviating the cognitive burden on RAs.

At this point, one might wonder whether human recognition of cycles is an appropriate benchmark. We regard it as the best *feasible* option (the "second best") given the lack of clear theoretical definitions (the "first best"). Manual classification by a team of RAs represents a best-effort practice in the literature and provides a relevant—if not perfect—benchmark in the following sense. First, most existing studies employ some rule-of-thumb definitions with calibrated thresholds, which are ultimately based on the researchers' eyeballing and judgment, the details of which are rarely documented. We make such procedures more explicit, systematic, and transparent, so that the overall scheme becomes more reproducible. Second, human recognition is central to the prominence of Edgeworth cycles as an antitrust topic. Despite the lack of universal definitions, the phenomena have become a perennial policy issue in many countries precisely because consumers and politicians can easily recognize cyclical patterns when they see them. In this regard, human recognition *is* the "ground truth" that eventually determines the phenomena's relevance to public policy. We interpret our RAs' responses as a proxy for the general public's responses to various patterns in gasoline prices.

We report three sets of results. First, when applied to the two Australian

[5] See [31] for a guide to the Australian data. [70, 96, 15] among others, study the German data.

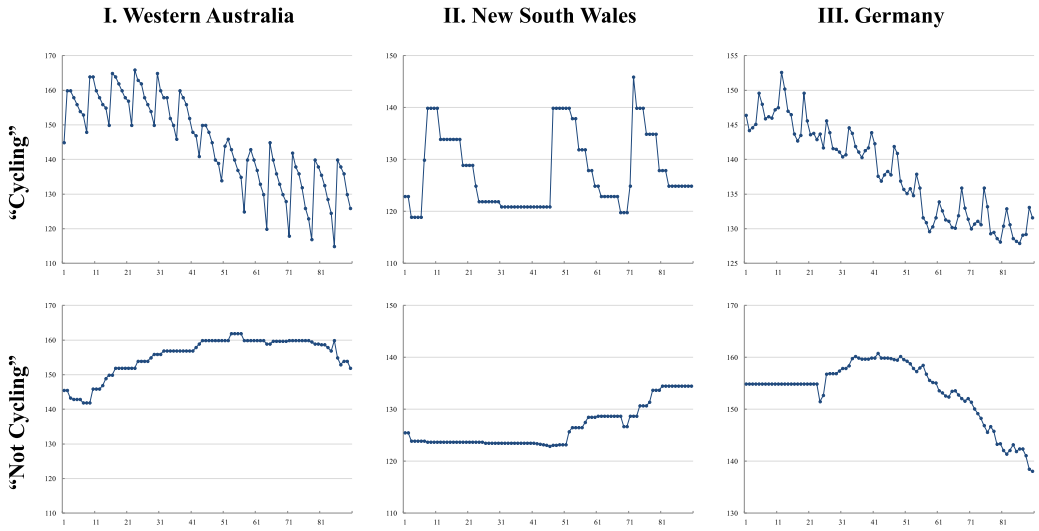


Figure 4.1: *Examples of Cycling and Non-cycling Station-Quarter Observations*

datasets, most of the methods—both existing and new—achieve high accuracy levels near or above 90% and 80%, respectively, because price cycles are clearly asymmetric and exhibit regular periodicity (hence, are easy to detect) in these regions. By contrast, German cycles are more subtle and diverse, defying many methods. All existing methods except Method 4 fail to detect cycles, even though as much as 40% of the sample is unanimously labeled as "cycling" by three RAs (see Figure 4.1 for examples). This failure is not an artifact of sample selection or human error because our interview with a German industry expert suggests Edgeworth cycles are known to exist. They are (in fact) called the "price parachute" (or *Preis Fallschirm*) phenomena, and are considered to be part of common pricing strategies among practitioners. The Bundeskartellamt (2011) [26] also confirms the existence of both weekly and daily cycles. Methods 7–10 attain 71%–80% accuracy even in this challenging environment.

Second, we assess the cost effectiveness of each method by using only 0.1% , 1%, 5%, 10%, $\cdots$, 80% of our manually labeled subsamples as "training" data. Results suggest simpler models (Methods 1–7) are extremely "cheap" to train, as they quickly approach their respective maximal accuracy with only a dozen observations. The nonparametric models (Methods 8–10) need more data to achieve near-maximal performance, but their data requirement is sufficiently small for practical purposes. Only a few hundred observations prove sufficient for even the most complex model (Method 10). The economic cost of manually classifying a few hundred observations is in the order of tens of RA hours, or a few

hundred US dollars at the current hourly wage of US$13.50 for undergraduate RA work at Yale University. Potential cost savings are sizable, as manually labeling the entire German dataset in 2014–2020 would require 4,800 RA hours, or US$64,800. Thus, our approach is economical and suitable for researchers and governments with limited resources.

Third, we investigate whether and how gasoline stations' markups are correlated with the presence of cycles. In WA and NSW, the average margins in (manually classified) "cycling" station-quarters are statistically significantly higher than in "non-cycling" ones. The relationship is reversed in Germany, where the margins in "cycling" observations are lower than in "non-cycling" ones. Hence, in general, the presence of cycles could be either positively or negatively correlated with markups. All of the automatic detection methods lead to the correct finding (i.e., positive correlations) in WA, but some of them fail in NSW. Furthermore, Methods 1–6 either fail to detect cycles or lead to false conclusions in Germany (i.e., find statistically significant *positive* correlations). This finding emerges under both "cyclicality only" and "cyclicality with asymmetry" definitions of Edgeworth cycles. Thus, whether researchers discover a positive, negative, or no statistical relationship between markups and cycles—a piece of highly policy-relevant empirical evidence—depends on the seemingly innocuous choice of operational definitions.

4.1.1 Related Literature, Contributions, and Replication Package.

Besides the contributions specific to the phenomena, our broader contribution is three-fold: (i) introducing certain "heavy-duty" machine-learning models and methods (a class of deep-neural-network architectures) to the empirical economics literature, (ii) precisely explaining the mechanisms inside these "black boxes", and (iii) demonstrating their usefulness with a concrete, public-policy-relevant example.

For the purpose of lowering the "entry barriers" for those empirical economists who are considering the use of advanced machine-learning tools, we have made the computer code (in Python), the dataset, and detailed documentations (including the read-me file and the Online Appendix) publicly available as a replication package on Github [72].

4.2 Theoretical Background

Even though the primary goal of this article is empirical, some conceptual anchoring clarifies the target of measurement.

4.2.1 What Are Edgeworth Cycles?

Maskin and Tirole (1988) [97] offer the following verbal description: "In the Edgeworth cycle story, firms undercut each other successively to increase their market share (price war phase) until the war becomes too costly, at which point some firm increases its price. The other firms then follow suit (relenting phase), after which price cutting begins again. The market price thus evolves in cycles" (pages 571–572). This description and its micro foundation—as a class of Markov perfect equilibria (MPE) in an alternating-move dynamic duopoly game—suggest four important characteristics: cyclicality, asymmetry, stochasticity, and strategicness.

Property 1: Cyclicality. The price should exhibit cyclicality, as the terminology suggests. However, this property is not so obvious in Edgeworth's (1925) [49] original conjecture. His writing focuses on the indeterminacy of static equilibrium in a price-setting game between capacity-constrained duopolists. Even though he mentions a price path that resembles Maskin and Tirole's description as an example, he uses the word "cycle" only once. More generally, he conjectures that "there will be an indeterminate tract through which the index of value will oscillate, or rather will vibrate irregularly for an indefinite length of time" (page 118). Thus, Edgeworth's original theory features not so much cyclicality as "perpetual motion" (page 121).

Nevertheless, we have chosen to focus on cyclicality in this paper. Theoretically, Maskin and Tirole's equilibrium strategies (their equation 23) explicitly feature price cycles. Empirically, it is this repetitive pattern that draws consumers' and politicians' attention; " perpetual motion" alone would not raise antitrust concerns.

Property 2: Asymmetry. The second characteristic is the asymmetry between relatively few large price increases and many small price decreases. Edgeworth (1925) [49] does not emphasize this property either, but it plays an important role in the Maskin-Tirole formalization and the subsequent empirical literature.

Property 3: Stochasticity. In Maskin and Tirole's Edgeworth-cycle MPE, big price increases are supposed to happen stochastically, not deterministically. The reason is that if one firm always "relents" whenever the low price is reached, the other firm will always wait and free-ride, which in turn would make the first firm more cautious about the timing of price increases. Thus, the frequency of cycles must be stochastic—with varying lengths of time spent at the low price—in equilibrium.[6] We do not impose stochastic frequencies as a necessary condition in our empirical procedures, but some of our methods are designed to accommodate cycles with varying frequencies (Methods 7 and 8).

Property 4: Strategicness. The cyclical patterns are supposed to emerge from dynamic strategic interactions between oligopolistic firms. If similar patterns are observed under monopoly, their underlying mechanism must be different from that of Edgeworth cycles.[7] Thus, whether market structure is monopolistic or oligopolistic is a theoretically important distinction. Empirically, however, market definition is rarely clear-cut in practice. Even when a gasoline station is located in a geographically isolated place, pricing decisions at large chains tend to be centralized at the city, region, or country level. Market structure at these aggregate levels is oligopolistic in all of our datasets. Consequently, we do not impose any geographical boundaries a priori. We simply analyze data at the individual station level.[8] Our idea is that once the station-level characterization is successfully completed, one can always compare cyclicality across stations in the same market (defined geographically or otherwise) and look for synchronicity—whenever such analysis becomes necessary.

[6] This theoretical property seems largely overlooked in the empirical literature, presumably because the first two properties make the phenomena sufficiently interesting and policy-relevant.

[7] Alternative explanations include consumers with heterogeneous search costs, intertemporal price discrimination, and "dynamic pricing" algorithms (broadly defined as any pricing strategy and its implementation(s) that tries to exploit consumer heterogeneity and time-varying price-elasticity of demand).

[8] This operational decision is not without its own risks. For example, if the grid of relevant prices were very coarse and two firms take turns to change prices, we might not be able to observe clear cycles at any specific station's time-series data even if such cycles exist at the aggregate level. Fortunately, gasoline prices reside on a relatively fine grid with the minimum interval of the Australian or Euro cent. Moreover, Maskin and Tirole's Edgeworth-cycle MPE requires a fine grid with sufficiently small intervals (denoted by k in their model). Therefore, we believe the risk of missing aggregate cycles is low.

4.2.2 Are Edgeworth Cycles Competitive or Collusive?

Whether Edgeworth cycles represent collusion is a subtle issue on which we do not take a stand. Several reasons contribute to its subtlety and our cautious attitude.

First, the theoretical literature seems agnostic about the distinction between competitive and collusive behaviors in the current context. On the one hand, Edgeworth's (1925) [49] narrative lacks any hint of cooperative actions or intentions. On the other hand, Maskin and Tirole (1988) [97] seem open to collusive interpretations: "Several of the results of this paper underscore the relatively high profits that firms can earn when the discount factor is near 1. Thus our model can be viewed as a theory of tacit collusion" (page 592). In the more recent literature, however, the term "tacit collusion" is usually associated with collusive equilibria in repeated-games models.[9] The latter rely on the concepts of monitoring, punishment, and history-dependent strategies as their underlying mechanism, none of which are prominently featured in Edgeworth cycles. Thus, even though Maskin and Tirole's own remarks suggest the possibility of collusive interpretations, we feel inclined to regard their Edgeworth-cycle MPE as a reflection of competitive interaction between forward-looking oligopolists.

Second, in terms of antitrust law, explicit communications of a cooperative nature are the single most important act that constitutes criminal price-fixing. That is, tacit collusion is not illegal as long as it truly lacks explicit communication. Notwithstanding this legal distinction, most of the theoretical literature does not discriminate between tacit and explicit collusion because the process through which firms reach collusive agreements is usually not modeled. Hence, an important gap lies between economic theory and legal enforcement, which complicates the interpretation of Edgeworth cycles in empirical research.

Third, partly reflecting this unresolved theory-enforcement divide, the empirical literature has documented many different instances of asymmetric price cycles, both *with* and *without* legally established evidence of criminal price-fixing. Accordingly, interpretations of observed cycles vary across papers on a case-by-case basis. The only common thread that unites the large empirical literature is the data patterns with clear cyclicality and asymmetry.

For these reasons, we do not (necessarily) interpret Edgeworth cycles as evidence of collusion. Consequently, we do not aim or claim to detect "collusion." Reliable methods to detect price cycles would nevertheless be useful for detecting

[9] Tirole and his coauthors exclusively focus on the repeated-games theory when they summarize the "economics of tacit collusion" for the European competition authority. See Ivaldi, Jullien, Rey, Seabright, and Tirole (2003) [76].

cycle-based collusion.

4.3 Data and Manual Classification

Retail-price data are publicly available for the universe of individual gasoline stations in WA, NSW, and Germany. We combine them with wholesale-price data, based on the region of each station (Australia) or the location of the nearest refinery (Germany). We compute station-level daily profit margins by subtracting the relevant wholesale price from the retail price,

$$p_{i,d} \equiv p^R_{i,d} - p^W_{i,d}, \tag{4.1}$$

where $p^R_{i,d}$ and $p^W_{i,d}$ are retail and wholesale prices at station i on day d, and simply refer to this markup measure $p_{i,d}$ as "price" in the following. We organize these daily prices by calendar quarter, so that station-quarter (i.e., a sequence of daily prices over 90 days for each station) becomes the unit of observation for cycle detection.

4.3.1 Data Sources and Preparation

Retail Prices. We use three datasets on retail gasoline prices that are publicly available and of high quality. FuelWatch and FuelCheck are legislated retail-fuel-price platforms operated by the state governments of WA and NSW, respectively. Their websites display real-time information on petrol prices, and the complete datasets can be downloaded. The Market Transparency Unit for Fuels of the Bundeskartellamt publishes similar data for every German gas station in minute intervals.

Sampling Frequencies. The raw data from WA contain daily retail prices for each station, which is the most granular level in this region because its law mandates each station must commit to a fixed price level for 24 hours. By contrast, the stations in NSW and Germany can change prices at any point in time, which we aggregate into daily prices by taking either end-of-day values (NSW) or intra-day averages (Germany). Intra-day changes are relatively rare in NSW, and hence, end-of-day values are representative of the actual transaction prices. In Germany, many stations change prices multiple times during the day, so we sample 24 hourly prices and take their average for each station-day (see section 3.3 for further details on Germany).

Wholesale Prices. The Australian Institute of Petroleum publishes average regional wholesale prices. The Argus Media group's *OMR Oil Market Report* collects daily regional wholesale prices and offers the database on a commercial basis.[10]

4.3.2 Manual-Classification Procedures

Whereas most existing studies treat the manual-verification process as an informal preparatory step (to be embodied by the analyst's eventual choice of methods and calibration of threshold parameters), we make it as systematic as possible. Our goal is to develop and compare the performance of multiple methods, and such "horse racing" requires a common benchmark.

To establish a "ground truth" based on human recognition of cycles, we employed a team of eight RAs to manually classify station-quarter observations.[11] Each station i in quarter t is classified as either "cycling," "maybe cycling," or " not cycling." The total number of manually labeled observations is 24,569 (WA), 9,693 (NSW), and 35,685 (Germany). The RAs' total working hours are approximately 260 (WA), 210 (NSW), and 480 (Germany). The manual labeling of the datasets proceeded in three stages.

WA. First, we labeled all station-quarters in the WA data with two RAs as a pilot project between July 2019 and June 2020. The first RA (a PhD student in economics) laid the ground work with approximately half of the WA data in close communication with one of the coauthors (Igami). The second RA (a senior undergraduate student majoring in economics) followed these examples to label the rest. Then, the first RA carefully double-checked all labels to maintain consistency. As a result, each station-quarter (i, t) in WA has one label based on the consensus of the two RAs.

NSW. Second, the NSW dataset is smaller but contains more ambiguous cases. Hence, we took a more organized/computerized approach by building a cloud-based computational platform to streamline the labeling process. The same coauthor manually labeled a random sample of 100 station-quarters in December 2020, which is used for generating automated training sessions for three new undergraduate RAs (a senior and a junior majoring in economics, and a junior

[10] Regional wholesale prices are the most detailed publicly available information on the operating costs of retail gasoline stations (to our knowledge). We do not observe station-specific costs.

[11] All of them are graduate or undergraduate students majoring in economics, mathematics, and statistics at Yale University.

mathematics major). In the automated training sessions, each of the three RAs was asked to classify random subsamples of the labeled observations, and to repeat the labeling practice until their judgments agreed with the coauthor's at least 80% of the time. Subsequently, each of the RAs independently labeled the entire dataset in February–April 2021. Thus, each (i, t) in NSW carries three labels.

Germany. Third, the same team of three RAs proceeded to label a 5% random sample of the German dataset in April–June 2021. In turn, these labels served as a source of "training sample" for yet another team of three RAs (two juniors majoring in economics and a freshman in statistics and data science). They labeled an additional 5% random sample in June 2021. In total, 10% of the German data is triple-labeled.

Risk of "Collusion" Is Low. In the computerized procedures for NSW and Germany, each RA is given *one randomly selected observation for labeling* at a time. We believe the risk of "collusion" among RAs is low because copying each other's answers would require (i) keeping records of random sequences of thousands of observations with their station-quarter identifiers, (ii) exchanging these long records, and (iii) matching each other's answers across different random sequences. Such a conspiracy is conceivable in principle but prohibitively time-consuming in practice. Honestly labeling all observations just once would be much easier.

Summary Statistics. Table 4.1 reports summary statistics. Based on these manual-classification results, we define $cycle_{i,t}$ as a binary variable indicating the presence of clear cycles. In WA, each observation is labeled exactly once, based on the consensus of two RAs. We set $cycle_{i,t} = 1$ if station-quarter (i, t) is labeled as "cycling," and 0 otherwise. In the NSW and German data, which contain more ambiguous patterns, we assigned three RAs to label each observation individually, and hence each (i, t) is triple-labeled. We set $cycle_{i,t} = 1$ for observations with triple "cycling" labels (i.e., based on three RAs' unanimous decisions), and 0 otherwise. Thus, we prepare the target for automatic detection in a relatively conservative manner.

Table 4.1: *Summary Statistics*

	(1)	(2)	(3)
Dataset	Western Australia	New South Wales	Germany
Sample period (yyyy/mm/dd)	2001/1 – 2020/6	2016/8 – 2020/7	2014/6 – 2020/1
Number of gasoline stations	821	1,226	14,780
Number of calendar quarters	77	15	26
Number of station-quarters	25,463	9,693	353,086
Of which:			
Labeled as "cycling" by 3 RAs	0 (0.0%)	6,878 (71.0%)	14,116 (39.6%)
Labeled as "cycling" by 2 RAs	0 (0.0%)	906 (9.4%)	7,173 (20.1%)
Labeled as "cycling" by 1 RA	15,007 (61.1%)	759 (7.8%)	6,280 (17.6%)
Not labeled as "cycling" by any RA	9,562 (38.9%)	1,150 (11.9%)	8,116 (22.7%)
Total manually labeled	24,569 (100.0%)	9,693 (100.0%)	35,685 (100.0%)
Not manually labeled	894	0	317,401

Note: Each "manually labeled" station-quarter observation in the WA data is single-labeled as either "cycling," "maybe cycling," or "not cycling," whereas the NSW and German data are triple-labeled.

4.3.3 Rationale for Daily Frequency and Quarterly Window

Several considerations led us to use the daily sampling frequency and the quarterly time window.

First, we prioritize setting a common time frame for all three datasets. Our goal is to compare the performance of various methods in multiple different datasets under the same protocol; a detailed case study of any single region/country is not our main objective. The daily frequency is the finest granularity that can be commonly used across all datasets because retail prices in WA are fixed for 24 hours due to regulation (see section 3.1). It is also the finest granularity used in most other studies (however, see below for our discussion of the German data).

Second, cyclicality implies repetition, the identification of which requires a sufficiently long time window. The existing studies on WA and NSW report cycles with frequencies of one to several weeks, whereas those on Germany report both weekly and intra-day cycles. The 12–13 weeks of a calendar quarter permit repeated observations of relatively long (e.g., monthly) cycles.

Third, shorter-than-daily (e.g., hourly) frequencies would be too "costly" for our research design, as systematic manual verification is its essential component. Eyeballing and labeling a 10% subsample of the entire German dataset at the hourly (instead of daily) frequency would require 24 times more labor: 480 hours $\times$ 24 = 11,520 hours. At the hourly wage of \$13.50, the total cost would be \$155,520.

Fourth, we avoid longer-than-quarterly time windows for two reasons. One is

that macroeconomic factors (such as business cycles, financial crises, and geopolitical upheavals in the world crude oil market) tend to feature prominently in a time horizon longer than 90 days, which increases noise. Another reason is that longer windows tend to complicate classification, as cycles might appear in only one part of the graph but not others.

For these reasons, the daily frequency and the quarterly horizon are suitable for our purposes. Note that our choice is driven by the comparative research design, practical considerations, and budget constraints, not conceptual limitations. All of the methods can be applied to time-series data of any frequency and length in principle.

On Intra-Day Cycles in the German Data. We are aware of multiple studies that document intra-day price cycles in Germany. The first investigation into the German retail fuel markets by Bundeskartellamt (2011) [26] studies data from four major cities (Hamburg, Leipzig, Cologne, and Munich) in January 2007–June 2010 and highlights three patterns. First, weekly cycles exist in both diesel and gasoline prices, with the highest prices on Fridays and the lowest prices on Sundays and Mondays. Second, intra-day cycles exist as well, with many small price reductions during the day and fewer, larger increases in the evening. Third, stations operated by Aral (BP) and Shell typically lead those price increases, in which one follows the other within three hours in 90% of the cases, followed by three other major chains.

Given the well-documented presence of intra-day cycles, one might wonder whether our focus on the daily data and multi-day cycles leads to an important omission. Our answer is "yes," but this issue is orthogonal to the main purpose of this research.

By aggregating the underlying minute-by-minute data to 24-hour averages, we lose these interesting short-run movements. Our choice of the daily frequency is driven by the comparative design of our research, which prioritizes the systematic comparisons across the three datasets and (costly) manual verification. Thus, researchers who wish to conduct an in-depth case study of the German fuel markets might want to analyze intra-day patterns as well.

Nevertheless, the presence of shorter cycles does not preclude that of longer cycles; as [26] confirms the existence of both (see above). One should also note that the intra-day cycles seem to follow a specific time schedule in which prices (i) rapidly increase at night between 20:00 and 24:00 hours and (ii) gradually decrease from around 6:00 in the following morning [124]. Such a deterministic pattern is more consistent with intertemporal price discrimination than Maskin

and Tirole's Edgeworth cycles [94]. Hence, while interesting, the intra-day cycles in Germany are outside the scope of this paper.

4.4 Models and Methods for Automatic Detection

This section explains (i) how we formalize the four existing methods, (ii) the six new methods that we propose, and (iii) the way we optimize the parameter values of each model.

4.4.1 Existing Methods Mostly Focus on Asymmetry

The existing methods in the literature almost exclusively focus on asymmetry. We formalize four of them as simple parametric models.

Method 1: Positive Runs vs. Negative Runs (" PRNR").

Castanias and Johnson (1993) [35] compare the lengths of positive and negative changes. We formalize this idea by classifying each station-quarter as cycling ($cycle_{i,t} = 1$) if and only if

$$mean\left(len\left(run^{+}\right)\right) < mean\left(len\left(run^{-}\right)\right) + \theta^{PRNR}, \tag{4.2}$$

where $len\left(run^{+}\right)$ and $len\left(run^{-}\right)$ denote the lengths of consecutive (multi-day) price increases/zero changes and decreases within quarter t, respectively. The means are taken over these "runs." $\theta^{PRNR} \approx 0$ is a scalar threshold, which we treat as a parameter.[12]

Method 2: Mean Increase vs. Mean Decrease (" MIMD").

Eckert (2002) [47] compares the magnitude of the mean increase and the mean decrease. Formally, station-quarter (i, t) is cycling if and only if

$$\left|mean_{d\in t}\left(\Delta p^{+}_{i,d}\right)\right| > \left|mean_{d\in t}\left(\Delta p^{-}_{i,d}\right)\right| + \theta^{MIMD}, \tag{4.3}$$

where $\Delta p^{+}_{i,d}$ and $\Delta p^{-}_{i,d}$ denote positive and negative daily price changes at station i (between days d and $d-1$), respectively, and $\theta^{MIMD} \approx 0$ is a scalar threshold.

[12] Eckert (2002) [47] proposes a more comprehensive version of this idea, which compares the *distributions* of positive and negative runs across lengths, by using the Kolmogorov-Smirnov test.

That is, a cycle is detected when the average price increase is greater than the average price decrease.[13]

Method 3: Negative Median Change (" NMC").

Lewis (2009) classifies $cycle_{i,t} = 1$ if and only if

$$median_{d \in t}\left(\Delta p_{i,d}\right) < \theta^{NMC}, \tag{4.4}$$

where $\Delta p_{i,d}$ denotes a price change between days d and $d-1$, and $\theta^{NMC} \approx 0$ is a scalar threshold. In other words, the significantly negative median change is taken as evidence of price cycles.[14]

Method 4: Many Big Price Increases (" MBPI").

Byrne and de Roos (2019) [30] identify price cycles with the condition

$$\sum_{d \in t} \mathbb{I}\left\{\Delta p_{i,d} > \theta_1^{MBPI}\right\} \geq \theta_2^{MBPI}, \tag{4.5}$$

where $\mathbb{I}\{\cdot\}$ is an indicator function that equals 1 if the condition inside the bracket is satisfied, and 0 otherwise. θ_1^{MBPI} and θ_2^{MBPI} are thresholds for "big" and " many" price increases, respectively. They set $\theta_1^{MBPI} = 6$ (Australian cents/liter) and $\theta_2^{MBPI} = 3.75$ (per quarter) in studying the WA data.[15] Thus, many instances of big price increases are taken as evidence of price cycles.

Other Existing Methods. These methods are among the most cited in the literature, but our listing is not exhaustive. Other influential papers use a variety of methods. Let us briefly discuss three of them. First, Noel (2007) [102] proposes a Markov switching model with three unobserved states, two of which correspond to positive and negative runs, respectively, and the third corresponds to a non-cyclical regime.[16] Second, Deltas (2008) [43] and many others regress re-

[13] Eckert (2003) [47] uses this method as well. Clark and Houde (2014) [40] propose its variant: the ratio of the median price increase to the median price decrease, with 2 as a threshold to define cyclical subsamples.

[14] Many subsequent studies use this method [140, 46, 91, 93, 48, 150, 30]. As a threshold for discretization, [93] uses −0.2 US cents per gallon, whereas [46, 150] use −0.5 US cents per gallon.

[15] Lewis (2009) [92] also uses a similar method, with $\theta_1^{MBPI} = 4$ (US cents/gallon) in a single day or two consecutive days.

[16] Because these states are modeled as unobserved objects, using this approach as a definition is not straightforward. Zimmerman et al. (2013) [150] propose another definition that shares

tail price on wholesale price to describe asymmetric responses. Third, Foros and Steen (2013) [55] regress price on days-of-week dummies to describe weekly cycles. These papers offer valuable insights, and their methods are suitable in their respective contexts. However, they are not specifically designed for defining or detecting cycles.

4.4.2 Our Proposals to Capture Cyclicality

We propose six new methods. Methods 5–6 are based on spectral analysis, and hence are attractive as formal mathematical definitions of regular cycles. By contrast, Methods 7–8 build on nonparametric regressions and machine-learning techniques, respectively, and are more suitable for capturing nuanced patterns and replicating human recognition of cycles. Methods 9–10 combine some or all of the previous methods.

This subsection is rather technical because we are introducing data-analysis techniques from outside the usual toolbox of empirical economists. If the reader is not interested in methodological details, a quick look at the first and the last few sentences of each method would be sufficient for an overview. If, instead, the reader wants to exactly follow our procedures, Appendix A.1 (and the replication package) provides additional details.

Method 5: Fourier Transform ("FT").

Fourier analysis is a mathematical method for detecting and characterizing periodicity in time-series data. When a continuous function of time $g(x)$ is sampled at regular time intervals with spacing Δx, the sample analog of the Fourier power spectrum (or " periodogram") is

$$P(f) \equiv \frac{1}{N}\left|\sum_{n=1}^{N} g_n e^{-2\pi i f x_n}\right|^2, \tag{4.6}$$

where f is frequency, N is the sample size, $g_n \equiv g(n\Delta x)$, $i \equiv \sqrt{-1}$ is the imaginary unit (not to be confused with our gas-station index), and x_n is the time

the spirit of Markov switching regressions: (i) Compare the probability that a price increase (decrease) is observed after two consecutive price increases (decreases); and (ii) if the conditional probability of a third consecutive increase is smaller than that of a third decrease, take it as an indicator of cycles. We regard their approach as a variant of Castanias and Johnson's method. Finally, Noel (2018) [104] defines the relenting and undercutting phases by consecutive days with cumulative increases and decreases of at least 3 Australian cents per liter, respectively, which is also close to Castanias and Johnson's (1993) [35] idea.

stamp of the n-th observation. It is a positive, real-valued function that quantifies the contribution of each frequency f to the time-series data $(g_n)_{n=1}^N$.[17]

We focus on the highest point of $P(f)$ and detect cycles if and only if

$$\max_f P_{i,t}(f) > \theta_{\max}^{FT}, \tag{4.7}$$

where $P_{i,t}(f)$ is the periodogram (4.6) of station-quarter (i,t), and $\theta_{\max}^{FT} > 0$ is a scalar threshold parameter.

Method 6: Lomb-Scargle ("LS") Periodogram.

The LS periodogram (Lomb 1976, Scargle 1982) characterizes periodicity in unevenly sampled time series.[18] It has been extensively used in astrophysics because astronomical observations are subject to weather conditions and diurnal, lunar, or seasonal cycles. Formally, it is a generalized version of the classical periodogram (4.6):[19]

$$P^{LS}(f) = \frac{1}{2}\left\{\frac{\left(\sum_n g_n \cos\left(2\pi f\left[x_n-\tau\right]\right)\right)^2}{\sum_n \cos^2\left(2\pi f\left[x_n-\tau\right]\right)} + \frac{\left(\sum_n g_n \sin\left(2\pi f\left[x_n-\tau\right]\right)\right)^2}{\sum_n \sin^2\left(2\pi f\left[x_n-\tau\right]\right)}\right\}, \tag{4.8}$$

where τ is specified for each frequency f as

$$\tau = \frac{1}{4\pi f}\tan^{-1}\left(\frac{\sum_n \sin\left(4\pi f x_n\right)}{\sum_n \cos\left(4\pi f x_n\right)}\right). \tag{4.9}$$

We propose the following threshold condition to detect cycles:

$$\max_f P_{i,t}^{LS}(f) > \theta_{\max}^{LS}, \tag{4.10}$$

where $\theta_{\max}^{LS} > 0$ is a scalar threshold parameter.

[17] Appendix A.1 (Method 5) introduces FT to readers who are not familiar with Fourier analysis.

[18] Our data are evenly sampled at the daily frequency and can be analyzed by FT alone, but the LS periodogram offers additional benefits. One is conceptual: it is interpretable as a kind of nonparametric regression—see Appendix A.1 (Method 6). Another is practical: its off-the-shelf computational implementation can offer more granular periodograms.

[19] Appendix A.1 (Method 6) explains how this expression relates to FT.

Method 7: Cubic Splines ("CS").

This method captures cycles' frequency in a less structured manner than FT and LS by using cubic splines (a spline is a piecewise polynomial function). That is, we smooth the discrete (daily) time series by interpolating it with a commonly used continuous function.[20] For each (i,t), we fit CS to its demeaned price series, $\overline{p}_{i,d} \equiv p_{i,d} - mean_{d\in t}\left(p_{i,d}\right)$, and count the number of times the fitted function $\overline{CS}_{i,t}(d)$ crosses the d -axis (i.e., equals 0). Operationally, we count the number of real roots and detect cycles with the condition,

$$\#roots\left(\overline{CS}_{i,t}(d)\right) > \theta^{CS}_{root}, \tag{4.11}$$

where $\theta^{CS}_{root} > 0$ is a scalar parameter. Thus, any frequent oscillations (not limited to the sinusoidal ones as in FT or LS) become a sign of cycles.

Method 8: Long Short-Term Memory ("LSTM").

Recurrent neural networks with LSTM (Hochreiter and Schmidhuber 1997) [71] are a class of artificial neural network (ANN) models for sequential data. LSTM networks have become a "de-facto standard" for recognizing and predicting complicated patterns in many applications, including speech, handwriting, language, and polyphonic music. Because LSTM is relatively new, we explain this method in greater detail.

Econometrically speaking, LSTM is a nonparametric model for time-series analysis. It is a recursive dynamic model whose behavior centers on a collection of pairs of $B_l \times 1$ vector-valued latent state variables, $\mathbf{s}^l_d$ and $\mathbf{c}^l_d$, where $l = 1, 2, \cdots, L$ is an index of layers. As this notation suggests, we use a multi-layer architecture (a.k.a. "deep" neural networks) to enhance the model's flexibility.[21] B_l represents the number of blocks per layer, which are analogous to "neurons" (basic computing units) in other ANN models. $\mathbf{s}^l_d$ is an output state that represents the current, "short-term" state, whereas $\mathbf{c}^l_d$ is called a cell state and retains "long-term memory." The latter is designed to capture lagged dependence between the state and input variables, thereby playing the role of a memory cell in electronic computers.

[20] We use a cubic Hermite interpolater, which is a spline where each piece is a third-degree polynomial of Hermite form. Appendix A.1 (Method 7) explains the details of this functional form.

[21] Except for the multi-layer design, our specification mostly follows Greff, Srivastava, Koutník, Steunebrink, and Schmidhuber (2017) [60], in which one of the original proponents of LSTM and his team compare many of its variants and show that their simple "vanilla" specification outperforms others.

These state variables evolve according to the following Markov process:

$$\mathbf{s}_d^l = \underbrace{\tanh\left(\mathbf{c}_d^l\right)}_{\text{“output”}} \circ \underbrace{\Lambda\left(\omega_1^l + \omega_2^l \Delta p_d + \omega_3^l \mathbf{s}_d^{l-1}\right)}_{\text{“output gate”}}, \text{ and} \tag{4.12}$$

$$\mathbf{c}_d^l = \underbrace{\tanh\left(\omega_4^l + \omega_5^l \Delta p_d + \omega_6^l \mathbf{s}_d^{l-1}\right)}_{\text{“input”}} \circ \underbrace{\Lambda\left(\omega_7^l + \omega_8^l \Delta p_d + \omega_9^l \mathbf{s}_d^{l-1}\right)}_{\text{“input gate”}} + \mathbf{c}_d^{l-1} \circ \underbrace{\left[1 - \Lambda\left(\omega_7^l + \omega_8^l \Delta p_d + \omega_9^l \mathbf{s}_d^{l-1}\right)\right]}_{\text{“ forget gate”}}, \tag{4.13}$$

where $d = 1, 2, \cdots, D$ is our index of days, $\Delta p_d \equiv p_d - p_{d-1}$ (we set $\Delta p_1 = 0$), $\tanh(x) \equiv \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is the hyperbolic tangent function, $\circ$ denotes the Hadamard (element-wise) product, and $\Lambda(x) \equiv \frac{e^x}{1+e^x}$ is the cumulative distribution function (CDF) of the logistic distribution.[22] The ωs are weight parameters with the following dimensionality: (i) ω_1^l, ω_2^l, ω_4^l, ω_5^l , ω_7^l, and ω_8^l are $B_l \times 1$ vectors; and (ii) ω_3^l, ω_6^l, and ω_9^l are $B_l \times B_{l-1}$ matrices. Thus, $\mathbf{B} \equiv (B_1, B_2, \cdots, B_L)$ determines the effective number of latent state variables and parameters, and hence the flexibility of the model.

The first layer $l = 1$ of time d takes as input the states of the last layer $l = L$ of time $d - 1$. Thus, $\left(\mathbf{s}_d^{l-1}, \mathbf{c}_d^{l-1}, B_{l-1}\right)$ in the above should be replaced by $\left(\mathbf{s}_{d-1}^L, \mathbf{c}_{d-1}^L, B_L\right)$ when $l = 1$. After the final layer L of the last day $D = 90$ of quarter t, we detect cycles in station-quarter (i, t) if and only if

$$s^*\left(\mathbf{p}_{i,t}; \theta^{LSTM}\right) \equiv \omega_{10} + \omega_{11}' \mathbf{s}_D^L > 0, \tag{4.14}$$

where ω_{10} is a scalar, ω_{11} is a $B_L \times 1$ vector, and $\theta^{LSTM} \equiv (\omega, L, \mathbf{B})$ collectively denotes all parameters, including (i) the many weights in $\omega \equiv \left(\left(\omega_1^l, \omega_2^l, \cdots, \omega_9^l\right)_{l=1}^L, \omega_{10}, \omega_{11}\right)$, (ii) the number of layers L, and (iii) the profile of the number of blocks in each layer, $\mathbf{B}$. We set $L = 3$ and $\mathbf{B} = (16, 8, 4)$, and find the value of ω that approximately maximizes the accuracy of prediction.

In summary, LSTM sequentially processes the daily price data in a flexible Markov model with many latent states, and uses the terminal state s^* as a latent score to detect cycles.

[22] See Appendix A.1 (Method 8) for further details on this specification and computational implementation.

Method 9: Ensemble in Random Forests ("E-RF").

This method combines Methods 1–7 within random forests (RF), which is a class of nonparametric regressions. Let

$$g_{i,t}^{m} \equiv LHS_{i,t}^{m} - RHS_{i,t}^{m} \tag{4.15}$$

denote a "gap," the scalar difference between the left-hand side (LHS) and the right-hand side (RHS) of the inequality that defines each method $m = 1, 2, \cdots, M$, excluding the threshold parameter, θ^{m}. For example, inequality (4.3) defines Method 2. Hence, $g_{i,t}^{2} = \left| mean_{d \in t}\left(\Delta p_{i,d}^{+}\right)\right| - \left| mean_{d \in t}\left(\Delta p_{i,d}^{-}\right)\right|$.[23]

Let

$$\mathbf{g}_{i,t} \equiv \left(g_{i,t}^{m}\right)_{m=1}^{M} \tag{4.16}$$

denote their vector, where $M = 7$.[24] We construct a decision-tree classification algorithm that takes $\mathbf{g}_{i,t}$ as inputs and predicts $cycle_{i,t} = 1$ if and only if

$$h\left(\mathbf{g}_{i,t}; \omega^{RF}, \kappa^{RF}\right) \equiv \sum_{k=1}^{K} \omega_{k}^{RF} \mathbb{I}\left\{\mathbf{g}_{i,t} \in R_{k}\right\} \equiv \sum_{k=1}^{K} \omega_{k}^{RF} \phi\left(\mathbf{g}_{i,t}; \kappa_{k}^{RF}\right) > 0, \tag{4.17}$$

where K is the number of adaptive basis functions, ω_{k}^{RF} is the weight of the k-th basis function, R_{k} is the k-th region in the M-dimensional space of $\mathbf{g}_{i,t}$, and κ_{k}^{RF} encodes both the choice of variables (elements of $\mathbf{g}_{i,t}$) and their threshold values that determine region R_{k}.[25] Because finding the truly optimal partitioning is a computationally difficult (combinatorial) problem, we use an RF algorithm to stochastically approximate it.[26] Thus, this method aggregates and generalizes Methods 1–7 in a flexible manner that permits (i) multiple thresholds and (ii) interactions between $g_{i,t}^{m}$s. We denote its full set of parameters by $\theta^{RF} \equiv \left(\omega^{RF}, \kappa^{RF}\right) \equiv \left(\left(\omega_{k}^{RF}\right)_{k=1}^{K}, \left(\kappa_{k}^{RF}\right)_{k=1}^{K}\right)$.

[23] All of Methods 1–7 except 4 are one-parameter models like this example. For Method 4, we define $g_{i,t}^{4} \equiv \sum_{d \in t} \mathbb{I}\left\{\Delta p_{i,d} > \theta_{1}^{MBPI*}\right\}$, where θ_{1}^{MBPI*} is the accuracy-maximizing value of θ_{1}^{MBPI}.

[24] Our computational implementation also incorporates two additional variants of each of Methods 5–7, which we explain in Appendix A.1 (Method 9). Hence, the eventual value of M is $7 + (2 \times 3) = 13$.

[25] See Murphy (2012, ch. 16) [99] for an introduction to adaptive basis-function models including RF.

[26] See Appendix A.1 (Method 9) for further details.

Method 10: Ensemble in LSTM ("E-LSTM").

This method combines Methods 1–8 within an extended LSTM by incorporating $\mathbf{g}_{i,t}$ in (4.16) as additional variables in the laws of motion:

$$\mathbf{s}_d^l = \tanh\left(\mathbf{c}_d^l\right) \circ \Lambda\left(\omega_1^l + \omega_2^l \Delta p_d + \omega_3^l \mathbf{s}_d^{l-1} + \omega_{12}^l \mathbf{g}\right), \text{ and} \tag{4.18}$$

$$\mathbf{c}_d^l = \tanh\left(\omega_4^l + \omega_5^l \Delta p_d + \omega_6^l \mathbf{s}_d^{l-1} + \omega_{13}^l \mathbf{g}\right) \circ \Lambda\left(\omega_7^l + \omega_8^l \Delta p_d + \omega_9^l \mathbf{s}_d^{l-1} + \omega_{14}^l \mathbf{g}\right) + \mathbf{c}_d^{l-1} \circ \left[1 - \Lambda\left(\omega_7^l + \omega_8^l \Delta p_d + \omega_9^l \mathbf{s}_d^{l-1} + \omega_{14}^l \mathbf{g}\right)\right], \tag{4.19}$$

where $\left(\omega_{12}^l, \omega_{13}^l, \omega_{14}^l\right)$ are $B_l \times M$ matrices of weight parameters for $\mathbf{g}_{i,t}$ (we suppress (i,t) subscript here). Other implementation details are the same as Method 8.

4.4.3 Optimization of Parameter Values ("Training")

Accuracy Maximization. Whereas the existing research typically calibrates (i.e., manually tunes) the threshold parameters, we optimize this process by choosing the parameter values that maximize accuracy, which we define as the percentage of correct predictions,

$$\%\ \text{correct}(\theta) \equiv \frac{\sum_{(i,t)} \mathbb{I}\left\{\widehat{cycle}_{i,t}(\theta) = cycle_{i,t}\right\}}{\#\ all\ predictions} \times 100, \tag{4.20}$$

where $\widehat{cycle}_{i,t}(\theta) \in \{0,1\}$ is the algorithmic prediction for observation (i,t) at parameter value θ, and $cycle_{i,t} \in \{0,1\}$ is the manual classification label (data). We analogously define two types of prediction errors, "false negative" and "false positive," in Appendix A.2. Thus,

$$\theta^* \equiv \arg\max_{\theta} \ \%\ \text{correct}(\theta) \tag{4.21}$$

characterizes the optimized (or "trained") model for each method.[27]

Splitting Data into Training and Testing Subsamples. We optimize and evaluate each method as follows, separately for each of the three datasets (WA, NSW, and Germany):

1. Randomly split each labeled dataset into an 80% "training" subsample and a 20% " testing" subsample.

[27] See Appendix A.2 for further details.

2. Optimize the parameter values of each model in the 80% training subsample.
3. Assess its "out-of-sample" prediction accuracy in the 20% testing subsample.[28]
4. Repeat these three steps 101 times.[29]
5. Report the medians of the optimized parameter values, as well as the medians and standard deviations of the prediction-accuracy results.

4.5 Results

Table 4.2 summarizes the performance of all methods for each dataset. We report the median accuracy, the composition of correct and incorrect predictions, and the associated parameter value(s), θ^*, for each method.

WA. Panel I shows the results in WA, where clear-cut cycles of deterministic frequencies are known to exist. Almost all methods achieve high accuracy near or above 90%. The flexible, nonparametric models of Methods 8–10 do particularly well with above 99% accuracy.

Some of the parameter values are informative about the underlying data patterns. For example, CS lags behind all other methods with (a still respectable) 85% accuracy. Its parameter value, $\theta^{CS}_{roots} = 22.5$, suggests the model is trained to focus on shorter cycles with wavelengths less than $90 \div \frac{22.5}{2} = 8$ days. Byrne and de Roos (2019) [30] show both weekly and two-weekly cycles exist in WA. Thus, the inferior performance of CS stems from missing the latter, longer cycles.

Another interesting result concerns MBPI, which achieves 90% accuracy. Byrne and de Roos (2019) [30] set $\theta^{MBPI}_1 = 6$ and $\theta^{MBPI}_2 = 3.75$ in their original study of WA. Our accuracy-maximizing values (5.05 and 5, respectively) turn out to be reasonably close to their calibrated values. This comparison illustrates how experienced researchers' parameter tuning could approximate the results of systematic numerical optimization. One can also interpret this finding as an external validation of our manual classification. Given the similar parameter values and the high accuracy, it follows that our manual classification must be broadly consistent with Byrne and de Roos's eyeballing results.

[28] This cross-validation procedure is particularly important for the nonparametric models of Methods 8–10, which contain many parameters and could potentially "over-fit" the training subsample.

[29] An odd number of bootstrap sample-splits facilitates the selection of the medians in step 5.

Table 4.2: *Performance of Automatic Detection Methods*

Method	(1) PRNR	(2) MIMD	(3) NMC	(4) MBPI	(5) FT	(6) LS	(7) CS	(8) LSTM	(9) E-RF	(10) E-LSTM
	I. Western Australia (# manually labeled observations: 24,569)									
Parameter 1	−1.16	6.13	−0.20	5.05	0.12	0.21	22.50	–	–	–
Parameter 2	–	–	–	5	–	–	–	–	–	–
Accuracy rank	5	4	9	6	8	7	10	1	3	1
% correct (median)	90.80	91.27	89.34	90.23	90.11	90.15	85.47	99.25	99.04	99.25
(Standard deviations)	(0.37)	(0.38)	(0.38)	(0.36)	(0.40)	(0.36)	(0.45)	(0.18)	(0.15)	(0.14)
of which cycling	55.27	55.70	57.08	60.74	58.24	57.92	56.41	60.62	60.97	60.34
of which not	35.53	35.57	32.25	29.49	31.87	32.23	29.06	38.62	38.07	38.91
% false negative	5.27	5.27	3.34	0.71	2.48	3.30	5.29	0.35	0.61	0.31
% false positive	3.93	3.46	7.33	9.06	7.41	6.55	9.24	0.41	0.35	0.45
	II. New South Wales (# manually labeled observations: 9,693)									
Parameter 1	4.20	5.76	1.01	14.90	0.20	0.57	4.50	–	–	–
Parameter 2	–	–	–	2	–	–	–	–	–	–
Accuracy rank	7	8	10	4	6	5	9	2	3	1
% correct (median)	78.55	78.39	70.96	81.59	80.71	80.82	73.90	89.63	87.42	90.30
(Standard deviations)	(0.85)	(0.88)	(0.97)	(0.86)	(0.80)	(0.80)	(0.89)	(0.67)	(0.69)	(0.67)
of which cycling	67.04	65.09	70.96	64.62	66.53	66.43	70.40	67.20	67.10	65.60
of which not	11.50	13.31	0.00	16.97	14.18	14.39	3.51	22.43	20.32	24.70
% false negative	3.30	4.85	0.00	6.55	5.47	4.02	0.77	4.33	8.35	2.99
% false positive	18.15	16.76	29.04	11.86	13.82	15.16	25.32	6.03	4.23	6.70
	III. Germany (# manually labeled observations: 35,685)									
Parameter 1	−3.48	0.30	−0.45	1.25	0.24	0.62	24.50	–	–	–
Parameter 2	–	–	–	14	–	–	–	–	–	–
Accuracy rank	9	6	7	5	8	10	4	3	2	1
% correct (median)	60.38	60.61	60.53	65.39	60.50	60.36	71.28	74.61	76.14	79.58
(Standard deviations)	(0.49)	(0.50)	(0.52)	(0.52)	(0.56)	(0.59)	(0.42)	(0.44)	(1.46)	(0.53)
of which cycling	0.00	1.25	0.07	14.77	0.00	0.00	25.88	23.46	23.96	29.96
of which not	60.38	59.37	60.46	50.62	60.50	60.36	45.40	51.16	52.18	49.63
% false negative	39.62	38.07	39.40	24.65	39.50	39.57	14.28	15.99	15.75	9.50
% false positive	0.00	1.32	0.07	9.96	0.00	0.07	14.45	9.40	8.11	10.91

Note: Appendix B.1 investigates whether combining some or all of Methods 1–4 leads to better performances. Appendix B.2 reports additional results for the variants of Methods 5–7. Columns (8)–(10) do not report parameter values because they contain too many parameters to be listed. We randomly split the sample into an 80% training subsample and a 20% testing subsample 101 times. In each split, the former subsample is used for setting parameter values, the medians of which are reported here. The accuracy statistics are also the medians from the 101 testing subsamples.

NSW. Panel II reports the results in NSW. Cycle detection in NSW is not as easy as in WA, but most methods achieve near or above 80% accuracy. The nonparametric methods are top performers again (87%–90%), followed by MBPI and the spectral methods (81%–82%). By contrast, CS (74%) and NMC (71%) make

mostly degenerate predictions in which they classify virtually all observations as cycles.

The poor performance of NMC is surprising in three ways. First, it performed well in WA. Second, it is one of the most widely used methods in the literature. Third, other methods that similarly focus on asymmetry (PRNR and MIMD) do significantly better (78%–79%). This finding alone does not necessarily invalidate the use of NMC in other datasets but cautions against overly relying on any single metric.

Germany. Panel III shows most methods fail in Germany, where cycles are more subtle and data are noisier (i.e., our RAs reach unanimous decisions less often). [30] E-LSTM is the only method that achieves accuracy near 80%, followed by E-RF (76%) and LSTM (75%). Somewhat surprisingly, CS (71%) outperforms all other parametric models; MBPI (65%) is the only existing method with non-degenerate predictions, presumably because it does not exclusively rely on asymmetry.

This profile of success and failure is intriguing. The methods that exclusively focus on asymmetry (Methods 1–3) and deterministic cycles (Methods 5–6) fail, whereas those that capture cyclicality in "fuzzier" manners (Methods 4 and 7) manage to make at least some correct (non-degenerate) predictions. These results suggest that not all of the German cycles conform to the idealized patterns of asymmetry or cyclicality and that less rigid classification rules could be relatively more robust to irregular patterns and noise.

The parameter values of CS ($\theta^{CS}_{roots} = 24.50$) and MBPI ($\theta^{MBPI}_{2} = 14$) suggest that the German cycles are approximately weekly. That is, $\theta^{CS}_{roots} = 24.50$ means at least as many ups and downs are often recorded in "cycling" observations, which translate into the wavelength of $90 \div \frac{24.5}{2} = 7.3$ days or shorter. Likewise, $\theta^{MBPI}_{2} = 14$ requires at least as many "big jumps" within a calendar quarter and hence implies the wavelength of $90 \div 14 = 6.4$ days or shorter. These numbers provide another opportunity for external validation: the detailed case study by Bundeskartellamt (2011) [26] confirms the presence of weekly cycles.

Summary. In summary, four findings emerge from Table 4.2. First, the four existing methods (Methods 1–4) work well in the clean data environments of Australia, but mostly fail in the noisier data from Germany. The spectral methods (Methods 5–6) show similar performance. Second, by contrast, CS (Method 7)

[30] As Table 4.1 shows, 71% of the NSW data is unanimously labeled as "cycling" by three RAs, whereas $9.4\% + 7.8\% = 17.2\%$ is labeled as such by only two or one RAs. In the German sample, only 39.6% is unanimously "cycling," whereas RAs disagree in $20.1\% + 17.6\% = 37.7\%$ of the data.

underperforms most other methods when cycles are clear and regular, but does relatively well in noisier cases. Third, LSTM (Method 8) is sufficiently flexible to capture both clear and noisy cycles: the most accurate stand-alone method. Fourth, the ensemble methods (Methods 9–10) effectively leverage the information content of Methods 1–8 and usually outperform all of them. The fact that E-RF performs so well is particularly interesting because it simply aggregates the descriptive statistics from Methods 1–7 in a more flexible manner (i.e., permitting their interactions and multiple thresholds).

Performance on Simulated Cycles. In Appendix A.3, we examine the 10 methods' performances on simulated data with four types of artificial patterns: white noise, theoretical Edgeworth cycles, "reverse Edgeworth" cycles, and sine waves of various lengths. We simulate 10,000 quarters of data based on each DGP and deploy the three pre-trained versions (WA, NSW, and Germany) of each method. Four findings emerge. First, Methods 1–4 and 7 either fail to detect most of these cycles or incorrectly classify white noise as cycles. Second, Methods 5–6 are the best performers in such a controlled environment. Third, the performances of Methods 8–10 are somewhere between these two groups of methods. Fourth, a little bit of additional noise could either help or hinder the performance of these 10 methods. These results suggest the real-world data are qualitatively different from simulated data with artificial cycles.

4.6 How Much Data Do We Need?

The accuracy "horse racing" in the previous section shows that more flexible methods tend to outperform simple parametric ones, which is not surprising. The real question is the cost of "training" complicated machine-learning algorithms, which are known to require a lot of data. This section investigates the cost-accuracy trade-offs of the 10 methods.

The accuracy of cycle detection naturally improves with the size of the training dataset. The rate of improvement is different across methods, however. Figure 4.2 shows performance when we restrict the training dataset to only 0.1%, 1%, 5%, 10%, $\cdots$, 80% of the available samples.

Methods 1–7 and 9 perform surprisingly well with only 0.1% of the data, which corresponds to 25, 10, and 36 observations in WA, NSW, and Germany, respectively. The labor cost of human-generated labels is negligible for such small samples (US$3.51, US$2.84, and US$6.48, respectively, based on the hourly

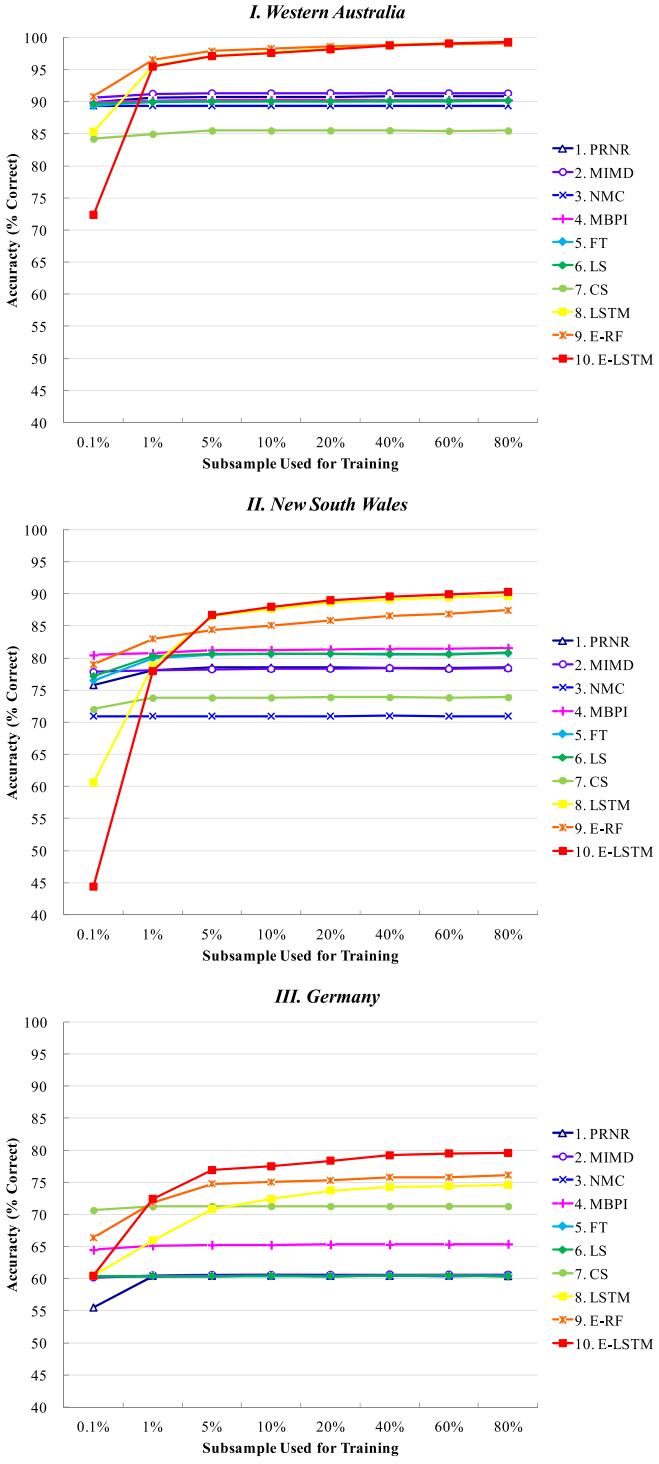


Figure 4.2: *Gains from Additional Data*

wage of US$13.50 for undergraduate RA work at Yale University as of 2021). These methods are extremely cost effective.

The fact that simple models with one or two parameters (Methods 1–7) require only a few dozen observations is not surprising. All we have to do is to adjust one or two numerical thresholds to distinguish cycles from non-cycles. However, the finding that E-RF (Method 9) is equally cheap *is* surprising. It is a highly nonlinear machine-learning model with potentially many thresholds and interactions. This result suggests that the building blocks of E-RF—the summary statistics derived from Methods 1–7—contain genuinely useful information that those stand-alone methods under-utilize.

Methods 8 and 10 contain a few thousand parameters and obviously need more data. For instance, E-LSTM's accuracy in NSW is below 50% when it uses only 10 observations (0.1% subsamples). Fortunately, their performance dramatically improves with a mere 1% subsample, and they start outperforming all other methods when 5% subsamples are used. The "critical" sample size above which they perform the best is in the order of several hundred observations. The associated cost of manual labeling is only tens of RA hours, or a few hundred US dollars. Thus, even though LSTM and E-LSTM require more data for a given accuracy level, their total cost is surprisingly low, making them the highest-accuracy methods within a limited amount of resources.

This finding is unexpected, but is definitely good news: heavy-duty machine-learning algorithms turn out to be not only useful, but also affordable in the context of detecting Edgeworth cycles. Our conjecture is that the cyclical patterns that humans recognize are relatively simple after all, even though explicitly articulating them might be difficult.

4.7 Economic and Policy Implications

The suspicion that price cycles might be related to collusive business practices has led many researchers and governments to collect and scrutinize large amounts of data on fuel markets. Some papers find that the presence of cycles is positively correlated with retail prices and markups, whereas others find the opposite relationships.[31]

[31] The former includes Deltas (2008) [43], Clark and Houde (2014)[40], and Byrne (2019)[29]; the latter includes Lewis (2009) [92], Zimmerman et al. (2013) [150], and Noel (2015) [103].

4.7.1 Cycles and Margins

Human-Recognized Cyclicality and Margins. Table 4.3 compares the retail-wholesale margins between "cycling" and " non-cycling" observations.[32] Column (0) is based on our manual classification and serves as a "ground truth" benchmark. The mean margins in cycling and non-cycling observations in WA are cents 11.86 and cents 9.47, respectively. The mean difference is cents 2.39. The t test (based on Welch's t statistic) rejects the null hypothesis that the difference in means is zero at the 0.1% significance level. Hence, price cycles are positively correlated with margins in WA. The same analysis yields similar results in NSW.

However, the pattern is reversed in Germany, where margins are *lower* in cycling station-quarters. Thus, in general, the presence of cycles (as recognized by human eyes) could be either positively or negatively correlated with margins, depending on regions/countries.[33]

Algorithmic Cycle Detection and Margins. Columns (1)–(10) report the same analysis based on the 10 algorithmic methods. In WA, all methods reach the same conclusion that margins are higher in cycling observations. Broadly similar results also emerge in NSW, even though one method fails (Method 3) and one reaches the opposite conclusion (Method 7). These discrepancies suggest that researchers find a positive or negative cycle-margin relationship depending on the operational definition of cycles.

Our analysis of the German data highlights this point even more vividly. Both the manual classification and Methods 7–10 suggest significantly negative relationships between cycles and margins, but Methods 2–6 lead to *positive* mean differences. These positive differences are highly statistically significant in Methods 2–4. Some of them entail degenerate predictions, but Method 4 features reasonable parameter values and achieves at least 65% accuracy. Hence, we cannot dismiss these discrepancies as purely random anomalies.

In summary, the choice of the detection method could lead to qualitatively different results and dictate the policy implications of empirical research on Edge-

[32] Our measure of profit margin is the difference between the retail price and the wholesale price before tax, as defined in equation (4.1), in the Australian cent in WA and NSW and the euro cent in Germany, respectively. Note the lack of volume data—a main limitation in this area of research—means that we cannot check the extent to which consumers buy at the bottom of price cycles.

[33] Determining the exact source of heterogeneity is beyond the scope of this paper. There can be many reasons and Edgeworth cycles are only one of the possible mechanisms. Our purpose is to illustrate with concrete examples how different methods could lead to different findings and policy implications.

Table 4.3: *Profit Margins by Cycle Status*

Method	(0) Manual	(1) PRNR	(2) MIMD	(3) NMC	(4) MBPI	(5) FT	(6) LS	(7) CS	(8) LSTM	(9) E-RF	(10) E-LSTM
I. Western Australia (# manually labeled observations: 24,569)											
Cycling											
# obs.	15,007	14,462	14,620	16,147	16,941	16,223	15,774	15,953	15,011	14,994	14,999
Mean	11.86	12.07	12.21	11.66	11.46	11.88	12.03	11.78	11.86	11.86	11.86
Std. dev.	4.01	3.80	3.74	3.98	4.13	3.87	3.85	4.04	4.01	4.01	4.01
Not cycling											
# obs.	9,562	10,107	9,949	8,422	7,628	8,346	8,795	8,616	9,558	9,575	9,570
Mean	9.47	9.30	9.05	9.52	9.73	9.08	8.94	9.35	9.47	9.47	9.47
Std. dev.	4.97	5.04	4.98	5.22	5.20	5.18	5.03	5.02	4.97	4.97	4.96
Difference											
Mean diff.	2.39	2.77	3.16	2.14	1.73	2.80	3.09	2.43	2.39	2.39	2.39
Welch's t	39.53	46.74	53.80	32.96	25.64	43.53	50.02	38.67	39.53	39.55	39.60
D. F.	17,247	17,771	17,314	13,648	12,134	13,263	14,608	14,723	17,236	17,282	17,295
p value	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$
II. New South Wales (# manually labeled observations: 9,693)											
Cycling											
# obs.	6,878	8,324	8,038	9,693	7,303	7,704	7,994	9,253	7,052	6,961	7,183
Mean	12.03	11.73	12.35	11.66	12.48	11.76	11.81	11.58	12.19	12.07	12.13
Std. dev.	5.51	5.80	5.58	6.04	5.48	5.89	5.84	5.99	5.54	5.53	5.56
Not cycling											
# obs.	2,815	1,369	1,655	0	2,390	1,989	1,699	440	2,641	2,732	2,510
Mean	10.76	11.25	8.33	—	9.18	11.28	10.97	13.48	10.25	10.64	10.33
Std. dev.	7.10	7.31	7.01	—	6.92	6.56	6.85	6.79	7.01	7.08	7.08
Difference											
Mean diff.	1.27	0.48	4.02	—	3.30	0.48	0.84	−1.90	1.94	1.43	1.80
Welch's t	8.50	2.31	21.94	—	21.24	2.97	4.70	−5.76	12.80	9.48	11.55
D. F.	4,266	1,663	2,106	—	3,423	2,870	2,252	472	3,939	4,103	3,648
p value	$< .001$	.021	$< .001$	—	$< .001$	.003	$< .001$	$< .001$	$< .001$	$< .001$	$< .001$
III. Germany (# manually labeled observations: 35,685)											
Cycling											
# obs.	14,116	0	1,013	72	8,763	7	7	14,281	11,762	13,574	15,299
Mean	98.18	—	99.57	99.67	98.73	114.11	115.64	98.19	98.38	98.16	98.18
Std. dev.	3.57	—	6.96	3.26	3.84	32.10	31.40	3.60	3.60	3.59	3.51
Not cycling											
# obs.	21,569	35,685	34,672	35,613	26,922	35,678	35,678	21,404	23,923	22,111	20,386
Mean	98.65	98.46	98.43	98.46	98.38	98.46	98.46	98.65	98.50	98.65	98.68
Std. dev.	4.37	4.08	3.96	4.08	4.15	4.05	4.05	4.36	4.30	4.34	4.45
Difference											
Mean diff.	−0.47	—	1.14	1.21	0.35	15.65	17.18	−0.46	−0.12	−0.49	−0.50
Welch's t	−11.11	—	5.19	3.14	7.26	1.29	1.45	−10.86	−2.77	−11.55	−11.86
D. F.	33,984	—	1,031	71	15,941	6	6	34,110	27,415	32,697	35,595
p value	$< .001$	—	$< .001$	.002	$< .001$	.245	.197	$< .001$	.006	$< .001$	$< .001$

Note: Columns (1)–(10) use the median-accuracy version of each method in Table 4.2. The unit of measurement (of means and standard deviations) is the Australian cent in WA and NSW, and the euro cent in Germany, respectively. The p value indicates the probability that the difference in means is zero based on Welch's t statistic and the approximate degrees of freedom.

worth cycles.

4.7.2 Additional Findings

The results sections constitute our main findings, but the curious patterns in the data present additional puzzles. We address some of them in the following paragraphs.

1. Why Existing Methods Work in Australia But Fail in Germany. Most of the cycles in Australia follow specific (almost deterministic) frequencies and exhibit strong asymmetry, whereas German cycles are noisier and not always asymmetric. The existence of asymmetric *non*-cycles in Germany further complicates the issue. Hence, asymmetry-based methods correctly identify cycles in Australia but not in Germany.

2. Why Margins And Cycles Correlate Positively in Australia But Negatively in Germany. In all datasets, the mean and the standard deviation of margins are positively correlated. That is, higher markups tend to accompany higher volatility. The reason is that retail and wholesale prices are relatively close so that the only direction in which margins can move significantly is *upward* (unless stations are willing to incur losses). We find volatility and cyclicality are correlated positively in Australia but negatively in Germany. Therefore, the average level and cyclicality of margins are correlated positively in Australia but negatively in Germany.

3. How Can Cycles Be Less Volatile Than Non-Cycles? Cyclicality implies systematic—but not necessarily large—movements; not all large/frequent movements follow cycles. Many German observations exhibit high volatility without any discernible patterns, which explains the existence of "volatile non-cycles" in the data.

4. Why Existing Methods Find "Positive Correlations." These methods' threshold rules tend to recognize high-mean, high-volatility cases as "cycles" because only sufficiently large movements can satisfy these conditions. In Germany, however, volatility is a poor predictor of cyclicality (see Question 3 above).

5. Could Intra-Day Cycles Be the Source of Curious Patterns in Germany? The answer is "yes" and "no." In general, our daily sampling frequency and 90-day window are suitable for identifying cycles with the frequencies of several days to a month or so. Shorter frequencies may not be well represented.

Nevertheless, if the "intra-day" cycles follow the frequency of exactly 24 hours (or any hours that can divide 24 evenly), they would be "averaged out" in the process of computing daily prices and would not affect our observations. The existing studies suggest that they do follow exactly 24-hour cycles. Hence, how intra-day cycles affect the multi-daily volatility in our data is not obvious.[34]

6. Why Manual Classification Provides a Relevant Benchmark. At this point, one might question (again) the relevance of human recognition as a benchmark. Our answer is still the same as before: It is the "second best" option. If we had a perfect mathematical definition, no detection problem would arise in the first place. In the absence of such a formula, the existing research relied on rules of thumb that were ultimately validated by selective eyeballing by the authors. We made this process more systematic and transparent.

4.7.3 Exploratory Data Analysis

As a further demonstration of the use of automatic cycle detection, this section investigates the distribution of price cycles across time and space. Obviously, such an exploratory data analysis becomes possible only after a scalable method to detect cycles is used on the entire dataset. We first describe time-series patterns and then explore cross-sectional correlations.

Time Series Patterns. How many stations exhibit price cycles at each point in time? The two panels of Figure 4.3 plot the fractions of stations that exhibit price cycles in Australia and Germany, respectively. Throughout this section, the recognition of cycles is based on the median-performance version (parameter values) of the most accurate algorithm (Method 10), which we apply to the entire dataset—both labeled and unlabeled—in each region/country.

The two regions of Australia, WA and NSW, show mostly high percentages of cycling stations. WA offers the longest data period. Byrne and de Roos (2019) [30] documented clear price cycles in two subperiods (2007–2008 and 2010–2015),

[34] One possibility is the existence of "medium frequency" cycles that are longer than 24 hours, but shorter than 3–4 days. However, we are not aware of any studies that document such cycles. In short, the coexistence of daily, weekly, and other cycles and their interactions constitute an open-ended question for further research.

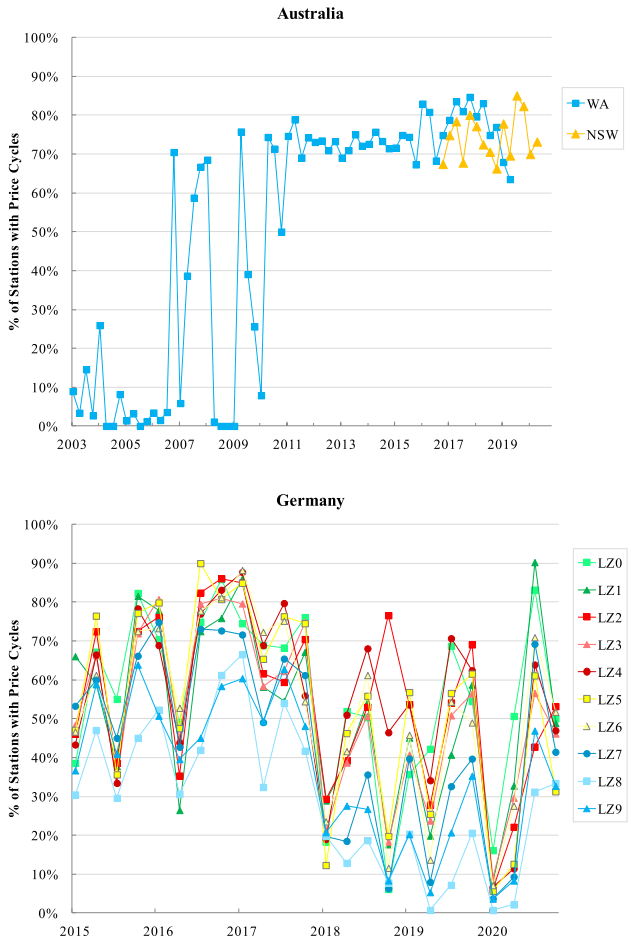


Figure 4.3: *How Many Stations Exhibit Price Cycles?*

Note: LZ0–LZ9 are Germany's 10 geographic zones (Postleitzonen). See main text for details.

both of which correspond to the periods in which cycles are prevalent according to our method.[35] Thus, the results of our method confirm Byrne and de Roos's description of the WA data in terms of time series. The NSW dataset starts relatively recently in 2016:Q4. Its range of approximately 70%–90% is comparable to WA.

[35]Readers might wonder what causes sudden increases and decreases in WA in the 2000s. Some of them reflect genuine changes in the number of cycling stations; others could be due to noise in the original data because the WA database lacks a consistent station identifier. Even though we tried to reconstruct as "balanced" panel data as possible (based on street addresses and other observable characteristics), the recorded number of stations varies across time, sometimes quite dramatically.

The German picture is more "colorful," with greater heterogeneity across regions. We show the fraction of cycling stations in each of the 10 geographic zones (*Postleitzonen*, henceforth LZs). LZ0 and LZ1 (in green) correspond to former East Germany; LZ2–LZ6 (in red and yellow) are northwestern regions; LZ7–LZ9 (in blue) roughly correspond to the southern states of Baden-Württemberg and Bavaria.[36] Three patterns emerge. First, whereas LZ0–LZ6 tend to move together in relatively high ranges, LZ7–LZ9 exhibit consistently lower percentages. Second, despite these differences in levels, all regions display similar fluctuations most of the time, and such fluctuations could be large. Third, as a general trend, the overall range shifted downward from 30%–90% in 2015–2017 to 0%–70% in 2018–2019. The timing of this change would seem to roughly coincide with the introduction and dissemination of automatic pricing algorithms (see Assad et al. (2021) [15]), but the clarification of their causal relationship would require further research.

Spatial Patterns. We now explore spatial patterns within each region. The geographical scope of price cycles (e.g., local, city-wide, or regional) and their synchronization patterns might shed light on their mechanism and potentially inform the definition of relevant markets for antitrust purposes.

Specifically, we investigate whether multiple gasoline stations tend to exhibit price cycles at the same time, and if so, how such tendencies change with the distance between them.

We construct our measure of "correlation" between stations as follows. First, within each region, we list all possible pairs of stations and split them into seven distance bins (less than 1km, 1–5km, ..., 50–100km, and above 100km) based on their Euclidean distances.[37] Second, for each pair, we calculate the *percentage of quarters in which their cycle statuses match* (i.e., either both stations exhibit cycles or neither of them does). Third, for each distance bin in each region, we take the average of these percentages, either across all pairs or across pairs of same-brand stations. This procedure creates a summary statistic of how well the presence or absence of cycles is synchronized across multiple stations in each region—and how their "correlation" varies with distance.[38]

Figure 4.4 reports the spatial patterns of "correlation" in four graphs: (i) all

[36] For maps and further details, see Wikipedia page on "Postal codes in Germany" at https://en.wikipedia.org/wiki/Postal_codes_in_Germany (accessed on January 10, 2023).

[37] Note we consider only pairs that share at least 12 calendar quarters of valid data in common.

[38] We say "correlation" in quotes because we use the "percentage of quarters with matched cycle statuses" instead of correlation coefficient, which is undefined when a station always (or never) shows cycles.

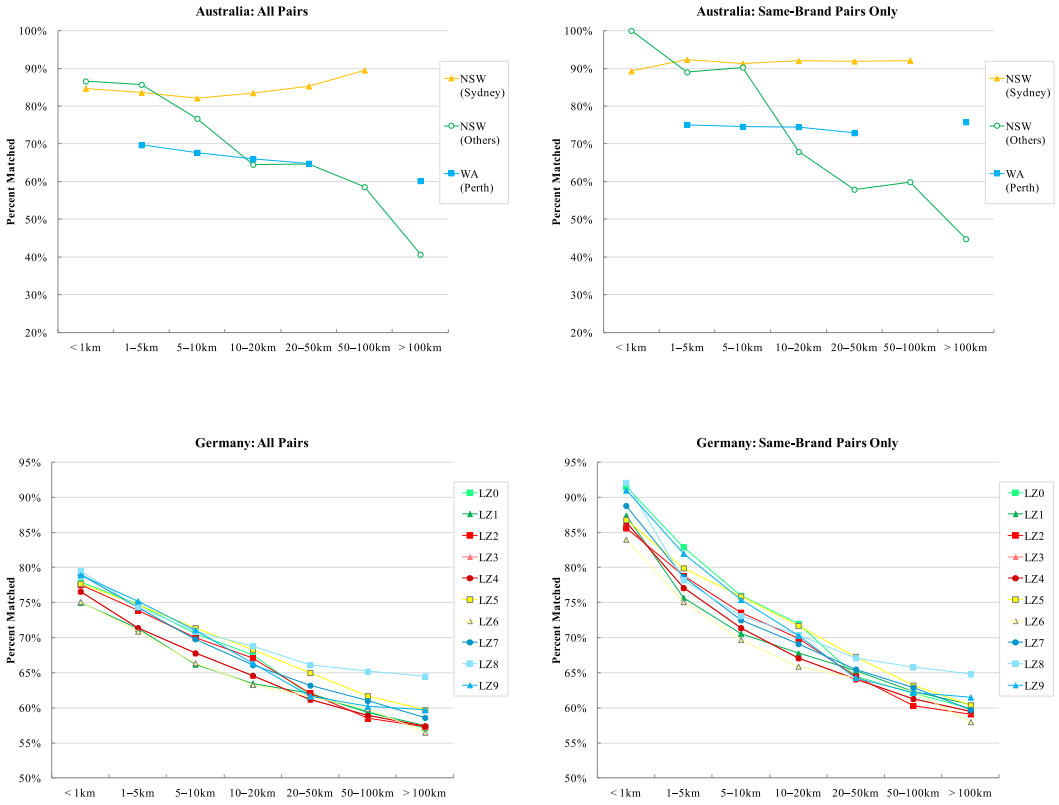


Figure 4.4: *How Presence of Cycles Correlates between Stations*

Note: See main text for the definition of "percent matched" statistics. The WA graphs do not show markers or lines in two distance bins (less than 1km and 50–100km) because the WA data have relatively few pairs in these bins, which we grouped in the adjacent bins for the purpose of visualization.

pairs in WA and NSW, (ii) same-brand pairs in WA and NSW, (iii) all pairs in Germany, and (iv) same-brand pairs in Germany. Four patterns emerge. First, the majority of the station-pair-quarter observations shares cycle status, with the exception of the most distant (>100 km) bin in rural NSW. Second, the cities and the rural areas of Australia exhibit qualitatively different patterns. The station pairs within Perth and Sydney (the capital cities of WA and NSW, respectively) tend to show high correlations with limited variability across distance bins, whereas the rest of NSW features " correlations" that decrease with distance.[39] Third, all 10 LZs of Germany show similar patterns in which "correlations" steadily decrease with distance. Fourth, pairs of same-brand stations

[39]All of the WA stations (with sufficient observations for these plots) are in Perth, which is why we do not split WA into urban and rural areas as in NSW.

tend to be more correlated than all/any pairs in both Australia and Germany, especially in the 0km–10km bins in Germany.

4.8 Practical Recommendations

Based on our findings in sections 5–7, we suggest the following steps as a practical (but not necessarily the most rigorous) guide for automating the detection of Edgeworth cycles:

1. Choose the data frequency and time window that would permit the identification of hypothesized cycles. That is, the sampling frequency must be shorter than that of suspected cycles, and the time horizon should accommodate at least a few repetitions. (For the sake of simple exposition, our explanation in the following keeps assuming the daily frequency and the quarterly window.)

2. Eyeball and manually categorize a random sample of 100 station-quarter observations in terms of cyclicality (but not necessarily asymmetry).[40] If sufficient numbers of both cyclical and non-cyclical cases are found, proceed to the next step. If not, increase the sample size.

3. As a first attempt to algorithmically distinguish cycles from non-cycles, calibrate one of the simpler methods. We recommend the two-parameter model of Method 4 (MBPI) because it is the only one (among Methods 1–4) that captures the notion of cyclicality.

4. For more formal, mathematical definitions of cyclicality, use Methods 5 (FT) or 6 (LS), both of which are readily implementable in many programming languages for scientific computing. Method 7 (CS) is another option with similarly off-the-shelf implementations.

5. If the performance of these methods is unsatisfactory, try Methods 9 (E-RF), 8 (LSTM), and 10 (E-LSTM), in increasing order of complexity and expected accuracy.

[40] Adversarial circumstances, such as antitrust cases, could potentially introduce biases in the manual labeling of data. Hence, the selection and training of human labelers (in more formal contexts that the one assumed here) might have to be treated with the same care as in the selection and training of jury in trials.

6. Once the detection of cyclicality (as recognized by humans) is successfully automated, refine the classification of " cycling" observations in terms of asymmetry. The median-price-change statistic from Method 3 (NMC) offers a simple way to capture asymmetry. For example, one can distinguish between the Edgeworth-type asymmetry (i.e., the median change is negative), the inverse-Edgeworth asymmetry (i.e., the median change is positive), and symmetry (i.e., the median change is approximately zero). Methods 1 (PRNR) and 2 (MIMD) can be used for the same purpose.

7. If desired, this asymmetry-based classification can be automated by using some clustering algorithm on the distribution (e.g., a histogram) of the median price change across station-quarter observations. This process can be designed as either supervised or unsupervised machine-learning tasks.

8. Once the classification based on both cyclicality and asymmetry is complete, compute the mean margin and other statistics for each type of observation (e.g., Table 4.3). Welch's t statistic and the associated degrees of freedom can be used for testing the null hypothesis that the means of the two subsamples (of potentially different sizes) are equal.

9. The previous step assumes that the dataset contains only prices and margins. If additional data are available on the characteristics of gasoline stations and their locations (as well as other demand- and supply-side factors such as competition), control for these additional covariates in a suitable regression model.

10. At any point after step 4, one might also consider another refinement based on the frequency of cycles. Cycles of multiple lengths may coexist within a single dataset. Methods 5–7 would be useful for this purpose.

Thus, even though Method 10 (E-LSTM) is the top runner in terms of cycle-detection accuracy, other methods (including the existing ones) have important roles to play, both as a tool for initial inspection and as a summary statistic for refinement.

4.9 Conclusion

We propose scalable methods to detect Edgeworth cycles so that the growing amount of "big data" on fuel prices can be scrutinized. The failure of the existing

methods in noisy data suggests further investigation would benefit from distinguishing "cyclicality" from "asymmetry." Our nonparametric methods achieve the highest accuracy; such flexible models typically require large amounts of training data, but the requirement is minimal in this context. Whether researchers discover a positive or negative statistical relationship between markups and cycles depends on the choice of method. Because such "facts" are supposed to inform regulations and competition policy, these methodological considerations are directly policy relevant.

Data/Code Availability. The replication package [72] and the Online Appendix are publicly available on Github.

4.10 Appendix A: Methodological Details and Simulations

4.10.1 A.1 Details of the New Methods

Fourier Transform (Method 5). The Fourier transform of a continuous function $g(x)$ is

$$G(f) \equiv \int_{-\infty}^{\infty} g(x) e^{-2\pi i f x} dx. \tag{4.22}$$

Let us define the Fourier transform operator $\mathcal{F}$ such that $\mathcal{F}\{g\} = G$, which is a linear operation. A sinusoidal signal (i.e., sine wave) with frequency f_0 has a Fourier transform consisting of a weighted sum of the Dirac delta functions at $\pm f_0$.[41] The practical implication of these properties is that any signal made up of a sum of sinusoidal components will have a Fourier transform consisting of a sum of delta functions that mark the frequencies of those sinusoids. Thus, the Fourier transform directly measures additive periodic content in a continuous function. The power spectral density (PSD, or the power spectrum) of a function,

$$P_g \equiv |\mathcal{F}\{g\}|^2, \tag{4.23}$$

is a positive, real-valued function of frequency f, and provides a convenient way to quantify the contribution of each frequency f to the signal $g(x)$.

When a continuous time series is sampled at regular time intervals with spacing Δx, as is the case in our data, one can use the discrete version of (4.22):

$$G_{obs}(f) = \sum_{n=-\infty}^{\infty} g(n\Delta x) e^{-2\pi i f n \Delta x}. \tag{4.24}$$

Acknowledging the finite sample size N and focusing on the relevant frequency range $0 \leq f \leq \frac{1}{\Delta x}$, one can define N evenly spaced frequencies with $\Delta f = \frac{1}{N\Delta x}$ covering this range. Let $g_n \equiv g(n\Delta x)$ and $G_k \equiv G_{obs}(k\Delta f)$. Then, the sample analog of (4.22) is

[41] The Dirac delta function is $\delta(f) \equiv \int_{-\infty}^{\infty} e^{-2\pi i f x} dx$, and hence, we can write $\mathcal{F}\left\{e^{2\pi f_0 x}\right\} = \delta(f - f_0)$. The linearity of $\mathcal{F}$ and Euler's formula for the complex exponential ($e^{ix} = \cos x + i \sin x$) lead to the following identities: $\mathcal{F}\{\cos(2\pi f_0 x)\} = \frac{1}{2}[\delta(f - f_0) + \delta(f + f_0)]$ and $\mathcal{F}\{\sin(2\pi f_0 x)\} = \frac{1}{2i}[\delta(f - f_0) + \delta(f + f_0)]$. See VanderPlas (2018) [134] for further details.

$$G_k = \sum_{n=0}^{N} g_n e^{-2\pi i k n/N}. \tag{4.25}$$

One can construct the sample analog of the Fourier power spectrum (4.23) as (4.6) in the main text. This is the "classical" or "Schuster" periodogram.[42]

A potential drawback of the threshold rule in (4.7) is that it exclusively focuses on the highest point and ignores the rest. As an alternative rule, we can compare the highest point with the heights of other, less powerful frequencies. One way to capture relative heights of multiple frequencies is to measure the " concentration" of power in a limited number of frequencies. We use the Herfindahl-Hirschman Index (HHI) for an additional check for "significant" cycles:

$$HHI_{i,t} \equiv \sum_f \left(\frac{P_{i,t}(f)}{\sum_f P_{i,t}(f)} \right)^2 > \theta_{hhi}^{FT}, \tag{4.26}$$

where $\theta_{hhi}^{FT} \in (0,1]$ is a scalar threshold parameter.[43] A high value of $HHI_{i,t}$ indicates strong periodicity at certain frequencies relative to other, weaker frequencies.

Lomb-Scargle Periodogram (Method 6). Even though the classical periodogram in (4.6) appears different from (4.8), (4.6) can be rewritten as

$$P(f) = \frac{1}{N}\left[\left(\sum_n g_n \cos(2\pi f x_n) \right)^2 + \left(\sum_n g_n \sin(2\pi f x_n) \right)^2 \right].$$

Thus, the only major difference between (4.6) and (4.8) is the denominators in (4.8).

Statistically, one can interpret the Lomb-Scargle periodogram as a collection of least-squares regressions in which one fits a sinusoidal model at each frequency f:

$$\hat{g}(x; f) = A_f \sin\left(2\pi f \left(x - \phi_f\right)\right), \tag{4.27}$$

[42] See Press et al. (1992, section 12.2) [112] for computational implementation.

[43] The HHI is a summary statistic that is typically used to measure the degree of market-share concentration in oligopolistic industries. A high value of the HHI indicates the market is close to monopoly.

where amplitude A_f and phase ϕ_f are the parameters to be estimated by minimizing the sum of squared residuals:

$$SSR^{LS}(f) \equiv \sum_n (g_n - \hat{g}(x_n; f))^2 . \tag{4.28}$$

Scargle (1982) [120] shows the following periodogram is identical to (4.8):

$$\tilde{P}^{LS}(f) = \frac{1}{2}\left[SSR_0^{LS} - SSR^{LS}(f)\right],$$

where SSR_0^{LS} is the sum of squared residuals from the restricted model in which the only regressor is a constant term. The idea is that the frequencies with good fit will exhibit high $\tilde{P}^{LS}(f)$.

The HHI variant of the LS method is

$$HHI_{i,t}^{LS} \equiv \sum_f \left(\frac{P_{i,t}^{LS}(f)}{\sum_f P_{i,t}^{LS}(f)} \right)^2 > \theta_{hhi}^{LS}. \tag{4.29}$$

Cubic Splines (Method 7). A spline is a piecewise polynomial function:

$$S_K(x) = \sum_{j=0}^{P} \beta_j x^j + \sum_{k=1}^{N} \beta_{P+k} (x - \tau_k)^P \mathbb{I}\{x \geq \tau_k\}, \tag{4.30}$$

where $K = 1+P+N$ is the number of coefficients, P is the order of the polynomial (not to be confused with the periodogram in Methods 5–6 or our notation for the price, p), and the support for x is covered by $N+1$ ordered subintervals that are joined by N knots ($\tau_1 < \tau_2 < \cdots < \tau_N$).[44] It is a special case of a sieve/series approximation that constitutes a class of nonparametric regression methods.[45] We use splines as an interpolator to smooth the discrete (daily) time series and facilitate further calculations. Specifically, we use a cubic Hermite interpolator, which is a spline where each piece is a third-degree polynomial of Hermite form (i.e., $P = 3$, $N = 88$, and βs are prespecified).[46]

[44] This N should not be confused with our notation for sample size in the discrete Fourier transform.

[45] Any continuous function can be uniformly well approximated by a polynomial of sufficiently high order, and the rate of approximation is $o\left(K^{-2}\right)$. Other series models include trigonometric polynomials, wavelets, orthogonal wavelets, B-splines, and artificial neural networks. See Hansen (2020, ch. 20) [67] for an introduction and Chen (2007) [37] for a review.

[46] On the unit interval $d \in (0,1)$, given a starting point p_0 at $d = 0$, an ending point p_1 at $d = 1$, and slopes m_0 and m_1, this polynomial is

$$p(d) = \left(2d^3 - 3d^2 + 1\right)p_0 + \left(d^3 - 2d^2 + d\right)m_0 + \left(-2d^3 + 3d^2\right)p_1 + \left(d^3 - d^2\right)m_1.$$

In addition to the indicator of frequent oscillations in (4.11), we propose a measure that captures amplitude as well. We subtract the lowest daily price in (i,t) from all of its daily prices, $\underline{p}_{i,d} \equiv p_{i,d} - min_{d\in t}\left(p_{i,d}\right)$, fit CS to $\left(\underline{p}_{i,d}\right)_{d\in t}$, and calculate its integral over $d \in [1,90]$. We set $cycle_{i,t} = 1$ if and only if

$$\int_1^{90} \underline{CS}_{i,t}(d) > \theta_{int}^{CS}, \tag{4.31}$$

where $\underline{CS}_{i,t}(d)$ is the fitted value of $\underline{p}_{i,d}$ at time d. Because this definite integral equals the area between the price series and its lowest level within (i,t), this condition captures cycles with large amplitude and sustained high prices.

We also construct a discrete (raw data) analog of the splines-integral measure as follows:

$$\sum_{d=1}^{90} \left|\bar{p}_{i,d}\right| > \theta_{abs}^{CS}, \tag{4.32}$$

where $\bar{p}_{i,d}$ is the demeaned price. The information content of this statistic is similar to the previous one, but its calculation is simpler.

Long Short-Term Memory (Method 8). Compared with Greff et al.'s (2017) [60] "vanilla" setup, we make two simplifications. First, our law of motion for $\mathbf{c}_d^l$ (4.13) uses the same set of parameters $\left(\omega_7^l, \omega_8^l, \omega_9^l\right)$ twice. This simplification corresponds to their "Coupled Input and Forget Gate" variant due to Cho et al. (2014) [39], which is also referred to as Gated Recurrent Units (GRUs) in the literature. Second, we do not include $\mathbf{c}_d^l$ or $\mathbf{c}_{d-1}^l$ inside Λ in (4.12) or inside tanh and Λ in (4.13). This omission corresponds to their "No Peepholes" variant. Greff et al. (2017) [60] show these simplifications reduce the number of parameters without compromising predictive accuracy.

We implement LSTM in TensorFlow-GPU 2.6 (tf.keras.models.Sequential). Our choice of network architecture and activation functions—which constitute the specification of effective functional forms—are as explained in the main text. The total number of weight parameters is 2,165. We set other tuning parameters and the details of numerical optimization as follows: (i) the dropout rate is 0.5, (ii) the optimizer is tf.keras.optimizer.RMSprop with the learning rate of 0.0005, (iii) the number of epochs is 100, and (iv) the batch size is 30.

This form ensures the observed values (p_0, p_1) and their slopes (m_0, m_1) are fitted exactly. It has become a default specification of CS in SciPy, a set of commonly used Python libraries for scientific computing.

Ensemble in Random Forests (Method 9). The relationship between "decision trees" and "random forests" is as follows, according to Murphy (2012, ch. 16) [99]. Because finding the truly optimal partitioning in a decision-trees model is computationally infeasible, some greedy, iterative procedures are used in the estimation/tuning of the parameters $\left(\omega^{RF}, \kappa^{RF}\right)$. However, the hierarchical nature of this process leads to unstable predictions. Averaging over multiple estimates from bootstrapped subsamples ("bootstrap aggregating" or "bagging") is a commonly used technique to reduce this variance. A further improvement is possible by randomly choosing a subset of input variables, in addition to " bagging." This technique is called " random forests" (Breiman 2001a) [22] and is known to perform well in many different contexts (e.g., Caruana and Niculescu-Mizil 2006) [34].

We implement E-RF in scikit-learn 0.24.2 (sklearn.ensemble. RandomForestClassifier), with default options for all settings.

Ensemble in Long Short-Term Memory (Method 10). Our E-LSTM implementation details are the same as in the basic LSTM (Method 8). The only difference is that the total number of weight parameters is larger at 2,933 to incorporate the additional input variables from Methods 1–7.

4.10.2 A.2 Parameter Optimization

We define two types of prediction errors as follows:

$$\begin{aligned}
\%\text{ false negative}(\theta) &\equiv \frac{\sum_{(i,t)} \mathbb{I}\left\{\widehat{cycle}_{i,t}(\theta) = 0, cycle_{i,t} = 1\right\}}{\#\ all\ predictions} \times 100, \text{ and} \\
\%\text{ false positive}(\theta) &\equiv \frac{\sum_{(i,t)} \mathbb{I}\left\{\widehat{cycle}_{i,t}(\theta) = 1, cycle_{i,t} = 0\right\}}{\#\ all\ predictions} \times 100.
\end{aligned}$$

They correspond to type II errors and type I errors in statistics, respectively.

We occasionally encounter cases in which a range of parameter values attain the same (maximum) accuracy. In such cases, we report the median of all θ^* values that we find in our grid search. These cases typically involve "degenerate" predictions in which $\widehat{cycle}_{i,t}(\theta) = 1$ or $\widehat{cycle}_{i,t}(\theta) = 0$ for all (i,t), and hence are mostly irrelevant for the purpose of finding well-performing θs.

Optimization Algorithm The objective function to be maximized for all parametric models was classification accuracy (defined as the fraction of correct classifications) as a function of theta. These functions are discrete in the output space and non-convex, making derivative-based optimization impossible. To optimize these functions within the context of the computational framework, given the limits of the functions, we implemented a grid-search based optimization algorithm, that we call Memoized Adaptive Grid Search (MAGS). MAGS is robust to badly behaved functions, while being reasonably efficient in terms of the number of function evaluations, given that it memoizes function evaluations and adapts the resolution of the search window to increase resolution as it converges on the goal. MAGS requires a search domain, as well as search resolution, search window cut, and a termination tolerance parameters. The MAGS algorithm is presented in Algorithm 4.

MAGS has no optimality guarantees, but it is guaranteed to converge given the rounding function on line 12, and the termination condition. In practice, it worked well to optimize these functions in time on the order of milliseconds or seconds in a fully serial implementation. The biggest factor in the run time is the function evaluations, which vary significantly in complexity, with the most computationally intensive function evaluations being the Lomg-Scargle Periodograms. The MAGS algorithm has the advantage of being trivially parallelizable, since the function evaluations are fully independent and constitute the vast majority of the runtime.

4.10.3 A.3 Performance on Simulated Cycles

This section studies the performance of each method on simulated data. Because artificial cycles and real-world cycles are qualitatively different, our purpose is not so much testing algorithms as understanding them better in a controlled environment.

Setup. We consider four kinds of DGPs: (i) white noise, (ii) Edgeworth cycles, (iii) "reverse Edgeworth" cycles, and (iv) sine waves of various lengths. First, white noise is simply a 90-day sequence of i.i.d. random draws from the standard normal distribution. If a method "detects" cycles in white noise, we interpret it as a false positive. Second, we simulate theoretical Edgeworth cycles by using Maskin and Tirole's (1988) [97] example.[47] Third, we simulate cycles with

[47] We set the annual discount factor to 0.9, which translates into the daily discount factor of $\delta = 0.9997$ (in Maskin and Tirole's notation). The probability of a big price increase at the

Algorithm 4: Memoized Adaptive Grid Search

Data: Objective Function $f(\theta)$, Upper and lower bounds on θ, resolution of search grid ϕ, domain cut size s, tolerance ϵ
Result: Optimal θ^* which maximizes f for $p \in [\theta_{low},\ \theta_{high}] : p = \epsilon \left\lfloor \frac{\theta}{\epsilon} + \frac{1}{2} \right\rfloor$

Function MAGS($f(\theta)$, θ_{low}, θ_{high}, ϕ, s, ϵ):
  $l \leftarrow \theta_{low}$
  $r \leftarrow \theta_{high}$
  $w \leftarrow \infty$
  $i \leftarrow 1$
  $m \leftarrow \{\}$ /* hash table with key θ and value $f(\theta)$ */
  while $w > \epsilon$ **do**
    $w \leftarrow r - l$
    $\psi \leftarrow \frac{w}{\phi * i}$
    $p \leftarrow l$
    while $p \leq r$ **do**
      $p \leftarrow \epsilon \left\lfloor \frac{p}{\epsilon} + \frac{1}{2} \right\rfloor$ /* round p to same precision as ϵ */
      if $p \notin m$ **then**
        insert $p \mapsto f(p)$ into m /* $p \mapsto f(p)$ is key value pair */
      end
      $p \leftarrow p + \psi$
    end
    $c \leftarrow p : \max(f(p) \in m)$
    $l \leftarrow \max(c - \frac{w}{s}, l)$
    $r \leftarrow \min(c + \frac{w}{s}, r)$
    $i \leftarrow i + 1$
  end
return c

the opposite asymmetry (i.e., few big *decreases* and many small increases) by reversing the time stamps of simulated Edgeworth cycles in the above. Fourth, we generate sine waves (i.e., symmetric cycles) of five different wavelengths: 3, 7,

bottom of a price cycle (in their notation) is

$$\alpha(\delta) = \frac{\left(3\delta^2 - 1\right)\left(1 + \delta^2 + \delta^4\right)}{8 + 7\delta^2 + 2\delta^4 + 3\delta^6} \approx 0.2997.$$

With two firms taking turns to change prices in the grid of seven different price levels, $\left\{0, \frac{1}{6}, \frac{2}{6}, \ldots, 1\right\}$, the Edgeworth-cycle MPE entails asymmetric cycles of approximately weekly frequency. We simulate each 90-day sequence of data by randomly drawing one of the seven price levels as firm 2's initial price, to which firm 1 best-responds on day 1, to which firm 2 best-responds on day 2, and so on. These best responses are based on the equilibrium strategy profile of Maskin and Tirole's Table II. We use firm 1's prices as simulated data.

14, 21, and 28 days. We generate 10,000 quarters of simulated data based on each of these eight DGPs. Figure 4.5 shows examples of simulated cycles.

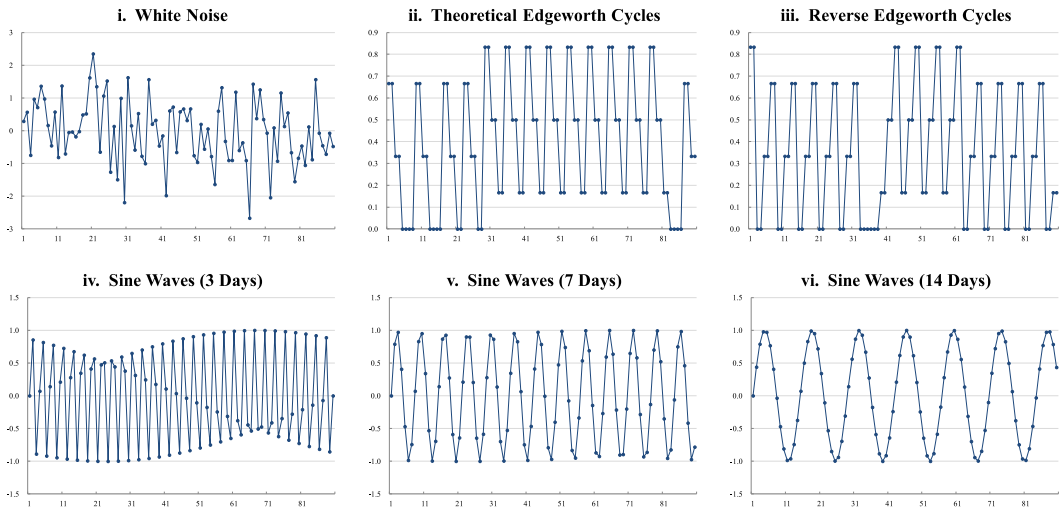


Figure 4.5: *Examples of Simulated Cycles*

Note: These pictures show examples of simulated price series before we transform them to match the mean and standard deviation of each dataset. The horizontal axes represent calendar days.

Each cycle-detection method comes in three versions as we optimize its parameters in three different datasets: WA, NSW, and Germany (Table 4.2). To match the mean and standard deviation of each (real) dataset, we rescale the simulated data through an affine transformation.

Results. Table 4.4 reports the percentages of simulated data (10,000 quarters each) that are classified as " cycling." Ideally, white noise should be classified as non-cycles and the rest as cycles, but that is not always the case. The top panel of Table 4.4 shows Method 4 (MBPI) and 7 (CS) mistakenly identify cycles in 100% of the white noise simulations. The reason is that they rely on counting the number of upward (and downward) price movements. White noise could trivially satisfy these criteria.

By contrast, Methods 1–3 are not fooled by white noise but fail to detect cycles in most other simulations, with the exception of Method 2 (MIMD) in theoretical Edgeworth cycles (53%). It is not surprising that these asymmetry-based methods fail to detect non-Edgeworth cycles; they are designed in such a way. Howerve, the result that Methods 1 (PRNR) and 3 (NMC) detect 0% of Edgeworth cycles *is* surprising. These false negatives are caused by the particular

Table 4.4: *Performance on Simulated Data (% Classified as Cycling)*

Method	(1) PRNR	(2) MIMD	(3) NMC	(4) MBPI	(5) FT	(6) LS	(7) CS	(8) LSTM	(9) E-RF	(10) E-LSTM
I. All Models Trained with Labeled Data from Western Australia										
White noise	0	0	36	100	25	19	100	0	0	0
Edgeworth	0	53	0	100	91	92	100	28	90	40
Reverse Edgeworth	0	0	0	70	91	92	100	0	6	0
Sine wave: 3 days	0	0	0	100	100	100	100	0	0	0
Sine wave: 7 days	0	0	0	85	100	100	100	72	3	54
Sine wave: 14 days	0	0	0	12	100	100	0	23	0	26
Sine wave: 21 days	0	0	0	0	100	100	0	0	0	0
Sine wave: 28 days	100	0	0	0	100	100	0	0	0	0
II. All Models Trained with Labeled Data from New South Wales										
White noise	100	0	80	100	22	15	100	0	3	0
Edgeworth	100	100	100	100	98	99	100	0	29	0
Reverse Edgeworth	65	0	100	15	98	99	100	0	0	0
Sine wave: 3 days	100	0	100	100	100	100	100	0	0	0
Sine wave: 7 days	100	0	100	20	100	100	100	0	27	0
Sine wave: 14 days	100	0	100	0	100	100	100	0	100	0
Sine wave: 21 days	100	0	100	0	100	100	100	87	100	100
Sine wave: 28 days	100	0	100	0	100	100	100	100	100	47
III. All Models Trained with Labeled Data from Germany										
White noise	0	24	17	100	0	0	100	85	4	100
Edgeworth	0	100	0	64	0	0	100	86	49	100
Reverse Edgeworth	0	0	0	100	0	0	100	96	4	18
Sine wave: 3 days	0	0	0	100	0	0	100	56	8	100
Sine wave: 7 days	0	0	0	100	0	4	100	36	0	100
Sine wave: 14 days	0	0	0	100	0	4	0	0	0	4
Sine wave: 21 days	0	0	0	92	0	4	0	0	0	0
Sine wave: 28 days	0	0	0	44	0	4	0	0	0	0

Note: Each result (%) is based on 10,000 quarters of simulated data. See text for details.

way in which Maskin and Tirole specify their model—each firm changes its price once every two days. There cannot be consecutive days of (strictly) positive or (strictly) negative "runs," which could fool Method 1. Likewise, Method 3 could be fooled by the 45 days of inaction in each 90-day sequence because the median price change is zero.

The "best" methods in the top panel are Methods 5 (FT) and 6 (LS) in the sense that they correctly reject most of the white noise as non-cycles and correctly detect most of the artificial cycles. In particular, their unique ability to detect cycles of any length is noteworthy. Even though they are trained in the WA data, which contain only weekly or two-week cycles, they correctly flag 100% of sine waves of both higher and lower frequencies.[48]

[48]We should note, of course, that these spectral methods are specifically designed for sine

Finally, the performance of the nonparametric/machine-learning models of Methods 8 (LSTM), 9 (E-RF), and 10 (E-LSTM) are somewhere in the middle. On the one hand, they correctly reject 100% of the white noise as non-cycles and correctly detect cycles in some of the simulated cycles (theoretical Edgeworth cycles and weekly sine waves). On the other hand, they ignore most of the reverse Edgeworth cycles and the other sine waves. In other words, they faithfully detect patterns that they are trained to recognize (i.e., the cycles with Edgeworth-type asymmetry or approximately weekly frequency in the WA data) and reject others.

The middle panel of Table 4.4 shows broadly similar results with the NSW-trained models. The original NSW data feature longer cycles of 2–4 week frequencies. Consequently, the nonparametric methods (8–10) respond only to the sine waves of relatively long wavelengths. The spectral methods (5 and 6) do well across all DGPs. Most of the existing methods (and Method 7) make degenerate predictions, but Method 2 happens to make perfectly correct predictions in the first two DGPs (white noise and Edgeworth cycles).

The bottom panel of Table 4.4 reports the performances of the models trained in the German data, in which cycles are noisy, nuanced, and generally difficult to detect. Almost all methods produce degenerate predictions on simulated data, with the exception of Method 2 (in the first two DGPs). Method 9 is another exception. Methods 4, 7, 8, and 10 make useless predictions by classifying most or all of white noise as cycles, but they are capable of distinguishing between shorter and longer cycles (i.e., they respond to shorter cycles but not longer ones). This distinction reflects the fact that typical cycles in Germany are weekly.

In conclusion, these simulations further clarify the performance characteristics of the algorithms. Certain methods (mostly Methods 1–4 and 7) struggle to reject white noise, whereas the spectral methods (Methods 5 and 6) perform well across the board—as long as they are trained on the data with clear cycles (WA and NSW). More flexible models (Methods 8–10) adapt to nuanced, specific data patterns. The 10 methods' relative performance rankings in the simulated data are often radically different from the ones in the real-world data (Table 4.2). Hence, perhaps the most important lesson from this exercise is that the real-world data are quite heterogeneous and qualitatively different from simulated data with artificial cycles. The analyst should use extreme caution when applying a pre-trained model to new datasets.

Artificial Cycles with Noise. How do performances change if we add noise to the simulated data? Figure 4.6 shows examples of simulated cycles with noise.

waves. Their real-world performance may not be as good, as we show in the main text.

Tables 4.5, 4.6, and 4.7 report results when small, medium, and large white noises are added to the artificial cycles, respectively. We do not add noise to the first "white noise" simulation because it is already pure noise. Nevertheless, the three tables keep listing the same white-noise results as a reminder that some of the predictions are degenerate (i.e., full of false positives).

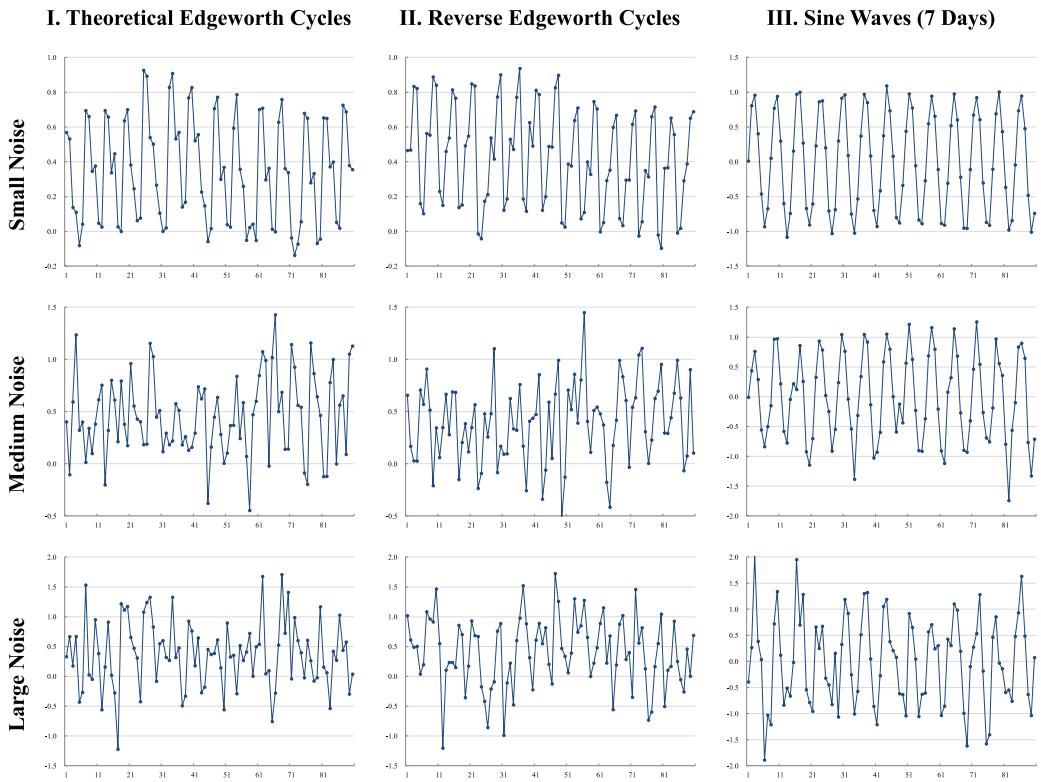


Figure 4.6: *Examples of Simulated Cycles with Noise*

Note: These pictures show examples of simulated price series before we transform them to match the mean and standard deviation of each dataset. The horizontal axes represent calendar days.

One might expect monotonically decreasing performances as we increase the noise level, which we operationalize as the standard deviation of the i.i.d. normal distribution. However, Table 4.5 shows small noise (standard deviation = 0.05) could actually help some of the existing methods that rely on asymmetry. Method 1 (trained in the WA data) now detects at least 10% of theoretical Edgeworth cycles, and Method 3 (also trained in the WA data) detects 93% of them, even though neither method could detect any *noiseless* Edgeworth cycles (Table 4.4). The reason is that the noise breaks the dominance of zero-price-change days in

theoretical Edgeworth cycles. The performance of Method 4 also improves, but its predictions are mostly degenerate (i.e., it classifies 100% of pure white noise as cycles) anyway. The additional noise mechanically increases the number of big price increases, which triggers this method to flag more cycles.

By contrast, Methods 5 and 6 are hardly affected by small noise. These spectral models correctly dismiss noise as noise because white noise does not contain any systematic frequency component. Method 7 is unaffected, but that is because it typically finds cycles in either 100% or 0% of the cases due to its simple rule. The impact of noise on Methods 8–10 is mixed. Noise decreases the percentages of detected Edgeworth cycles but often increases those of reverse Edgeworth cycles and 3-day sine waves. Overall, a little bit of noise could be either good or bad, or have no effect, depending on the nature of simulated cycles and detection algorithms.

Table 4.6 shows medium-sized noise (standard deviation = 0.25) induces broadly similar patterns, albeit with some differences. For example, Method 1 stops working. Method 3 seems to perform reasonably well, but it now detects only 72% of Edgeworth cycles, instead of 93% with small noise. Moreover, it starts detecting cycles in other simulations by chance (in the top panel) even though its NMC criterion is supposed to capture only Edgeworth-type asymmetry. Methods 5 and 6 still perform well (in the top and middle panel) but now detect only 63%–82% of the asymmetric cycles. Methods 8–10 are also negatively affected in most cases.

Finally, Table 4.7 reports the results under large noise (standard deviation = 0.5). Noise of any size is better than no noise for the use of Method 3 on theoretical Edgeworth cycles, but most of its predictions are now fooled by the presence of random shocks. Methods 5 and 6 are the only ones that continue to deliver valid classifications, albeit with the relatively low accuracy of 29%–39% for the asymmetric cycles.

Table 4.5: *Simulated Data with Small Noise (Standard Deviation = 0.05)*

Method	(1) PRNR	(2) MIMD	(3) NMC	(4) MBPI	(5) FT	(6) LS	(7) CS	(8) LSTM	(9) E-RF	(10) E-LSTM
I. All Models Trained with Labeled Data from Western Australia										
White noise	0	0	36	100	25	19	100	0	0	0
Edgeworth	10	0	93	100	90	92	100	22	7	17
Reverse Edgeworth	0	0	0	96	90	91	100	0	8	0
Sine wave: 3 days	0	0	27	100	100	100	100	0	0	0
Sine wave: 7 days	0	0	34	87	100	100	100	72	2	53
Sine wave: 14 days	0	0	7	17	100	100	0	22	0	26
Sine wave: 21 days	4	0	0	0	100	100	0	2	0	0
Sine wave: 28 days	6	0	0	0	100	100	0	0	0	0
II. All Models Trained with Labeled Data from New South Wales										
White noise	100	0	80	100	22	15	100	0	3	0
Edgeworth	100	7	100	100	97	98	100	0	2	0
Reverse Edgeworth	100	0	43	57	98	99	100	0	1	0
Sine wave: 3 days	100	0	92	100	100	100	100	0	0	0
Sine wave: 7 days	100	0	99	30	100	100	100	0	27	0
Sine wave: 14 days	100	0	99	0	100	100	100	0	99	0
Sine wave: 21 days	100	0	99	0	100	100	100	78	99	100
Sine wave: 28 days	100	0	100	0	100	100	100	100	100	45
III. All Models Trained with Labeled Data from Germany										
White noise	0	24	17	100	0	0	100	85	4	100
Edgeworth	0	93	48	99	0	0	100	85	41	100
Reverse Edgeworth	0	0	0	100	0	0	100	96	5	43
Sine wave: 3 days	0	7	5	100	0	0	100	55	14	100
Sine wave: 7 days	0	1	5	100	0	4	100	38	0	100
Sine wave: 14 days	0	0	0	100	0	4	0	0	0	2
Sine wave: 21 days	0	0	0	92	0	4	0	0	0	0
Sine wave: 28 days	0	0	0	46	0	4	0	0	0	0

Note: The "white noise" simulations and results are the same as in Table 4.4 (with standard deviation = 1 in the original simulation). We list them here for reference—as a reminder that some of the predictions are degenerate. Each result (%) is based on 10,000 quarters of simulated data. See the text for details.

Table 4.6: *Simulated Data with Medium Noise (Standard Deviation = 0.25)*

Method	(1) PRNR	(2) MIMD	(3) NMC	(4) MBPI	(5) FT	(6) LS	(7) CS	(8) LSTM	(9) E-RF	(10) E-LSTM
I. All Models Trained with Labeled Data from Western Australia										
White noise	0	0	36	100	25	19	100	0	0	0
Edgeworth	0	0	72	100	64	63	100	0	0	0
Reverse Edgeworth	0	0	10	100	64	63	100	0	1	0
Sine wave: 3 days	0	0	40	100	100	100	100	0	1	0
Sine wave: 7 days	0	0	40	99	100	100	100	56	3	27
Sine wave: 14 days	0	0	20	51	100	100	0	1	0	19
Sine wave: 21 days	0	0	11	34	100	100	0	0	0	0
Sine wave: 28 days	0	0	11	30	100	100	0	0	0	0
II. All Models Trained with Labeled Data from New South Wales										
White noise	100	0	80	100	22	15	100	0	3	0
Edgeworth	100	1	97	100	78	82	100	0	1	0
Reverse Edgeworth	100	0	50	100	77	82	100	0	0	0
Sine wave: 3 days	100	0	75	100	100	100	100	0	0	0
Sine wave: 7 days	100	0	90	91	100	100	100	0	29	0
Sine wave: 14 days	100	0	95	32	100	100	100	0	93	0
Sine wave: 21 days	100	0	95	19	100	100	100	0	99	0
Sine wave: 28 days	100	0	96	16	100	100	100	0	99	8
III. All Models Trained with Labeled Data from Germany										
White noise	0	24	17	100	0	0	100	85	4	100
Edgeworth	0	58	46	100	0	0	100	77	26	100
Reverse Edgeworth	0	5	3	100	0	0	100	90	6	99
Sine wave: 3 days	0	23	24	100	0	0	100	56	33	100
Sine wave: 7 days	0	12	16	100	0	4	100	61	0	100
Sine wave: 14 days	0	9	2	100	0	4	0	78	0	2
Sine wave: 21 days	0	7	1	100	0	4	0	13	0	0
Sine wave: 28 days	0	6	1	100	0	4	0	18	0	0

Note: The "white noise" simulations and results are the same as in Table 4.4 (with standard deviation = 1 in the original simulation). We list them here for reference—as a reminder that some of the predictions are degenerate. Each result (%) is based on 10,000 quarters of simulated data. See the text for details.

Table 4.7: *Simulated Data with Large Noise (Standard Deviation = 0.5)*

Method	(1) PRNR	(2) MIMD	(3) NMC	(4) MBPI	(5) FT	(6) LS	(7) CS	(8) LSTM	(9) E-RF	(10) E-LSTM
I. All Models Trained with Labeled Data from Western Australia										
White noise	0	0	36	100	25	19	100	0	0	0
Edgeworth	0	0	44	100	34	29	100	0	0	0
Reverse Edgeworth	0	0	29	100	34	29	100	0	0	0
Sine wave: 3 days	0	0	40	100	100	95	100	0	3	0
Sine wave: 7 days	0	0	38	100	100	100	100	2	4	1
Sine wave: 14 days	0	0	27	97	98	100	63	0	0	0
Sine wave: 21 days	0	0	22	95	100	100	49	0	0	0
Sine wave: 28 days	0	0	22	93	100	100	48	0	0	0
II. All Models Trained with Labeled Data from New South Wales										
White noise	100	0	80	100	22	15	100	0	3	0
Edgeworth	100	0	87	100	39	35	100	0	2	0
Reverse Edgeworth	100	0	75	100	38	35	100	0	1	0
Sine wave: 3 days	100	0	73	100	100	100	100	0	0	0
Sine wave: 7 days	100	0	87	100	100	100	100	0	4	0
Sine wave: 14 days	100	0	88	93	100	100	100	0	48	0
Sine wave: 21 days	100	0	88	88	100	100	100	0	55	0
Sine wave: 28 days	100	0	89	87	100	100	100	0	62	0
III. All Models Trained with Labeled Data from Germany										
White noise	0	24	17	100	0	0	100	85	4	100
Edgeworth	0	31	22	100	0	0	100	83	11	100
Reverse Edgeworth	0	19	13	100	0	0	100	87	7	100
Sine wave: 3 days	0	27	25	100	0	0	100	62	47	100
Sine wave: 7 days	0	18	16	100	0	4	100	79	0	100
Sine wave: 14 days	0	17	7	100	0	4	44	93	0	69
Sine wave: 21 days	0	16	5	100	0	4	33	94	0	35
Sine wave: 28 days	0	15	4	100	0	4	30	96	0	17

Note: The "white noise" simulations and results are the same as in Table 4.4 (with standard deviation = 1 in the original simulation). We list them here for reference—as a reminder that some of the predictions are degenerate. Each result (%) is based on 10,000 quarters of simulated data. See the text for details.

4.11 Appendix B: Additional Results

4.11.1 B.1 Combining Methods 1–4

This section investigates whether combining some or all of the existing methods leads to better performances. We construct 11 combinatorial methods based on Methods 1–4. Each combination comes in two specifications, AND and OR, depending on the logical operator combining its constituent methods. For example, the two variants of combination (7) in Table 4.8 are "Methods 1 AND 2 AND 3" and "Methods 1 OR 2 OR 3." The former detects cycles if all of Methods 1–3 do; the latter detects cycles if any of Methods 1–3 does.

Table 4.8: *Accuracy (%) of Combinatorial Methods*

Combination	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
Constituent methods:											
1. PRNR	Yes	Yes	Yes	–	–	–	Yes	Yes	Yes	–	Yes
2. MIMD	Yes	–	–	Yes	Yes	–	Yes	Yes	–	Yes	Yes
3. NMC	–	Yes	–	Yes	–	Yes	Yes	–	Yes	Yes	Yes
4. MBPI	–	–	Yes	–	Yes	Yes	–	Yes	Yes	Yes	Yes
	I. Western Australia (# manually labeled observations: 24,569)										
AND	90.34	90.88	90.89	91.18	91.74	89.58	90.34	90.33	90.89	91.18	90.33
OR	91.92	89.38	90.28	89.58	89.93	90.10	89.58	89.91	90.10	89.74	89.74
	II. New South Wales (# manually labeled observations: 9,693)										
AND	81.18	78.74	81.91	78.62	81.86	81.55	81.18	82.08	81.91	81.86	82.08
OR	76.18	70.96	78.38	70.96	78.31	70.96	70.96	76.03	70.96	70.96	70.96
	III. Germany (# manually labeled observations: 35,685)										
AND	60.44	60.44	60.44	60.49	60.84	60.48	60.44	60.44	60.44	60.49	60.44
OR	60.69	60.48	65.48	60.68	65.34	65.48	60.68	65.34	65.48	65.34	65.34

Note: "AND" and "OR" mean the constituent methods are combined with "and" and "or" operators, respectively. We randomly split the sample into an 80% training subsample and a 20% testing subsample 101 times. In each split, the former subsample is used for setting parameter values, whereas the latter subsample is used to evaluate the accuracy of predictions. All accuracy statistics are the medians from the 101 testing subsamples.

Table 4.8 shows that the performances of these combinatorial methods are similar to those of their constituent methods. The ranges of median accuracy results are 89%–92% in WA, 71%–82% in NSW, and 60%–65% in Germany, which are almost identical to those of individual Methods 1–4 in Table 4.2. Thus, combinations do not generate materially different predictions.

4.11.2 B.2 Variants of Methods 5–7

Table 4.9 reports the performances of the variants of Methods 5 (FT), 6 (LS), and 7 (CS). In Methods 5 and 6, the "max" and " HHI" variants are as explained in section 4.2 and Appendix A.1. The "peak" variant is similar to the "max" one except that we additionally use a peak-detection algorithm to ensure we are measuring the height of the highest (and well-behaved) peak in the power spectrum and not some accidental maximum due to noisy data. In Method 7, the "roots" variant is the baseline version in section 4.2. Its "integral" and "absolute value" variants are explained in Appendix A.1.

Table 4.9: *Performance of Automatic Detection Methods (Other Variants)*

Method	(5) FT_{max}	(5′) FT_{peak}	(5″) $FT2_{hhi}$	(6) LS_{max}	(6′) LS_{peak}	(6″) LS_{hhi}	(7) CS_{roots}	(7′) CS_{int}	(7″) CS_{abs}
I. Western Australia (# manually labeled observations: 24,569)									
Parameter 1	0.12	0.14	0.04	0.21	0.23	0.44	22.50	551.47	246.08
Parameter 2	—	—	—	—	—	—	—	—	—
% correct (median)	90.11	88.40	87.61	90.15	89.66	81.83	85.47	83.42	85.14
(Standard deviations)	(0.40)	(0.45)	(0.39)	(0.36)	(0.43)	(0.54)	(0.45)	(0.54)	(0.42)
of which cycling	58.24	57.31	59.12	57.92	57.10	54.13	56.41	55.92	57.14
of which not	31.87	31.09	28.49	32.23	32.56	27.70	29.06	27.49	28.00
% false negative	2.48	4.05	1.91	3.30	4.50	6.15	5.29	4.82	3.32
% false positive	7.41	7.5	10.48	6.55	5.84	12.03	9.24	11.76	11.54
II. New South Wales (# manually labeled observations: 9,693)									
Parameter 1	0.20	0.27	0.21	0.57	0.81	29.21	4.50	783.11	459.83
Parameter 2	—	—	—	—	—	—	—	—	—
% correct (median)	80.71	81.85	81.23	80.82	82.21	81.38	73.90	75.45	79.63
(Standard deviations)	(0.80)	(0.70)	(0.83)	(0.80)	(0.81)	(0.84)	(0.89)	(0.79)	(0.87)
of which cycling	66.53	66.99	64.72	66.43	66.89	67.30	70.40	68.13	67.25
of which not	14.18	14.85	16.50	14.39	15.32	14.08	3.51	7.32	12.38
% false negative	5.47	4.54	6.19	4.02	4.07	4.54	0.77	3.20	3.56
% false positive	13.82	13.62	12.58	15.16	13.72	14.08	25.32	21.35	16.81
III. Germany (# manually labeled observations: 35,685)									
Parameter 1	0.24	0.90	0.67	0.62	1.93	42.96	24.50	994.19	4,623
Parameter 2	—	—	—	—	—	—	—	—	—
% correct (median)	60.50	60.57	60.35	60.36	60.50	60.52	71.28	60.29	60.49
(Standard deviations)	(0.56)	(0.50)	(0.53)	(0.59)	(0.57)	(0.53)	(0.42)	(0.48)	(0.48)
of which cycling	0.00	0.00	0.00	0.00	0.00	24.30	25.88	0.00	0.00
of which not	60.50	60.57	60.35	60.36	60.50	47.00	45.40	60.29	60.49
% false negative	39.50	39.41	39.65	39.57	39.50	15.26	14.28	39.51	0.00
% false positive	0.00	0.01	0.00	0.07	0.00	13.43	14.45	0.20	39.51

Note: See the text of Appendix sections A.1 and B.2 for the definition of each method.

Conclusion

This dissertation has demonstrated how state-of-the-art computational tools and methods can be deployed to improve the analytical and control processes of energy markets including electric power markets and associated grids, and retail gasoline markets.

Part I demonstrated how significant improvements in the performance of computational models in time critical applications can be achieved by improved modeling techniques and optimizing the hardware utilization, either by selecting a method which reduces the bottlenecks or by adopting scheduling techniques better suited for the computational nature of the problem at hand. The case study presented in Chapter 1 showed how in the real-world trading environment, the careful management of the resources during the execution of embarrassingly parallel LP simulations improved the throughput of computations by 38%, reducing daily computation time from 6 166 CPU hours to about 2 280 CPU hours. Specifically, it was shown that on the basis of a simple benchmark of empirical measurements, without knowledge of the hardware architecture or characteristics of the solution algorithm, a procedure for parallel processing of a large number of simulations could be proposed and parameterized to significantly increase computational throughput for a given problem on a given computational resource.

Chapter 2 presented a selection of modeling and solution techniques to increase both the performance and scalability of power market optimization models. Specifically, it showed how dramatically the performance of a given mathematical model definition can vary depending on both the assembly and implementation of the model as well as the modeling framework used that is responsible for interfacing with the optimization solver. It further showed how improved scaling can make the difference between a model that can be feasibly deployed for simulations with a period of greater than a week, and one that can only be deployed for simulations with a time period of up to a few days.

Part II showed how the development of efficient tools to manage highly dynamic and stochastic power grids plays a part in lowering the carbon dioxide intensity of power production. Chapter 3 presented a novel data-driven approach to approximately solving a fully constrained formulation of the power generation unit commitment problem. By leveraging exascale computational tools such as ExaGO, stochastic modeling efforts can exploit parallel computing and modern computational resources to deliver high-resolution results in short time-frames. The outcome of such computations can be used to improve both grid operational efficiency and reliability. One of the key advantages of the presented algorithms is that they exploit the data generated by optimizations that are already an essential part of grid control workflows; imposing only minimal incremental computational burden by recycling data through graph-based algorithms of linear computational complexity. This minimal intervention approach showed promising results on non-trivial test grids of 500 and 2000 buses, consistently providing unit commitments that strictly enforced all security constraints (including non-linear) and minimized or fully eliminated the need for load shedding across contingencies with power imbalance. With further research, this data-driven unit commitment refinement approach could be tested, refined, and subsequently deployed across a wide array of grids, with larger contingency sets, and stochastic weather scenarios. A refined and robust implementation of such an approach could prove to be an efficient and practical method to include contingency and stochastic weather information into the unit commitment optimization process; providing more reliable and efficient grids with high penetrations of renewable energy.

Part III showed that the deployment of high-performance data analysis frameworks can unlock new possibilities in economics research, and provide tools and insights to policy makers that allow for the effective regulation and monitoring of high-impact markets such as retail gasoline. The scalable methods presented in Chapter 4 to detect Edgeworth cycles propose new paradigms to the economics community about how the growing amount of “big data” on fuel prices and can be scrutinized. While simple heuristic filtering and detection methods used in the existing body of fuel price literature failed to discern cycles in noisy data, the methods proposed, especially the non-parametric and machine learning methods, showed significant accuracy even in the most ambiguous cases. These flexible models typically require large amounts of training data, but analysis shows that such requirements are not a barrier to their successful deployment in academic or governance contexts. It was further shown that the definition of cycli-

cality itself is critical in monitoring and policy making considerations, since the definition used can be the difference between a positive and a negative statistical relationship between markups and cycles.

Closing remarks I will finish by thanking all of my collaborators that contributed to the work presented herein, and also played a pivotal role in my growth and education during this very informative period of my life. Throughout my doctoral studies I have been mentored and encouraged by so many inspiring colleagues. I am pleased to present this thesis as an important paving stone in my career path. I hope that I can look back upon this path at the end, and say that it has led humanity closer to a state of energy abundance.

Bibliography

[1] NVIDIA CUDA cuSPARSE Documentation, Accessed: 2024. URL https://docs.nvidia.com/cuda/cusparse/index.html. Accessed on: February 21, 2024.

[2] X. Abellan, J. Naranjo, C. Simarro, J. Rodriguez, P. Rodenas, and T. Holt. GREASY: CPU affinity extension, Apr. 2021. URL https://doi.org/10.5281/zenodo.4668583.

[3] S. Abhyankar, S. Peles, A. Mancinelli, and C. Rutherford. Evaluation of AC optimal power flow problems on graphical processing units. In *Proceedings of IEEE PES General Meeting*, 2021.

[4] S. Abhyankar, S. Peles, A. Mancinelli, and C. Rutherford. ExaGO Manual version 1.6, 2023.

[5] S. Abhyankar, S. Peles, A. Mancinelli, and C. Rutherford. Exago git repository, 2023.

[6] S. Y. Abujarad, M. Mustafa, and J. Jamian. Recent approaches of unit commitment in the presence of intermittent renewable energy resources: A review. *Renewable and Sustainable Energy Reviews*, 70:215–223, 2017. ISSN 1364-0321. URL https://www.sciencedirect.com/science/article/pii/S1364032116310140.

[7] D. H. Ahn, N. Bass, et al. Flux: Overcoming scheduling challenges for exascale workflows. *Future Generation Computer Systems*, 110:202–213, 2020. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2020.04.006.

[8] C. Alappat, A. Basermann, A. R. Bishop, H. Fehske, G. Hager, O. Schenk, J. Thies, and G. Wellein. A recursive algebraic coloring technique for hardware-efficient symmetric sparse matrix-vector multiplication. *ACM*

Trans. Parallel Comput., 7(3), June 2020. ISSN 2329-4949. URL https://doi.org/10.1145/3399732.

[9] M. I. Alomoush. Microgrid dynamic combined power–heat economic-emission dispatch with deferrable loads and price-based energy storage elements and power exchange. *Sustainable Energy, Grids and Networks*, 26: 100479, 2021. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2021.100479.

[10] G. A. Antonopoulos, S. Vitiello, G. Fulli, and M. Masera. *Nodal Pricing in the European Internal Electricity Market*. Publications Office of the European Union, Luxembourg, 2020. ISBN 978-92-76-17571-1. doi: 10.2760/41018.

[11] H. Anzt, W. Sawyer, S. Tomov, P. Luszczek, I. Yamazaki, and J. Dongarra. Optimizing krylov subspace solvers on graphics processing units. In *Fourth International Workshop on Accelerators and Hybrid Exascale Systems (AsHES), IPDPS 2014*, Phoenix, AZ, 05-2014 2014. IEEE, IEEE.

[12] M. ApS. *MOSEK Fusion API for Python 10.1.9*, 2023. URL https://docs.mosek.com/latest/pythonfusion/index.html.

[13] I. Aravena, D. K. Molzahn, S. Zhang, C. G. Petra, F. E. Curtis, S. Tu, A. Wächter, E. Wei, E. Wong, A. Gholami, K. Sun, X. A. Sun, S. T. Elbert, J. T. Holzer, and A. Veeramany. Recent developments in security-constrained ac optimal power flow: Overview of challenge 1 in the arpa-e grid optimization competition, 2022.

[14] L. A. Arias, E. Rivas, F. Santamaria, and V. Hernandez. A review and analysis of trends related to demand response. *Energies*, 11(7), 2018. ISSN 1996-1073. URL https://www.mdpi.com/1996-1073/11/7/1617.

[15] S. Assad, R. Clark, D. Ershov, and L. Xu. Algorithmic pricing and competition: Empirical evidence from the german retail gasoline market. Working Paper, 2021.

[16] X. Bai and H. Wei. Semi-definite programming-based method for security-constrained unit commitment with operational and optimal power flow constraints. *IET Generation, Transmission & Distribution*, 3:182–197, 2009.

[17] A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye. Grid structural characteristics as validation criteria for synthetic networks. *IEEE Transactions on Power Systems*, 32(4):3258–3265, 2017.

[18] A. B. Birchfield, T. Xu, K. S. Shetye, and T. J. Overbye. Building synthetic power transmission networks of many voltage levels, spanning multiple areas. In *51st Hawaii International Conference on System Sciences*, 2018.

[19] M. Bollhöfer, A. Eftekhari, S. Scheidegger, and O. Schenk. Large-scale sparse inverse covariance matrix estimation. *SIAM Journal on Scientific Computing*, 41(1):A380–A401, 2019. URL https://doi.org/10.1137/17M1147615.

[20] M. Bollhöfer, O. Schenk, R. Janalik, S. Hamm, and K. Gullapalli. *Parallel Algorithms in Computational Science and Engineering*, chapter State-of-the-Art Sparse Direct Solvers, pages 3–33. Springer International Publishing, 2020. ISBN 978-3-030-43736-7. doi: 10.1007/978-3-030-43736-7_1.

[21] M. Bollhöfer, O. Schenk, R. Janalik, S. Hamm, and K. Gullapalli. State-of-the-art sparse direct solvers. pages 3–33, 2020. URL https://doi.org/10.1007/978-3-030-43736-7_1.

[22] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[23] T. Brijs, A. van Stiphout, S. Siddiqui, and R. Belmans. Evaluating the role of electricity storage by considering short-term operation in long-term planning. *Sustainable Energy, Grids and Networks*, 10:104–117, 2017. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2017.04.002.

[24] BSC Support Team. Greasy user guide. URL: https://github.com/BSC-Support-Team/GREASY, 2014. Version 2.14.

[25] R. Buizza and M. Leutbecher. The forecast skill horizon, 06/2015 2015. URL https://www.ecmwf.int/node/8450.

[26] Bundeskartellamt. *Fuel Sector Inquiry. Final Report in accordance with §32e GWB - May 2011 - Summary*, 2011.

[27] R. C. Burchett, H. H. Happ, and K. A. Wirgau. Large scale optimal power flow. *IEEE Transactions on Power Apparatus and Systems*, 101:3722–3732, 1982.

[28] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff. *Pyomo - Optimization Modeling in Python, 3rd Edition*. Springer, 2021.

[29] D. P. Byrne. Gasoline pricing in the country and the city. *Review of Industrial Organization*, 55:209–235, 2019.

[30] D. P. Byrne and N. de Roos. Learning to coordinate: A study in retail gasoline. *American Economic Review*, 109(2):591–619, 2019.

[31] D. P. Byrne, J. S. Nah, and P. Xue. Australia has the world's best petrol price data: Fuelwatch and fuelcheck. *Australian Economic Review*, 51(4): 564–577, 2018.

[32] C. G. Petra, I.Aravena Solis, N. Gawande, V. Amatya, A. Li, J. Li, S. Abhyankar, S. Peles, M. Schanen, K. Kim, A. Maldonado, M. Anitescu. ExaSGD - Optimization grid dynamics at Exascale. https://ecpannualmeeting.com/assets/overview/posters/exasgd_poster_3column_v4.pdf, 2020.

[33] J. Carpentier. Contribution to the economic dispatch problem. *Bulletin de la Société Française des Electriciens*, 8:431–447, 1962.

[34] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168, 2006.

[35] R. Castanias and H. Johnson. Retail gasoline price fluctuations. *Review of Economics and Statistics*, 75(1):171–174, 1993.

[36] A. Chandra and M. Tappata. Consumer search and dynamic price dispersion: an application to gasoline markets. *RAND Journal of Economics*, 42 (4):681–704, 2011.

[37] X. Chen. Large sample sieve estimation of semi-nonparametric models. In J. Heckman and E. Leamer, editors, *Handbook of Econometrics, Volume 6B*. North Holland (Elsevier), Amsterdam, Netherlands, 2007.

[38] K. Cheung, D. Gade, C. Silva-Monroy, S. M. Ryan, J.-P. Watson, R. J.-B. Wets, and D. L. Woodruff. Toward scalable stochastic unit commitment. *Energy Systems*, 6(3):417–438, Sep 2015. ISSN 1868-3975. URL https://doi.org/10.1007/s12667-015-0148-6.

[39] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint*, art. 1406.1078, 2014.

[40] R. Clark and J.-F. Houde. The effect of explicit communication on pricing: Evidence from the collapse of a gasoline cartel. *Journal of Industrial Economics*, 62(2):191–228, 2014.

[41] Cosmin Petra. HiOP User Guide version 0.3. https://github.com/LLNL/hiop/blob/master/doc/hiop_usermanual.pdf, 2017.

[42] S. Das, P. Acharjee, and A. Bhattacharya. Charging scheduling of electric vehicle incorporating grid-to-vehicle and vehicle-to-grid technology considering in smart grid. *IEEE Transactions on Industry Applications*, 57(2): 1688–1702, 2021. doi: 10.1109/TIA.2020.3041808.

[43] G. Deltas. Retail gasoline price dynamics and local market power. *Journal of Industrial Economics*, 56(3):613–628, 2008.

[44] H. W. Dommel and W. F. Tinney. Optimal power flow solutions. *IEEE Transactions on Power Apparatus and Systems*, 87:1866–1876, 1968.

[45] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, and I. Yamazaki. Accelerating numerical dense linear algebra calculations with gpus. *Numerical Computations with GPUs*, pages 1–26, 2014.

[46] J. Doyle, E. Muehlegger, and K. Samphantharak. Edgeworth cycles revisited. *Energy Economics*, 32(3):651–660, 2010.

[47] A. Eckert. Retail price cycles and response asymmetry. *Canadian Journal of Economics*, 35(1):52–77, 2002.

[48] A. Eckert and H. Eckert. Regional patterns in gasoline station rationalization in canada. *Journal of Industry, Competition and Trade*, 14:99–122, 2013.

[49] F. Y. Edgeworth. The pure theory of monopoly. In *Papers Relating to Political Economy*, volume 1, pages 111–142. MacMillan, London, 1925.

[50] S. Ekisheva and H. Gugel. North american ac circuit outage rates and durations in assessment of transmission system reliability and availability. In *2015 IEEE Power Energy Society General Meeting*, pages 1–5, 2015.

[51] ENTSO-E. Statistics and data, 2021. Accessed: 2021-12-01.

[52] I. Farhat and M. El-Hawary. Optimization methods applied for solving the short-term hydrothermal coordination problem. *Electric Power Systems Research*, 79(9):1308–1320, 2009. ISSN 0378-7796. URL https://www.sciencedirect.com/science/article/pii/S0378779609000947.

[53] E. A. Feinberg and D. Genethliou. *Load Forecasting*, pages 269–285. Springer US, Boston, MA, 2005. ISBN 978-0-387-23471-7. doi: 10.1007/0-387-23471-3_12. URL https://doi.org/10.1007/0-387-23471-3_12.

[54] J. Feinberg and H. P. Langtangen. Chaospy: An open source tool for designing methods of uncertainty quantification. *Journal of Computational Science*, 11:46–57, 2015. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2015.08.008.

[55] O. Foros and F. Steen. Vertical control and price cycles in gasoline retailing. *Scandinavian Journal of Economics*, 115(3):640–661, 2013.

[56] P. Fotis, S. Karkalakos, and D. Asteriou. The relationship between energy demand and real GDP growth rate: The role of price asymmetries and spatial externalities within 34 countries across the globe. *Energy Economics*, 66:69–84, 2017. ISSN 0140-9883. doi: https://doi.org/10.1016/j.eneco.2017.05.027.

[57] S. Frank and S. Rebennack. An introduction to optimal power flow: Theory, formulation, and examples. *IIE Transactions*, 48(12):1172–1197, 2016. doi: 10.1080/0740817X.2016.1189626. URL https://doi.org/10.1080/0740817X.2016.1189626.

[58] F. García-Muñoz, F. Díaz-González, and C. Corchero. A novel algorithm based on the combination of ac-opf and ga for the optimal sizing and location of ders into distribution networks. *Sustainable Energy, Grids and Networks*, 27:100497, 2021. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2021.100497.

[59] I. Gomes, R. Melicio, and V. Mendes. Dust effect impact on pv in an aggregation with wind and thermal powers. *Sustainable Energy, Grids and Networks*, 22:100359, 2020. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2020.100359.

[60] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.

[61] N. Guha, Z. Wang, M. Wytock, and A. Majumdar. Machine learning for ac optimal power flow, 2019.

[62] J. S. Guzmán-Feria, L. M. Castro, N. González-Cabrera, and J. Tovar-Hernández. Security constrained opf for ac/dc systems with power rescheduling by power plants and vsc stations. *Sustainable Energy, Grids and Networks*, 27:100517, 2021. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2021.100517.

[63] M. Haberg. Fundamentals and recent developments in stochastic unit commitment. *International Journal of Electrical Power & Energy Systems*, 109:38–48, 2019. ISSN 0142-0615. URL https://www.sciencedirect.com/science/article/pii/S014206151832547X.

[64] H. Haes Alhelou, M. E. Hamedani-Golshan, T. C. Njenda, and P. Siano. A survey on power system blackout and cascading events: Research motivations and challenges. *Energies*, 12(4), 2019. ISSN 1996-1073. URL https://www.mdpi.com/1996-1073/12/4/682.

[65] N. Hanford, V. Ahuja, M. Farrens, D. Ghosal, M. Balman, E. Pouyoul, and B. Tierney. Improving network performance on multicore systems: Impact of core affinities on high throughput flows. *Future Generation Computer Systems*, 56:277–283, 2016. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2015.09.012.

[66] S. Hanifi, X. Liu, Z. Lin, and S. Lotfian. A critical review of wind power forecasting methods—past, present and future. *Energies*, 13(15), 2020. ISSN 1996-1073. doi: 10.3390/en13153764. URL https://www.mdpi.com/1996-1073/13/15/3764.

[67] B. E. Hansen. Econometrics. Manuscript, 2020.

[68] S. Harasis, Y. Sozer, and M. Elbuluk. Reliable islanded microgrid operation using dynamic optimal power management. *IEEE Transactions on Industry Applications*, 57(2):1755–1766, 2021. doi: 10.1109/TIA.2020.3047587.

[69] W. E. Hart, J.-P. Watson, and D. L. Woodruff. Pyomo: Modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

[70] J. Haucap, U. Heimeshoff, and M. Siekmann. Fuel prices and station heterogeneity on retail gasoline markets. *Energy Journal*, 38(6):81–103, 2017.

[71] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[72] T. Holt, M. Igami, and S. Scheidegger. Replication package for: Detecting edgeworth cycles, Nov. 2023. URL https://doi.org/10.5281/zenodo.10126406.

[73] T. Holt, M. Igami, and S. Scheidegger. Detecting edgeworth cycles. *The Journal of Law and Economics*, 2023.

[74] T. Holt, S. Abhyankar, T. Kuruganti, O. Schenk, and S. Peles. Data-driven unit commitment refinement-a scalable approach for complex modern power grids. In T. X. Bui, editor, *Proceedings of the 57th Hawaii International Conference on System Sciences*, pages 3082–3092. HICSS Conference Office, January 2024.

[75] C. Hunziker, J. Lehmann, T. Keller, T. Heim, and N. Schulz. Sustainability assessment of novel transformer technologies in distribution grid applications. *Sustainable Energy, Grids and Networks*, 21:100314, 2020. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2020.100314.

[76] M. Ivaldi, B. Jullien, P. Rey, P. Seabright, and J. Tirole. The economics of tacit collusion. Technical report, Final Report for DG Competition, European Commission, 2003.

[77] A. Jain, S. P. Ong, W. Chen, B. Medasani, X. Qu, M. Kocher, M. Brafman, G. Petretto, G.-M. Rignanese, G. Hautier, D. Gunter, and K. A. Persson. Fireworks: a dynamic workflow system designed for high-throughput applications. *Concurrency and Computation: Practice and Experience*, 27(17): 5037–5059, 2015. ISSN 1532-0634. doi: 10.1002/cpe.3505. CPE-14-0307.R2.

[78] J. Kardoš, D. Kourounis, and O. Schenk. *Parallel Algorithms in Computational Science and Engineering*, chapter Structure-Exploiting Interior Point Methods, pages 63–93. Springer International Publishing, 2020. ISBN 978-3-030-43736-7. doi: 10.1007/978-3-030-43736-7_3.

[79] J. Kardoš, D. Kourounis, and O. Schenk. Two-level parallel augmented schur complement interior-point algorithms for the solution of security constrained optimal power flow problems. *IEEE Transactions on Power Systems*, 35(2):1340–1350, March 2020. ISSN 1558-0679. doi: 10.1109/TPWRS.2019.2942964.

[80] J. Kardoš, T. Holt, O. Schenk, V. Fazio, L. Fabietti, and F. Spazzini. High-performance data analytics techniques for power markets simulation. In *2021 International Conference on Smart Energy Systems and Technologies (SEST)*, pages 1–6, 2021.

[81] J. Kardoš, T. Holt, V. Fazio, L. Fabietti, F. Spazzini, and O. Schenk. Massively parallel data analytics for smart grid applications. *Sustainable Energy, Grids and Networks*, 31:100789, 2022. ISSN 2352-4677. URL https://www.sciencedirect.com/science/article/pii/S2352467722000972.

[82] T. Klein. Autonomous algorithmic collusion: Q-learning under sequential pricing. *RAND Journal of Economics*, 52(3):538–599, 2021.

[83] B. Knueven, J. Ostrowski, and J.-P. Watson. On mixed-integer programming formulations for the unit commitment problem. *INFORMS Journal on Computing*, 32(4):857–876, 2020.

[84] D. Kourounis, A. Fuchs, and O. Schenk. Toward the next generation of multiperiod optimal power flow solvers. *IEEE Transactions on Power Systems*, 33(4):4005–4014, July 2018. ISSN 0885-8950. doi: 10.1109/TPWRS.2017.2789187.

[85] T. Kristiansen. The flow based market coupling arrangement in Europe: Implications for traders. *Energy Strategy Reviews*, 27:100444, 2020. ISSN 2211-467X. doi: https://doi.org/10.1016/j.esr.2019.100444.

[86] T. Kumamoto, H. Aki, and M. Ishida. Provision of grid flexibility by distributed energy resources in residential dwellings using time-of-use pricing. *Sustainable Energy, Grids and Networks*, 23:100385, 2020. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2020.100385.

[87] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, and P. Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382

(6677):1416–1421, 2023. doi: 10.1126/science.adi2336. URL https://www.science.org/doi/abs/10.1126/science.adi2336.

[88] I. Lampropoulos, T. Alskaif, J. Blom, and W. van Sark. A framework for the provision of flexibility services at the transmission and distribution levels through aggregator companies. *Sustainable Energy, Grids and Networks*, 17:100187, 2019. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2018.100187.

[89] S. Lechtenböhmer and S. Samadi. Blown by the wind. replacing nuclear power in german electricity generation. *Environmental Science and Policy*, 25:234–241, 2013. ISSN 1462-9011. doi: https://doi.org/10.1016/j.envsci.2012.09.003. URL https://www.sciencedirect.com/science/article/pii/S1462901112001499.

[90] M. Leutbecher and T. Palmer. Ensemble forecasting, 02/2007 2007. URL https://www.ecmwf.int/node/10729.

[91] M. Lewis and M. Noel. The speed of gasoline price response in markets with and without edgeworth cycles. *Review of Economics and Statistics*, 93 (2):672–682, 2011.

[92] M. S. Lewis. Temporary wholesale gasoline price spikes have long-lasting retail effects: The aftermath of hurricane rita. *Journal of Law and Economics*, 52:581–605, 2009.

[93] M. S. Lewis. Price leadership and coordination in retail gasoline markets with price cycles. *International Journal of Industrial Organization*, 30: 342–351, 2012.

[94] M. Linder. Price cycles in the german retail gasoline market - competition or collusion? *Economics Bulletin*, 38(1):593–602, 2018.

[95] N. Malaya. The epyc cpu and instinct mi250x gpus in frontier. Frontier Training Workshop, Feb. 2023. URL: https://www.olcf.ornl.gov/wp-content/uploads/2-15-23-AMD-CPU-GPU-Frontier-Public.pdf.

[96] S. Martin. Market transparency and consumer search: Evidence from the german retail gasoline market. Working Paper, 2018.

[97] E. Maskin and J. Tirole. A theory of dynamic oligopoly, ii: Price competition, kinked demand curves, and edgeworth cycles. *Econometrica*, 56(3): 571–599, 1988.

[98] C. E. Murillo-Sánchez, R. D. Zimmerman, C. L. Anderson, and R. J. Thomas. Secure planning and operations of systems with stochastic sources, energy storage, and active demand. *IEEE Transactions on Smart Grid*, 4(4):2220–2229, 2013. doi: 10.1109/TSG.2013.2281001.

[99] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, MA, 2012.

[100] L. Musolff. Algorithmic pricing facilitates tacit collusion: Evidence from e-commerce. Working Paper, 2021.

[101] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, 2 edition, 2006. ISBN 978-0-387-30303-1.

[102] M. D. Noel. Edgeworth price cycles: Evidence from the toronto retail gasoline market. *Journal of Industrial Economics*, 55(1):69–92, 2007.

[103] M. D. Noel. Do edgeworth price cycles lead to higher or lower prices? *International Journal of Industrial Organization*, 42:81–93, 2015.

[104] M. D. Noel. Calendar synchronization of gasoline price increases. *Journal of Economics and Management Strategy*, 28:355–370, 2018.

[105] North America Electric Reliability Corporation (NERC). Reliability standards for the bulk electric systems of north america, July 2020. Available at https://www.nerc.com/pa/Stand/Pages/default.aspx.

[106] I. K. Nti, M. Teimeh, O. Nyarko-Boateng, and A. F. Adekoya. Electricity load forecasting: a systematic review. *Journal of Electrical Systems and Information Technology*, 7(1):13, 09 2020. ISSN 2314-7172. doi: 10.1186/s43067-020-00021-8. URL https://doi.org/10.1186/s43067-020-00021-8.

[107] U. S. D. of State. The long-term strategy of the united states: Pathways to net-zero greenhouse gas emissions by 2050. Technical report, United States Department of State and the United States Executive Office of the President, 2021.

[108] A. Olivier, D. G. Giovanis, B. Aakash, M. Chauhan, L. Vandanapu, and M. D. Shields. UQpy: A general purpose Python package and development environment for uncertainty quantification. *Journal of Computational Science*, 47:101204, 2020. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2020.101204.

[109] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26(1):80–113, 2007. doi: https://doi.org/10.1111/j.1467-8659.2007.01012.x.

[110] C. G. Petra and I. Aravena. Solving realistic security-constrained optimal power flow problems. *Operations Research*, 2023. to appear, preprint available at https://arxiv.org/pdf/2110.01669.pdf.

[111] G. Pirovano, P. Faggian, P. Bonelli, M. Lacavalla, P. Marcacci, and D. Ronzio. *Combining Meteorological and Electrical Engineering Expertise to Solve Energy Management Problems*, pages 133–154. Springer New York, New York, NY, 2014. ISBN 978-1-4614-9221-4. doi: 10.1007/978-1-4614-9221-4_6. URL https://doi.org/10.1007/978-1-4614-9221-4_6.

[112] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, MA, second edition, 1992.

[113] M. Rahmani, S. H. Hosseinian, and M. Abedi. Stochastic two-stage reliability-based security constrained unit commitment in smart grid environment. *Sustainable Energy, Grids and Networks*, 22:100348, 2020. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2020.100348.

[114] L. Rakai and W. Rosehart. Gpu-accelerated solutions to optimal power flow problems. In *47th Hawaii International Conference on System Sciences*, pages 2511–2516, 2014.

[115] S. Regev. *Preconditioning Techniques for Sparse Linear Systems*. PhD thesis, Stanford University, 2022.

[116] S. Regev, N.-Y. Chiang, E. Darve, C. G. Petra, M. A. Saunders, K. Swirydowicz, and S. Peles. Hykkt: A hybrid direct-iterative method for solving kkt linear systems. *Optimization Methods and Software*, 38(2):332–355, 2023. doi: 10.1080/10556788.2022.2124990.

[117] M. Reolon Scuzziato, E. Cristian Finardi, and A. Frangioni. Solving stochastic hydrothermal unit commitment with a new primal recovery technique based on lagrangian solutions. *International Journal of Electrical Power & Energy Systems*, 127:106661, 2021. ISSN 0142-0615. doi: https://doi.org/10.1016/j.ijepes.2020.106661.

[118] B. F. Ronalds, A. Wonhas, and A. Troccoli. *A New Era for Energy and Meteorology*, pages 3–16. Springer New York, New York, NY, 2014. ISBN 978-1-4614-9221-4. doi: 10.1007/978-1-4614-9221-4_1. URL https://doi.org/10.1007/978-1-4614-9221-4_1.

[119] B. Saravanan, S. Das, S. Sikri, and D. P. Kothari. A solution to the unit commitment problem—a review. *Frontiers in Energy*, 7(2):223–236, Jun 2013. ISSN 2095-1698. URL https://doi.org/10.1007/s11708-013-0240-3.

[120] J. D. Scargle. Studies in astronomical time series analysis. ii. statistical aspects of spectral analysis of unevenly spaced data. *Astrophysical Journal*, 263:835–853, 1982.

[121] O. Schenk and K. Gärtner. On fast factorization pivoting methods for sparse symmetric indefinite systems. *ETNA. Electronic Transactions on Numerical Analysis*, 23:158–179, 2006.

[122] O. Schenk, K. Gärtner, and W. Fichtner. Efficient sparse lu factorization with left-right looking strategy on shared memory multiprocessors. *BIT Numerical Mathematics*, 40:158–176, 2000. doi: 10.1023/A:1022326604210.

[123] D. Schneider. The exascale era is upon us: The frontier supercomputer may be the first to reach 1,000,000,000,000,000,000 operations per second. *IEEE Spectrum*, 59(1):34–35, 2022.

[124] M. Siekmann. Characteristics, causes, and price effects: Empirical evidence of intraday edgeworth cycles. Technical report, DICE Discussion Paper, No. 252, 2017.

[125] J. C. Smith, M. O'Malley, D. Osborn, R. Piwko, and R. J. Thomas. R&d requirements for integration of wind generation. In *45th Hawaii International Conference on System Sciences*, 2012.

[126] X. Su, C. He, T. Liu, and L. Wu. Full parallel power flow solution: A gpu-cpu-based vectorization parallelization and sparse techniques for newton–raphson implementation. *IEEE Transactions on Smart Grid*, 11(3):1833–1844, 2020.

[127] D. Suleimenova, H. Arabnejad, W. N. Edeling, et al. Tutorial applications for verification, validation and uncertainty quantification using VECMA

toolkit. *Journal of Computational Science*, 53:101402, 2021. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2021.101402.

[128] K. Swirydowicz, N. Koukpaizan, S. Abhyankar, and S. Peles. Towards efficient alternating current optimal power flow analysis on graphical processing units. In *29th International Conference on Information, Communication and Automation Technologies (ICAT23)*, June 2023.

[129] Swissgrid. Production and consumption, July 2021. URL https://www.swissgrid.ch/en/home/operation/grid-data/generation.html.

[130] J. Treibig, G. Hager, and G. Wellein. LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments. In *Proceedings of the 39th International Conference on Parallel Processing Workshops*, page 207–216. IEEE Computer Society, 2010. ISBN 9780769541570. doi: 10.1109/ICPPW.2010.38.

[131] P. Valero-Lara, I. Martínez-Pérez, R. Sirvent, X. Martorell, and A. J. Peña. NVIDIA GPUs Scalability to Solve Multiple (Batch) Tridiagonal Systems. Implementation of cuThomasBatch. In *Parallel Processing and Applied Mathematics - 12th International Conference (PPAM)*, 2017.

[132] W. van Ackooij, I. Danti Lopez, A. Frangioni, F. Lacalandra, and M. Tahanan. Large-scale unit commitment under uncertainty: an updated literature survey. *Annals of Operations Research*, 271(1):11–85, 2018. doi: 10.1007/s10479-018-3003-z.

[133] W. van Ackooij, I. Danti Lopez, A. Frangioni, F. Lacalandra, and M. Tahanan. Large-scale unit commitment under uncertainty: an updated literature survey. *Annals of Operations Research*, 271(1):11–85, Dec 2018. ISSN 1572-9338. URL https://doi.org/10.1007/s10479-018-3003-z.

[134] J. T. VanderPlas. Understanding the lomb–scargle periodogram. *Astrophysical Journal Supplement Series*, 236(16):1–28, 2018.

[135] P. A. Verwiebe, S. Seim, S. Burges, L. Schulz, and J. Müller-Kirchenbauer. Modeling energy demand – a systematic literature review. *Energies*, 14 (23), 2021. ISSN 1996-1073. doi: 10.3390/en14237859. URL https://www.mdpi.com/1996-1073/14/23/7859.

[136] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006. ISSN 1436-4646. URL https://doi.org/10.1007/s10107-004-0559-y.

[137] J. Wang and C. G. Petra. An adaptive sampling sequential quadratic programming method for nonsmooth stochastic optimization with upper-c^2 objective. *SIAM Journal On Optimization*, 2023. to appear, preprint available at https://arxiv.org/abs/2204.09631.

[138] Z. Wang. Collusive communication and pricing coordination in a retail gasoline market. *Review of Industrial Organization*, 32:35–52, 2008.

[139] Z. Wang, U. Munawar, and R. Paranjape. Stochastic optimization for residential demand response with unit commitment and time of use. *IEEE Transactions on Industry Applications*, 57(2):1767–1778, 2021. doi: 10.1109/TIA.2020.3048643.

[140] N. Wills-Johnson and H. Bloch. The shape and frequency of edgeworth price cycles in an australian retail gasoline market. Working Paper, Curtin University of Technology, 2010.

[141] L. A. Wilson, J. M. Fonner, J. Allison, O. Esteban, H. Kenya, and M. Lerner. Launcher: A simple tool for executing high throughput computing workloads. *Journal of Open Source Software*, 2(16):289, 2017. doi: 10.21105/joss.00289.

[142] W. Xia and C. A. Shoemaker. Improving the speed of global parallel optimization on PDE models with processor affinity scheduling. *Computer-Aided Civil and Infrastructure Engineering*, 37(3):279–299, 2022. doi: https://doi.org/10.1111/mice.12737.

[143] T. Xu, A. B. Birchfield, K. M. Gegner, K. S. Shetye, and T. J. Overbye. Application of large-scale synthetic power system models for energy economic studies. In *50th Hawaii International Conference on System Sciences*, 2017.

[144] I. Yamazaki, H. Anzt, S. Tomov, M. Hoemmen, and J. Dongarra. Improving the performance of ca-gmres on multicores with multiple gpus. In *IPDPS 2014*, Phoenix, AZ, 05-2014 2014. IEEE.

[145] D. Yang, W. Wang, C. A. Gueymard, T. Hong, J. Kleissl, J. Huang, M. J. Perez, R. Perez, J. M. Bright, X. Xia, D. van der Meer, and I. M. Peters. A

review of solar forecasting, its dependence on atmospheric sciences and implications for grid integration: Towards carbon neutrality. *Renewable and Sustainable Energy Reviews*, 161:112348, 2022. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2022.112348. URL https://www.sciencedirect.com/science/article/pii/S1364032122002593.

[146] Y. Yang and L. Wu. Machine learning approaches to the unit commitment problem: Current trends, emerging challenges, and new strategies. *The Electricity Journal*, 34(1):106889, 2021. ISSN 1040-6190. Special Issue: Machine Learning Applications To Power System Planning And Operation.

[147] A. B. Yoo, M. A. Jette, and M. Grondona. SLURM: Simple linux utility for resource management. In D. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 44–60, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-39727-4.

[148] Q. P. Zheng, J. Wang, and A. L. Liu. Stochastic optimization for unit commitment—a review. *IEEE Transactions on Power Systems*, 30(4):1913–1924, 2015.

[149] J. Zhu. *Optimization of Power System Operation*. Wiley-IEEE, Piscataway, NJ, 2009.

[150] P. R. Zimmerman, J. M. Yun, and C. T. Taylor. Edgeworth price cycles in gasoline: Evidence from the united states. *Review of Industrial Organization*, 42:297–320, 2013.

[151] R. D. Zimmerman and C. E. Murillo-Sánchez. MATPOWER Optimal Scheduling Tool (MOST) User's Manual, Oct. 2020.

[152] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on Power Systems*, 26(1):12–19, Feb 2011. ISSN 0885-8950. doi: 10.1109/TPWRS.2010.2051168.

[153] K. Świrydowicz, E. Darve, W. Jones, J. Maack, S. Regev, M. A. Saunders, S. J. Thomas, and S. Peleš. Linear solvers for power grid optimization problems: A review of gpu-accelerated linear solvers. *Parallel Computing*, 111:102870, July 2022. ISSN 0167-8191. doi: 10.1016/j.parco.2021.102870. URL http://dx.doi.org/10.1016/j.parco.2021.102870.