To the Graduate Council:
I am submitting herewith a dissertation written by Marzieh Bakhshi entitled "The Battery Charging Problem and Linear Programs with Random Coefficient Matrix." I have examined the final paper copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Industrial and System Engineering.

Jame Ostrowski, Major Professor

We have read this dissertation
and recommend its acceptance:

Mingzhou Jin

Hugh Medal

Hector Pulgar

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

To the Graduate Council:
I am submitting herewith a dissertation written by Marzieh Bakhshi entitled "The Battery Charging Problem and Linear Programs with Random Coefficient Matrix." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Industrial and System Engineering.

Jame Ostrowski, Major Professor

We have read this dissertation
and recommend its acceptance:

Mingzhou Jin

Hugh Medal

Hector Pulgar

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

The Battery Charging Problem and Linear Programs with Random Coefficient Matrix

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Marzieh Bakhshi

December 2024

This dissertation is dedicated to my mom and my sisters who have been sources of strength, support, and motivation for me throughout my life.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. James Ostrowski, for his support and guidance over the past four years. I am deeply grateful to my husband for his support. Without his encouragement and support, this dissertation would not have been possible.

I would also like to thank Dr. Mingzhou Jin and Dr. Hugh Medal for the valuable lessons I gained from their courses. Their contributions have significantly enriched my academic journey.

Finally, I appreciate the Department of Industrial and Systems Engineering for providing me the opportunity to be in the Ph.D. program.

Abstract

This dissertation includes two main topics. First it concerns the problem of scheduling a generic battery energy storage in order to maximize the (expected) arbitrage as a result of optimizing charge and discharge decisions. We draw tools from network optimization, stochastic optimization, and reinforcement learning to analyze the problem and propose alternative solution methods.

We reformulate a standard mixed integer linear model of a battery as a shortest path problem and as result we propose a polynomial algorithm to solve the problem in deterministic setting. Then we study the optimization of a battery under the uncertainty of charge and discharge prices. We compare the performance of various policy generation methods in Markovian or near-Markovian settings to optimize expected arbitrage under price uncertainty. Our simulations leverage both a Markov model of prices and historical data to evaluate the relative performance of traditional stochastic optimization methods, such as mean value problem, two-stage stochastic programming and stochastic dynamic programming, against an approximate model-free method, Q-learning.

The second topic of the dissertation, explores linear programs with random coefficient matrix. We first provide theoretical and computational results on the magnitude of optimal objective of linear programs with random coefficient matrix. Then we adapt an algorithm to find near-optimal solution of such programs. At last, we study two stage stochastic programs with random technology matrix and propose

an algorithm for decoupling the first and second stages of these programs when the technology matrix is standard Gaussian.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter we discuss the motivations for our study presented in the subsequent chapters. We first present some background and common methods for optimization of a battery energy storage participating in arbitrage in both deterministic and stochastic settings. Then we discuss two stage stochastic programs and the motivation for proposing a decomposition algorithm for it.

1.1 Optimization of an energy storage participating in arbitrage

The utilization and optimization of energy storage systems for a variety of purposes are fundamental challenges in energy system applications. The deployment of renewable energy is expected to continue growing as technology advances, costs decrease, and the urgency to address climate change increases. Energy storage is widely regarded as essential for integrating large amounts of variable renewable energy into the grid. Various energy storage technologies are currently being developed, each with different characteristics. Among these, battery energy storage systems are expected to play a crucial and significantly larger role (Child et al., 2018) (Aghahosseini et al., 2020). Energy storage systems are valuable not only for grid

balancing but also for a variety of other applications. They can be used to take advantage of daily fluctuations in electricity prices (arbitrage) (Mercier et al., 2023), delay or avoid the costs of network upgrades, and smooth the intermittent supply of renewable energy generators, thus enhancing the value of their energy.

The first main topic of this dissertation concerns the optimization of an energy storage for arbitrage in both deterministic and stochastic setting. We consider a basic storage system, where the storage unit could be a battery, a pumped hydro facility, a compressed air storage system, or alternative storage forms like a building that can be pre-cooled to decrease air conditioning use during peak times.

In a deterministic setting, in chapter 2 we study a mixed integer linear formulation of a battery and the convex hull of the feasible region. We explore the vertices of the convex hull and drawing tools from network optimization, we reformulate the mixed integer linear formulation as a shortest path problem. This reformulation yields a polynomial algorithm for solving the battery charging problem.

A common method for estimating arbitrage profits involves assuming perfect foresight of electricity prices, which inevitably leads to an overestimation of achievable profits in practice. More advanced methods utilizing stochastic optimization techniques have also been suggested to simulate the scheduling decisions more realistically. There are many sources of uncertainty while optimizing a battery for arbitrage: real time prices, amount of energy produces by renewables, demand, rate of energy conversion loss. This study is only concerns about the uncertainty of prices.

The objective of the study presented in chapter 3, is the performance comparison of various optimization methods of policy generation in Markovian or near Markovian setting in order to optimize expected arbitrage under price uncertainty. Our simulations utilize both a Markov model of prices and historical data to evaluate the relative performance of traditional stochastic optimization (such as two stage stochastic programming and stochastic dynamic programming) and an approximate model-free method, Q-learning, which has gained significant popularity in recent literature. The construction of state space for dynamic programming and Q-learning

methods is a subtle concept (Powell and Meisel, 2015). In our study we use Markovian or second order Markovian information of prices to describe the state space.

1.2 Random coefficient matrices in optimization programs

The second major focus of this dissertation is the investigation of optimization problems involving randomly generated coefficient matrices. In chapter 4 we first examine linear programs (LPs) with a random coefficient matrix. Secondly, we explore two stage stochastic programs with a random technology matrix.

LPs with the coefficient matrix generated randomly, have been actively studied in various contexts including in the context of estimating the average-case complexity of the simplex method. Despite significant progress in that direction, fundamental questions regarding properties of random LPs remain open. In this work we are interested in estimating the objective function of random LPs in the limit. In chapter 4 we setup a generic LP with certain assumptions and empirically study the limiting distribution of the optimal objective and provide an algorithm to find a near optimal solution of such programs.

When dealing with large two stage stochastic programs, the one practical way to solve them is through the use of decomposition algorithms. These algorithms break down complex optimization problems into more manageable subproblems, making them essential for large scale applications. Decomposition algorithms have been extensively studied for two-stage stochastic programs. Notable methods include Benders decomposition, Lagrangian decomposition, and their various adaptations.

Current decomposition algorithms are typically designed for general technology matrices. In our study, we focus specifically on two stage stochastic programs with Gaussian technology matrices. This focus allows us to exploit the unique properties of Gaussian matrices to develop more efficient and tailored decomposition algorithms.

In particular, we propose a method that take advantage of the rotational invariance of standard Gaussian matrices. This characteristic enables us to design an algorithm that decouples the first and second stages of the stochastic programs. By doing so, we aim to enhance the computational efficiency and scalability of solving two stage stochastic programs with Gaussian technology matrices.

Chapter 2

Battery charging problem

This chapter is based on a manuscript prepared for publication by Marzieh Bakhshi and James Osrowski:

A Polynomial Algorithm for the Lossless Battery Charging Problem, Marzieh Bakhshi, James Ostrowski (2023). Submitted.

A preprint of this paper is available at https://link.springer.com/article/10.1007/s12667-023-00621-z.

2.1 Introduction

Energy storage is an increasingly attractive solution to reduce electricity costs and carbon footprints and to increase energy power systems' flexibility and reliability. In recent years the usage of energy storage has increased significantly as it has becoming more economically viable due to declining technological costs and surging renewable penetration into power grids. An energy storage system also provides energy arbitrage, which refers to purchasing, via charging, the energy when energy prices are low and selling, via discharging, energy when the prices are high. In order to maximize this revenue, a battery storage requires a proper management strategy able to make charge/discharge decisions according to price signals. Making these

decisions efficiently in terms of time and computational efforts becomes more critical when energy storage is integrated within more complex optimization problems.

The standard mixed integer linear programming (MILP) model for the battery charging problem that participates in energy arbitrage can be formulated as

$$\max \quad \sum_{t=1}^{T}(P_t^d d_t - P_t^c c_t) \tag{2.1a}$$

$$\text{s.t.} \quad s_t = \eta s_{t-1} + c_t - d_t \qquad \forall t \in \mathcal{T}, \tag{2.1b}$$

$$s_0 = S_{\text{init}}, \tag{2.1c}$$

$$s_T = S_{\text{final}}, \tag{2.1d}$$

$$\underline{S} \leq s_t \leq \overline{S} \qquad \forall t \in \mathcal{T}, \tag{2.1e}$$

$$0 \leq c_t \leq \overline{C} u_t \qquad \forall t \in \mathcal{T}, \tag{2.1f}$$

$$0 \leq d_t \leq \overline{D} v_t \qquad \forall t \in \mathcal{T}, \tag{2.1g}$$

$$u_t + v_t = 1 \qquad \forall t \in \mathcal{T}, \tag{2.1h}$$

$$u_t, v_t \in \{0, 1\} \qquad \forall t \in \mathcal{T}. \tag{2.1i}$$

The above formulation is a state of charge (SOC) formulation with mutually exclusive charge and discharge modes over the time interval $[0, T]$. This interval is discretized into unit time periods resulting in the set $\mathcal{T} = \{1, 2, ..., T\}$. The storage parameters and variables are given in Table 2.1. The charging and discharging prices at each time t is denoted by P_t^c and P_t^d, respectively.

In our study we assume the storage efficiency η in the above formulation is one. This means that the we do not lose energy across time periods. We call this restriction the *lossless battery charging problem*. This assumption is necessary for the convex hull description and the polynomial time algorithm: we do not know if there are polynomial time algorithms when $\eta \neq 1$. The formulation (2.1) is at first glance very simple, and one might expect the solution would be easy to come by. If all of the prices are positive, there is no incentive for both c_t and d_t to be nonzero in any linear

Table 2.1: Parameters and variables of formulation (2.1).

Parameters

S_{init}	initial state of charge
$\overline{S}$	maximum storage capacity
$\underline{S}$	minimum storage capacity
$\overline{C}$	maximum charging capacity
$\overline{D}$	maximum discharging capacity
η	storage efficiency

Variables

s_t	state of charge at time period t
c_t	quantity of energy charged at time t
d_t	quantity of energy discharged at time t
u_t	binary charging status at time t
v_t	binary discharging status at time t

relaxation solution, meaning that optimal solution to the linear programming solution will have the same objective value as the MILP. However, the polyhedron defining the feasible region is quite complicated. Observe that there are 2^T many vertices that represent the action of simply doing nothing throughout the entire interval, where $S_{\text{init}} = S_{\text{final}}$ and $c_t = d_t = 0$ for all t, and u_t (v_t) is either zero (one) or one (zero) arbitrarily. This complicated structure can impact solution times when prices become more general.

It should be noted that problem (2.1) abstracts away the charging and discharging inefficiencies of the battery. Assume η_c and η_d denote charge and discharge efficiencies. Typically when charging, the grid may observe that the battery is charging at a rate of x, but the battery is only increasing its state of charge by $\eta_c x$, for some $0 \leq \eta_c \leq 1$. Similarly with discharging and discharge efficiency $0 \leq \eta_d \leq 1$. However, this can be abstracted by including the inefficiencies into the price terms. Suppose that grid price was P, then discharging d units from the battery would inject $\eta_d d$ units to the grid, giving a price of $\eta_d P$ per unit discharged. Similarly, charging c requires buying $\frac{c}{\eta_c}$ from the grid, so the price would be $\frac{P}{\eta_d}$.

2.2 Literature Review

To avoid some complexity, some studies (Shen et al., 2020; Joshi et al., 2021) reformulate binary variables with weaker yet linear constraints to model an energy storage as a linear program. However, these studies are not viable when energy prices are negative. During negative price periods the storage operator is paid to consume electric energy, and will therefore attempt to charge the storage device as much as possible. If there is more than enough energy to fully charge the storage device, the LP solution is incentivized to charge and discharge simultaneously. Other approaches in literature to address this problem of fractional solutions are to ignore the non-simultaneous charge and discharge constraint during negative price periods (Brijs et al., 2016) or exclude negative prices in the model in the first place (Pozo

et al., 2014). However, this is problematic since with the increasing penetration of renewable energy, the presence of negative prices are becoming more frequent (Seel et al., 2021).

While the above methods are interested in solving the battery charging problem directly, there is also a need to understand the convex hull of battery charging solutions. Often times a battery is a small component of a much larger system. In this context, branching on variables associated with the battery may have very little impact in changing the dual bound compared to variables associated with larger components; however, doing so may be required to arrive at an integer solution. By studying the optimality conditions of the battery charging problem, we are able to recast the battery charging problem as a shortest path problem, yielding both polynomial-time algorithms as well as convex hull descriptions.

Not much is known about the complexity of the energy storage problem. Adding non-zero and time-dependent minimum and maximum charge and discharge limits ($\underline{C}_t u_t \leq c_t \leq \overline{C}_t u_t$ and $\underline{D}_t v_t \leq d_t \leq \overline{D}_t v_t$) would make the above problem NP-hard (Pan et al., 2022). To the authors' knowledge, however, there have been no other results related to problem complexity.

While one often thinks of batteries when thinking of energy storage, we note that there are far more diverse types of energy storage. Large pumped storage hydro facilities operate in the same way, charging by pumping water into a reservoir and discharging by running the stored water through turbines. Fuel-constrained generators can be thought of as a (more complicated) storage device by considering delivery of the fuel as charging and the burning of fuel as discharging. At the micro-grid level, HVAC devices act as energy storage, where "charging" on a hot summer day is done by cooling the house to below the desired temperature in advance (Vedullapalli et al., 2019). Throughout this paper, however, we will use the terms energy storage device and battery interchangeably.

The rest of this paper is organized as follows. The optimality conditions and solution methods are given in section 2. Despite polynomial run times and convex

hulls, the approach is likely not computationally tractable for real-world problems. We simplify the model by adding realistic assumptions on prices in Section 3. In Sections 4 and 5 we discuss our polyhedral and computational results. We conclude the paper in Section 6.

2.3 Optimality Conditions and Algorithms

In the following we study some properties of the convex hull of the formulation (2.1). We first restrict our study to a special type of vertices.

Definition 2.1. *We say a vertex* $(s_t, c_t, d_t, u_t, v_t)_{t\in\mathcal{T}}$ *in the convex hull of the formulation (2.1) is min or max on the boundary (MOB) if state of the charge meets its minimum or maximum bounds only on the boundary (and not in the interior). For the time interval* $[0, T]$ *this means*

$$s_0, s_T \in \{S_{init}, S_{final}\}, \tag{2.2}$$

$$\underline{S} < s_t < \overline{S} \quad \forall t \in (0, T). \tag{2.3}$$

Theorem 2.1. *For a MOB vertex A in the convex hull of the formulation (2.1) there exists at most one time period* t *with* $0 < t < T$ *when the battery is either (strictly) partially charging or partially discharging; that is, there is at most one* t *such that either* $0 < c_t < \overline{C}$ *or* $0 < d_t < \overline{D}$.

Proof. Proof: Aiming at a contradiction, assume there are two time periods t and t' with $0 < t, t' < T$ and without loss of generality a partial charge at time t and a partial discharge at time t'. Since A is a MOB vertex, there is an $\epsilon > 0$ with $\underline{S} < s_t \pm \epsilon < \overline{S}$ for all $t \in (0, T)$. We also have $0 \leq c_t \pm \epsilon \leq \overline{C}$ and $0 \leq d_{t'} \pm \epsilon \leq \overline{D}$ (otherwise we can choose a smaller value for ϵ). Now, consider solutions B and C

with almost identical charging and discharging actions as A, except that:

$$c_t^B = c_t^A + \epsilon \ , \quad d_{t'}^B = d_{t'}^A + \epsilon \tag{2.4}$$

$$c_t^C = c_t^A - \epsilon \ , \quad d_{t'}^C = d_{t'}^A - \epsilon. \tag{2.5}$$

As a result of our choice in ϵ, we have that B and C are feasible schedules and $A = \frac{1}{2}B + \frac{1}{2}C$. This contradicts with A being a vertex. □

2.4 Optimizing Over MOB Vertices

Consider a MOB vertex in interval $[0, T]$. As a result of the Theorem 2.1 possible optimal actions on the entire interval for this vertex are full charge or discharges, at most one partial charge or discharge and no charge and discharges (do nothings). Let n^c, n^d and n^n denote the number of time periods with full charge, full discharge, and do nothing, respectively. Then Theorem 2.1 implies that $n^c+n^d+n^n \in \{T-1, T\}$. Given these numbers, we compute the amount of the non-zero partial charge or discharge in the interval by the formula

$$p = S_{\text{final}} + \overline{D}n^d - \overline{C}n^c - S_{\text{init}}, \tag{2.6}$$

taking into account that $n^c + n^d + n^n = T - 1$. Note that $p \in \left(-\overline{D}, \overline{C}\right) \setminus \{0\}$, the negative values correspond to partial discharges while positive values correspond to partial charges. Also, the order that the charges and discharges are performed does not effect the calculation of the value p. This is due to the assumption $\eta = 1$. It is possible for p to take multiple distinct values for the given vertex, depending on the arrangement of the numbers n^c, n^d and n^n. In such cases, we consider each arrangement separately and calculate the value of p for each one. For example let $T = 4$, $S_{\text{init}} = \underline{S} = 0$, $S_{\text{final}} = \overline{S} = 10$, $\overline{C} = 6$, and $\overline{D} = 3$. If $n^c = 2$, $n^d = 1$ and $n^n = 0$, then according to the formula (2.6) $p = 1$ corresponding to partial charge.

If $n^c = 2$, $n^d = 0$ and $n^n = 1$, then $p = -2$ which corresponds to partial discharge of value 2. We denote the set of all possible values of partial charge/discharge by $\mathcal{P}$. In the following Lemma we show that the size of the set $\mathcal{P}$ is linear in T.

Lemma 2.1.1. *The number of unique values of p amongst all MOB vertices is linear in terms of T.*

Proof. Proof: We show that for a fixed value of $n^n \in \{0, 1, ..., T\}$, there is at most one possible value for p. Towards a contradiction, let n_1^c and n_1^d be values such that

$$-\overline{D} < p_1 = S_{\text{final}} + \overline{D}n_1^d - \overline{C}n_1^c - S_{\text{init}} < \overline{C}, \tag{2.7}$$

and let $n_2^c \neq n_1^c$ and n_2^d be values such that

$$-\overline{D} < p_2 = S_{\text{final}} + \overline{D}n_2^d - \overline{C}n_2^c - S_{\text{init}} < \overline{C} \tag{2.8}$$

with $n_1^c + n_1^d = n_2^c + n_2^d = T - 1 - n^n$. Letting $n_2^c = n_1^c + j$ for $j \in \mathcal{T}$, we have $n_2^d = n_1^d - j$. Then

$$-\overline{D} < p_2 = S_{\text{final}} + \overline{D}(n_1^d - j) - \overline{C}(n_1^c + j) - S_{\text{init}} < \overline{C},$$

which leads to

$$(j-1)\overline{D} + j\overline{C} < p_1 < (j+1)\overline{C} + j\overline{D}, \quad j \in \mathcal{T}.$$

This is inconsistent with (2.7). □ □

Shortest path problem for MOB vertices

We represent the space of all possible MOB solutions of the formulation (2.1) as a directed acyclic graph. Each node denotes the number of full charges, number of full discharges, number of do nothings, and partial charge/discharges that have occurred

up to time t. we index the elements of the set $\mathcal{P}$ defined above by i, i.e., let p_i be the ith element in the set $\mathcal{P}$. We represent the nodes of the graph by

$$(n^c, n^d, n^n, \mathbf{0})_t \ \forall n^c, n^d, n^n \in \{0, \ \ldots, \ T\}, \ t = n^c + n^d + n^n \leq T - 1 \tag{2.9}$$

$$(n^c, n^d, n^n, e_i)_t \ \forall n^c, n^d, n^n \in \{0, \ \ldots, \ T\}, \ t - 1 = n^c + n^d + n^n \leq T - 1. \tag{2.10}$$

The zero vector, $\mathbf{0}$, of dimension $|\mathcal{P}|$ indicates that no partial charge/discharge has performed, whereas the vector e_i indicates that the ith partial charge in the set $\mathcal{P}$ has been performed. The arcs entering a node associated with time t, represents a possible action occurring at time t. To represent a full charge at time t, we add arcs leaving nodes of type $(n^c, n^d, n^n, \mathbf{0})_{t-1}$ and $(n^c, n^d, n^n, e_i)_{t-1}$ to nodes $(n^c + 1, n^d, n^n, \mathbf{0})_t$ and $(n^c+1, n^d, n^n, e_i)_t$, respectively with the cost of $-\overline{C}P_t^c$. Construction of arcs reflecting full discharges, do nothings, and partial charge/discharges are analogous. We then add $\mathfrak{s} = (0, 0, 0, \mathbf{0})$ as the start node and $\mathfrak{t}$ as terminal node with arcs of respective cost leaving $\mathfrak{s}$ and entering $\mathfrak{t}$. Now the optimal MOB vertex can be found by solving the shortest $\mathfrak{s} - \mathfrak{t}$ path in the described graph, where the objective value is equal to the negative of the shortest path length. We enforce the constraints such as the initial and final state of the charge on the boundary of the time interval $[0, T]$ and the minimum and maximum state of the charges on interior time periods by removing the infeasible nodes from the graph through the following pruning rules. We remove nodes of types:

(a) $(n^c, n^d, n^n, \mathbf{0})^t$ if $S_{\text{init}} + \overline{C}n_t^c - \overline{D}n_t^d \notin (\underline{S}, \overline{S})$,

(b) $(n^c, n^d, n^n, e_i)^t$ if $S_{\text{init}} + \overline{C}n_t^c - \overline{D}n_t^d + p_i \notin (\underline{S}, \overline{S})$,

(c) $(n^c, n^d, n^n, e_i)^T$ if $S_{\text{init}} + \overline{C}n^c - \overline{D}n^d + p_i \neq S_{\text{final}}$,

(d) $(n^c, n^d, n^n, \mathbf{0})^t$ if $S_{\text{init}} + \overline{C}n_t^c - \overline{D}n_t^d \notin [S_{\text{final}} - \overline{C}(T - t), S_{\text{final}} + \overline{D}(T - t)]$,

(e) $(n^c, n^d, n^n, e_i)^t$ if $S_{\text{init}} + \overline{C}n_t^c - \overline{D}n_t^d + p_i \notin [S_{\text{final}} - \overline{C}(T - t), S_{\text{final}} + \overline{D}(T - t)]$.

Example 2.2. Let $T = 3$, $S_{\text{init}} = \underline{S} = 0$, $S_{\text{final}} = \overline{S} = 10$, $\overline{C} = 6$, and $\overline{D} = 4$. The charge and discharge prices at each time are: $P_1^c = 3$, $P_2^c = 5$, $P_3^c = 7$, $P_1^d = 2$, $P_2^d = 4$, $P_3^d = 5$.

We find all possible partial charge values, $|\mathcal{P}|$, by enumerating over the possible values of n^n taking into account the equation $n^n + n^c + n^d = T - 1$. For a fixed value of n^n, we find a unique value of partial charge/discharge that satisfies

$$p = 10 + 6n^d - 4n^c \ , \quad -4 < p < 6.$$

For $n^n = 0$ (and $n^c + n^d = 2$), we have $p = -2$ for $n^c = 2$ and $n^d = 0$. For $n^n = 1$ (and $n^c + n^d = 1$), we find a partial charge of value $p = 4$ for $n^c = 1$ and $n^d = 0$. The values $n^n = 2, 3$ are impossible since they lead to $n^c + n^d = 0$ and $n^c + n^d = -1$. Hence in this case $\mathcal{P} = \{-2, 4\}$. Figure 2.1 illustrates the associated directed graph. The $\mathfrak{s} - \mathfrak{t}$ shortest path,

$$\mathfrak{s} \to (1, 0, 0, \mathbf{0}) \to (1, 0, 1, \mathbf{0}) \to (1, 0, 1, e_2) \to \mathfrak{t},$$

represents the optimal schedule: a full charge at time 1, do nothing at time 2, and partial charge of value 4 at time 3. The optimal cost of this path is -46. We prove this optimality in the following theorem.

Theorem 2.3. *Solving $\mathfrak{s} - \mathfrak{t}$ shortest path problem on the above graph provides a MOB vertex with the highest objective value relative to other MOBs.*

Proof. Proof: We show a one-to-one correspondence between $\mathfrak{s}-\mathfrak{t}$ paths in the directed graph and MOB vertices. First we prove that given a MOB vertex, there is an $\mathfrak{s} - \mathfrak{t}$ path with the opposite cost. Let $A = (s_t, c_t, d_t, u_t, v_t)_{t\in\mathcal{T}}$ be a MOB vertex with objective value of M_A. By Theorem 2.1, there is at most one time period $t_p \in \mathcal{T}$ such that either $0 < c_{t_p} < \overline{C}$ or $0 < d_{t_p} < \overline{D}$. Without loss of generality, we can assume that there is a time interval, t_p, with a partial charge, denoted by p_1. The implication of Theorem 2.1 is that for all $t \in \mathcal{T} \setminus \{t_p\}$ with $u_t = 1$ we have $c_t = \overline{C}$ and for all

$t \in \mathcal{T}$ with $v_t = 1$ we have $d_t = \overline{D}$. Then the objective value of vertex A is as follows,

$$M_A = \sum_{t \in \mathcal{T}: d_t = \overline{D}} \overline{D} P_t^d \quad - \sum_{t \in \mathcal{T} \setminus \{t_p\}: c_t = \overline{C}} \overline{C} P_t^c \; - \; c_{t_p} P_{t_p}^c. \tag{2.11}$$

We now define the sequence of vertices in the respective $\mathfrak{s} - \mathfrak{t}$ path. Let n_t^c, n_t^d, n_t^n denote the number of full charges, full discharges, and no charges/discharges from time zero till time t. The corresponding $\mathfrak{s} - \mathfrak{t}$ path transverses vertices $(n_t^c, n_t^d, n_t^n, \mathbf{0})$ for all $t < t_p$ and vertices $(n_t^c, n_t^d, n_t^n, e_1)$ for all $t \geq t_p$. Because A is feasible, we have that $s_t = \overline{C} n_t^c + \overline{D} n_t^d \in (\underline{S}, \overline{S})$ for $t < t_p$ and $s_t = \overline{C} n_t^c + \overline{D} n_t^d + p_1 \in (\underline{S}, \overline{S})$ for $t >= t_p$, so none of these vertices were removed in the graph as a result of pruning rules (a) - (c). Similarly, we have that $(n_t^c, n_t^d, n_t^n, \mathbf{0}) - (n_{t-1}^c, n_{t-1}^d, n_{t-1}^n, \mathbf{0})$ for $t < t_p$, $(n_{t_p}^c, n_{t_p}^d, n_{t_p}^n, e_1) - (n_{t_p-1}^c, n_{t_p-1}^d, n_{t_p-1}^n, \mathbf{0})$, and $(n_t^c, n_t^d, n_t^n, e_1) - (n_{t-1}^c, n_{t-1}^d, n_{t-1}^n, e_1)$ for $t > t_p$ are all unit vectors, corresponding to edges in the graph. The costs of the edges are as follows:

$$p_t^c * \overline{C} \text{ if } (n_{t-1}^c, n_{t-1}^d, n_{t-1}^n, w_{t-1}) - (n_t^c, n_t^d, n_t^n, w_t) = e_1 \rightarrow c_t = \overline{C} \tag{2.12}$$

$$-p_t^d * \overline{D} \text{ if } (n_{t-1}^c, n_{t-1}^d, n_{t-1}^n, w_{t-1}) - (n_t^c, n_t^d, n_t^n, w_t) = e_2 \rightarrow d_t = \overline{D} \tag{2.13}$$

$$0 \text{ if } (n_{t-1}^c, n_{t-1}^d, n_{t-1}^n, w_{t-1}) - (n_t^c, n_t^d, n_t^n, w_t) = e_3 \rightarrow c_t = d_t = 0 \tag{2.14}$$

$$c_{t_p} P_{t_p}^c \text{ if } t = t_p \tag{2.15}$$

Summing the costs of the edges used in the $\mathfrak{s} - \mathfrak{t}$ path gives the negative of (2.11). Now we prove that Given an $\mathfrak{s} - \mathfrak{t}$ path there is a MOB vertex with opposite cost. Let $\{(n_t^c, n_t^d, n_t^n, w_t)\}_{t \in T}$ be nodes that define an $\mathfrak{s} - \mathfrak{t}$ path where $w \in \mathbb{R}^{|\mathcal{P}|}$. We can

construct a MOB vertex solution to (2.1) as follows.

$$c_t = \overline{C} \text{ if } (n^c_{t-1}, n^d_{t-1}, n^n_{t-1}, w_{t-1}) - (n^c_t, n^d_t, n^n_t, w_t) = e_1 \tag{2.16}$$

$$d_t = \overline{D} \text{ if } (n^c_{t-1}, n^d_{t-1}, n^n_{t-1}, w_{t-1}) - (n^c_t, n^d_t, n^n_t, w_t) = e_2 \tag{2.17}$$

$$c_t = 0 \text{ and } d_t = 0 \text{ if } (n^c_{t-1}, n^d_{t-1}, n^n_{t-1}, w_{t-1}) - (n^c_t, n^d_t, n^n_t, w_t) = e_3 \tag{2.18}$$

$$c_t = p_j \text{ or } d_t = p_j \text{ if } (n^c_{t-1}, n^d_{t-1}, n^n_{t-1}, w_{t-1}) - (n^c_t, n^d_t, n^n_t, w_t) = e_{j+3}, \tag{2.19}$$

where the last value assignment is decided if p_j is a partial charge or discharge. We note that the s, u, and v variables can be uniquely determined by the charge and discharge values and that $s_0 = S_{\text{init}}$, $s_T = S_{\text{final}}$ and $s_t \in (\underline{S}, \overline{S})$ for all $0 < t < T$ as a result of the pruning rules for the nodes in the graph. As in the previous case, this vertex has the opposite cost of the $\mathfrak{s} - \mathfrak{t}$ path. □

Corollary 2.3.1. *The optimal MOB vertex can be found in polynomial time.*

Proof. Proof: There are no negative cost cycles in the graph. Considering the vertices of type (2.9) and (2.10), since $0 \leq n^c, n^d, n^n, |\mathcal{P}| \leq T$, then the number of nodes is of order $O(T^4)$. As a result, the number of edges is of order $O(T^8)$ in the worst case where we have a complete graph. Also, the cost of the edges are polynomial relative to the size of the inputs of the problem, since the computation requires summation and multiplication by constant numbers. □

Corollary 2.3.2. *There exists a polynomial-sized integer formulation for the minimum cost $\mathfrak{s} - \mathfrak{t}$ path for these graphs. This formulation corresponds to the convex hull of all MOB vertices in the given interval.*

2.5 Optimizing Over All Vertices

Here we show how to utilize the previous construction in order to optimize the base model over all vertices. The key idea is that any arbitrary vertex in the convex hull of

the model can be decomposed into MOB vertices viewed as optimal solutions to subproblems of the base model. Let $(s_t, c_t, d_t, u_t, v_t)_{t\in\mathcal{T}}$ be a generic vertex, and t_1 denote the first time period where s_{t_1} is either $\underline{S}$ or $\overline{S}$. We truncate all variables with indices greater than t_1 to obtain $(s_t, c_t, d_t, u_t, v_t)_{t\in\{1,\dots,t_1\}}$ which can be viewed as a MOB vertex and a solution to an instance of the base model defined over the time interval $[0, t_1]$ with $S_{\text{init}} = s_0$ and $S_{\text{final}} = s_{t_1}$. Similarly, if t_2 denotes the second time period where s_{t_2} is either $\underline{S}$ or $\overline{S}$, then projecting the vertex onto $(s_t, c_t, d_t, u_t, v_t)_{t\in\{t_1,\dots,t_2\}}$, gives us a MOB vertex as a solution to the subproblem defined over $[t_1, t_2]$ with $S_{\text{init}} = s_{t_1}$ and $S_{\text{final}} = s_{t_2}$. This process continues until we reach the last time period t_n with $s_{t_n} \in \{\underline{S}, \overline{S}\}$ and the MOB vertex $(s_t, c_t, d_t, u_t, v_t)_{t\in\{t_n,\dots,T\}}$. Taking advantage of this independent structure of the base model, we construct a graph for which solving the shortest path problem is equivalent to finding an optimal solution to the base model.

2.5.1 Shortest path problem for all vertices

We define a directed graph where nodes are the time periods when the state of the charge of the battery meets it bounds. We denote the nodes by (t, b) where $t \in \mathcal{T}$ and $b \in \{\overline{S}, \underline{S}\}$. There is an arc from node (t, b) to node (t', b') if $t' > t$ and the MOB subproblem on interval $[t, t']$ with the initial state of charge b and final state of charge b' is feasible. The cost of this arc is the optimal objective value of the aforementioned subproblem. We add dummy nodes $\mathfrak{s} = (0, S_{\text{init}})$ and $\mathfrak{t}$ with arcs from $\mathfrak{s}$ to all feasible nodes (t, b) for $t \in \mathcal{T}$ and an arc from node (T, S_{final}) to $\mathfrak{t}$ (with the cost of zero).

Example 2.4. Consider the same parameters in example (2.2). Figure 2.2 illustrates the above graph where enforcing $S_{\text{final}} = 10$, the length of $\mathfrak{s} - \mathfrak{t}$ shortest path is 38. Then our optimal schedule with the objective value of -38 is: full charge at time 1, partial charge at time 2 and do nothing at time 3, which corresponds to the path

$$\mathfrak{s} \longrightarrow (2, 10) \longrightarrow (3, 10) \longrightarrow \mathfrak{t}.$$

The infeasible nodes are represented in gray and all arcs and paths traversing them are removed form the graph. Next theorem proves the optimality of this solution.

Now we formulate the shortest path problem with the following set of variables corresponding to the nodes of the graph.

$$\left\{ x_{[t,t']^{b,b'}} \mid t, t' \in \mathcal{T} \cup \{0\},\ t < t',\ b, b' \in \{S_{\text{init}}, \underline{S}, \overline{S}, S_{\text{final}}\} \right\}$$

Note that not all nodes are feasible given the set of parameters. In this case we set the associated variables to zero.

$$\min \quad \sum_{b,b' \in \{\underline{S},\ \overline{S}\}} \sum_{t=0}^{T} \sum_{t'=t+1}^{T} c_{[t,t']^{b,b'}} x_{[t,t']^{b,b'}} \tag{2.20a}$$

$$\text{s.t.} \quad \sum_{t=1}^{T} x_{[0,t]^{S_0,b}} = 1 \qquad b \in \{\underline{S},\ \overline{S}\} \tag{2.20b}$$

$$x_{[0,t]^{S_0,b}} + \sum_{b' \in \{\underline{S},\overline{S}\}} \sum_{t'=1}^{t} x_{[t',t]^{b',b}} = \sum_{b'' \in \{\underline{S},\overline{S}\}} \sum_{t''=t+1}^{T} x_{[t,t'']^{b,b''}} \qquad t \in \mathcal{T}, b \in \{\underline{S}, \overline{S}\}. \tag{2.20c}$$

In this formulation, $\mathcal{T} = \{1, 2, \ldots, T\}$ and for simplicity, and without loss of generality we assumed $S_0, S_{\text{final}} \in \{\underline{S},\ \overline{S}\}$.

Theorem 2.5. *The $\mathfrak{s}-\mathfrak{t}$ shortest path on the above graph provides the optimal solution to the base model (2.1) with the cost of negative optimal objective value.*

Proof. Proof: We show for a given optimal solution of the base model with optimal objective value of z, there is an $\mathfrak{s} - \mathfrak{t}$ path in the graph with the cost of $-z$. This is trivial due to the construction of the graph and decomposition of the base model into independent subproblems using MOB vertices as described above. Now we show that the shortest $\mathfrak{s} - \mathfrak{t}$ path in the graph with the length of l provides an optimal solution to the base model with the optimal objective value of $-l$. Let $A = (s_t, c_t, d_t, u_t, v_t)_{t \in \mathcal{T}}$

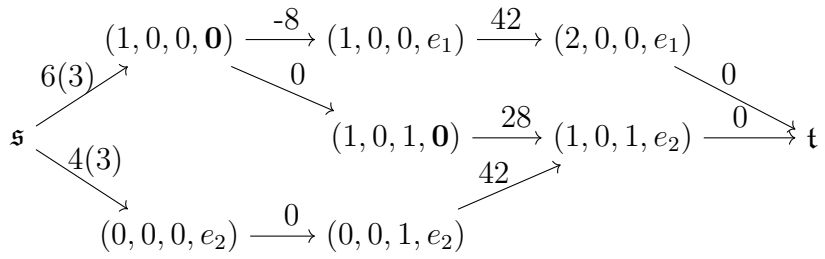

Figure 2.1: Directed graph for optimizing over MOB vertices

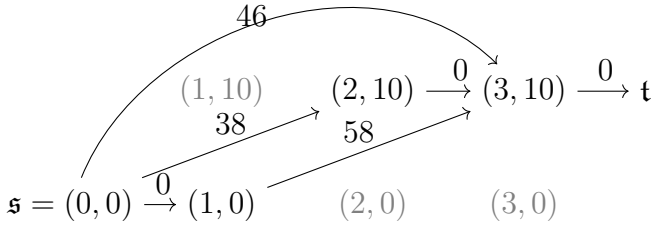

Figure 2.2: Directed graph for optimizing over all vertices

be feasible solution to the base model corresponding to the shortest path. If this solution is not optimal, then there is a time period $t \in \mathcal{T}$ such that changing c_t or d_t provides an improvement to the objective value. Since t belongs to one of the subintervals of $[0, T]$ then it is possible to improve the optimal objective value of this subinterval. This leads to improvement of the length of the $\mathfrak{s} - \mathfrak{t}$ shortest path which is a contradiction.

□

Corollary 2.5.1. *The size of the above graph is polynomial.*

Proof. Proof: The number of vertices is of order $O(T)$ and the number of edges is $O(T^2)$. By theorem 2.3.1 processing each vertex is of polynomial time. Hence, the computation of the edges are polynomial relative to the size of the inputs. □

Corollary 2.5.2. *The base model (2.1) can be optimized over all vertices in polynomial time.*

Proof. Proof: The statement is true by Corollaries 2.3.1 and 2.5.1. □

2.6 Exploiting realistic prices

Usually battery storage capacities are relatively small compared to conventional energy generators participating in the wholesale markets; thus, their impact on energy prices could be neglected and they are usually modelled as price takers (Arteaga and Zareipour, 2019). In this case it is common to consider equal and fixed prices for buying and selling at a given time period. However, this is the price of delivered energy. Charging and discharging inefficiencies affect how much energy is actually stored in the grid. Let P_m be the market price for buying and selling energy at a given time. Assume for the sake of argument that P_m is positive. From the perspective of electricity in the battery, and considering inefficiencies, discharging d units of energy *in the battery* delivers $< d$ units to the market, yielding a price of $< P_m$ per unit

of energy in battery. Similarly, buying b from the market yields less than b units of electricity in the battery, making the price of charging, in terms of energy in the battery, strictly greater than P_m. With this in mind that we investigate how the model simplifies with this realistic pricing scenarios. For this section, we assume for all time periods $t \in \mathcal{T}$ we have

$$\frac{P_t^c}{P_t^d} > 1. \tag{2.21}$$

This implies that at each time t the charge and discharge prices have the same sign. Also, when prices are positive we have $P_t^c > P_t^d$, while when the prices are negative we have $P_t^c < P_t^d$.

Definition 2.2. *We say a feasible solution of the linear relaxation of formulation (2.1) is orthogonal at time period $t \in \mathcal{T}$ if $c_t d_t = 0$.*

Remark. An integer solution can be recovered from an orthogonal solution to the formulation (2.1).

Theorem 2.6. *If the charge and discharge prices at time period $t \in \mathcal{T}$ are strictly positive, then the LP relaxation of the formulation (2.1) returns an orthogonal solution at time t.*

Proof. Proof: Towards a contradiction assume that there is a time period t with $u_t v_t \neq 0$ in the solution of the LP relaxation. This implies that $c_t > 0$ and $d_t > 0$. For $\epsilon > 0$ we have that by reducing both c_t and d_t by ϵ, the problem remains feasible (as all other time periods are unaffected), but doing so increases the profit. □ □

Theorem 2.7. *If an optimal MOB solution to the battery scheduling problem has two or more time periods where the battery is neither fully charging or fully discharging (do nothing or partial charge/discharge), then in all time periods with negative prices, the battery is charging or discharging at full capacity.*

Proof. Proof: Aiming at a contradiction, assume t is a time period with negative prices; $P_t^c, P_t^d < 0$; when the system is not charging at full capacity, i.e. $c_t < \overline{C}$. Let

t' be another time period without a full charge/discharge. Assume prices at time t' are negative, i.e. $P^c_{t'}, P^d_{t'} < 0$ (since otherwise increasing charge at time t, decreasing or remaining at zero charge at time t' is more profitable which contradictory to the optimality of the solution.) Now we consider two cases:

Case 1: in both two time periods t and t', charge and discharge variables are zero (do nothing in both times). Let $\epsilon > 0$ be given. In this case, increasing charge at time t and decreasing the charge at time t' by ϵ will result in a feasible solution while changing the profit by $-\epsilon P^c_t + \epsilon P^d_{t'}$. Similarly, decreasing the charge at time t by ϵ and increasing the charge at time t' will result in a feasible solution while changing the profit by $\epsilon P^d_t - \epsilon P^c_{t'}$. As a result of the optimality, we must have

$$-\epsilon P^c_t + \epsilon P^d_{t'} \leq 0 \tag{2.22}$$

$$\epsilon P^d_t - \epsilon P^c_{t'} \leq 0. \tag{2.23}$$

From assumption (2.21), for negative prices we have $P^c_t < P^d_t$. However, the above equations lead to

$$P^d_{t'} \leq P^c_t < P^d_t \leq P^c_{t'}, \tag{2.24}$$

which contradicts (2.21).

Case 2: in time period t charge and discharge variables are zero (do nothing), and in time t' we have partial charge. (The case of partial charge in both times is impossible as a result of being a MOB vertex.) The same argument as above is valid in this case with taking into consideration that here we increase the discharge variable by ϵ at time t to have the inequality (2.23). □

The impact of the above theorem is the following. Suppose we decomposed the MOB shortest path problem into two different subproblems based on the following disjunction. We consider one subproblem where at most one partial/do-nothing is allowed and another where there are at least two (at most one of which is a partial

charge). In the former subproblem, there is only one possible value for the partial charge. In addition, we can remove all arcs associated with do-nothing. In the latter we can remove all arcs associated with both partial and do-nothing from from nodes representing time periods with negative prices. Note that if all prices are negative, than the solution obtained by solving the later subproblem can not be optimal.

2.7 Computational Results

In this section we represent the performance of the integer program (IP) given in (2.1) versus our shortest path (SP) approach. As mentioned at the beginning, in most cases the battery charging problem is very easy, even in the case of negative prices as long as there is a high variance in the prices (as one will always want to sell at the highest-priced times and sell at the lowest). The first two columns of Table 2.2 show results from negative priced instances where prices were chosen over a uniform distribution. None of these problems are difficult for Gurobi using the original formulation, and while both methods are fast, the integer programming approach is much faster than the polynomial algorithm. The hardest battery charging problems are those where the prices are negative and constant. This is shown in the last two columns, where Gurobi struggles to find optimal solutions to the longer-period problems, whereas the shortest-path approach optimizes them quickly.

2.8 Discussion and Conclusion

Motivated by increasing integration of energy storage into power systems and increasing occurrence of negative energy prices we investigate how to solve the lossless battery scheduling problem in polynomial time. We recast the MILP formulation (2.1) of the problem as a shortest path problem over directed acyclic graphs. Our approach provides convex hull description of the problem.

Table 2.2: Performance of IP and SP for negative prices and parameters: $\underline{S} = 0, \overline{S} = 10, \overline{C} = 4, \overline{D} = 3, S_{\text{init}} = 2$.

	Random Prices		Constant Prices	
T	IP (s)	SP (s)	IP (s)	SP (s)
6	0.007	0.009	0.021	0.020
9	0.008	0.0356	0.011	0.035
12	0.0118	0.085	0.044	0.076
15	0.016	0.1296	0.094	0.141
18	0.018	0.285	0.103	0.268
21	0.0156	0.371	0.482	0.363
25	0.0132	0.612	1.16	0.578
28	0.0224	0.876	4.351	0.81
31	0.027	1.124	78.22	1.089
34	0.0314	1.490	89.809	1.511
37	0.039	1.978	> 180	1.941

We view differently the existing possible actions of a battery storage, i.e. charging, discharging, and doing nothing at a given time period. In our approach, possible actions of a battery are: full charge/discharge, partial charge/discharge, and do nothing. This shift of viewpoint coupled with observations on the state of charge patterns allows us to recognize a special type of vertices; MOB vertices; in the MILP polytope. We find an optimal schedule over these special vertices through the use of the shortest path problem. Then we demonstrate how to decompose a generic vertex into MOB vertices and solve another shortest path in order to optimize over all vertices.

In regard of the MILP formulation (2.1) and its recast as a shortest path note some extensions that can still be solved in polynomial time:

- The minimum charge and discharge capacities in formulation (2.1) is assumed to be zero. In the case where these parameters are non zero, our method remains polynomial. A slightly modified Theorem 2.1 still hold for the nonzero parameters with the exception that there is only one time period where the charging and discharging rates are not at their bounds. In this case, the vertices for the MOB shortest path problem would include a reference to the number of minimum charges and discharges in addition to the number of full charges, full discharges, and partial charges.

- Minimum Up/Down time on the generators can be enforced in the shortest path polytope. This can be done by adding an additional index to each vertex in the MOB subproblem to reflect how many consecutive charges/discharges there have been.

However, some extensions may not necessarily be solvable in polynomial time.

- We assumed that there is no loss in the battery over time. The results still hold if the battery experiences a constant loss with respect to time. However, if the loss is proportional to the state of charge, the proposed algorithm would not

work as the ordering of charge and discharge decisions the equation (2.6) are now important, making the resulting shortest path exponentially large.

- Similarly, we assume linear charge and discharge prices. Moving to piecewise linear makes the problem NP-Hard. This can be seen by (Pan et al., 2022), as they show that the fuel constrained unit commitment self scheduling problem with a piecewise linear objective function is NP-Hard. In this context, the fuel constrained generator can be thought of as a battery with zero charge capacity. The main reason for the change in complexity is that Theorem 2.1 is not consistent with piecewise linear objective functions.

Chapter 3

Optimization of a battery under price uncertainty

3.1 Introduction

The linear program (2.1) presented in the previous chapter, can be employed to determine the optimal solution and arbitrage for a deterministic battery problem. With perfect information on electricity prices, this solution represents the upper bound for potential profit. However, electricity prices often fluctuate significantly and are challenging to predict with high accuracy. This uncertainty drives us to adopt probabilistic models, specifically stochastic programming and dynamic programming, and a model-free approximation to better address these fluctuations and to estimate potential arbitrage without overestimation. The goal of this chapter is to maximize expected energy arbitrage of a battery only in the presence of price uncertainty utilizing various optimization techniques and compare their performance in Markov and near Markov setting.

3.1.1 Literature review

Research directions in energy arbitrage: Numerous studies have examined energy storage optimization for arbitrage. Among the primary lines of research in the literature are the followings.

- *Optimizing energy storage solely for arbitrage under price uncertainty, without accounting for other services that an storage can provide:* In (Sioshansi et al., 2009) authors analyze and evaluate the arbitrage value of a storage device using perfect and imperfect foresight of electricity prices. They use a back-casting heuristic (which is essentially applying mean value problem policy to future) that assumes historical price and repeated load patterns. Reference (Vejdan and Grijalva, 2018) evaluates the arbitrage value of in real time and day ahead markets using back-casting and normal errors.

- *Interaction between storage and renewables:* Early study (Greenblatt et al., 2007) determines that when greenhouse gas emissions taxes are high, compressed air energy storage is a more favorable supplemental resource for wind generation compared to natural gas turbines. Reference (Kim and Powell, 2011) addresses a wind energy commitment problem in the context of storage and analytically derives the optimal policy for an infinite time horizon. Authors in (Dicorato et al., 2012) study the planning and operation of a wind energy storage system in an electricity market using forecasts of the prices. The paper by (Zhou et al., 2019) achieves a near-optimal policy for managing wind generation with energy storage, considering the constraints of finite transmission capacity and negative electricity prices.

- *Co-optimizing battery storage for energy arbitrage and capacity and ancillary services as an alternative or complement to energy arbitrage:* (Walawalkar et al., 2007) observes that for sodium sulfur batteries and flywheel energy storage system frequency regulation offers higher profits than energy arbitrage. (Xi

et al., 2014; Feng et al., 2022; Cheng and Powell, 2016) also have shown ancillary services can be a substantial revenue source for energy storage.

- *Distributed versus grid level storage devices:* (Xi et al., 2014) co-optimize multiple storage applications while accounting for market and system uncertainty and they study the value of a battery that is installed in a residential home as a stationary storage device. The study (Wankmüller et al., 2017) investigates the representation of battery degradation in grid level energy storage applications. (Penaranda et al., 2021) is a case study for grid-scale battery energy storage in the Colombian electricity market for arbitrage purposes.

- *Pump hydro storage:* Of all types of energy storage systems, the pumped hydro storage (PHS) has the most mature technology and the largest capacity at acceptable cost (Ding et al., 2012). There is an abundance research on optimizing PHS in literature. (Toufani et al., 2023) provides an overview of the research dealing with optimization of PHS systems under uncertainty. The study (Ding et al., 2012) proposes a coordination operation strategy for wind farms and PHS plants utilizing day ahead wind power output forecasts which leads to increased energy arbitrage. The study (Chazarra et al., 2017) analyzes the value of perfect information in forecasting of spot prices of a PHS in the joint energy and reserve markets.

- *Impact of degradation of a battery on arbitrage:* It has been shown that neglecting battery degradation reduces long term profitability of the battery. Authors in (García-Miguel et al., 2022) review different methods of degradation control for short time operation and classify different practices found in literature. (Wankmüller et al., 2017) in a case study using historical data from MISO electricity market has shown a reduction of revenue due to degradation in the range $12 - 46\%$. Reference (Perez et al., 2016) presents a combined economic-degradation model that quantifies effects of various operational

policies on energy arbitrage. The paper (Cao et al., 2020) uses a model-free deep reinforcement learning (DRL) method to optimize the battery energy arbitrage considering an accurate battery degradation model. The authors in this study adapt a semi-empirical lithium-ion battery degradation model which can account for irregular cycling operations to estimate the battery degradation costs.

Methodologies: Various optimization methods have been employed in literature to evaluate the energy arbitrage of an storage. The performance of these methods rely heavily on forecasting information such as renewable energy power generation, load and price signals. One of the main challenges is the difficulties for forecasting the prices in future. The paper (Nowotarski and Weron, 2018) reviews recent advances in probabilistic price forecasting. Some of main approaches for arbitrage optimization and electricity price forecasting are:

- *(approximate) stochastic dynamic programming (SDP):* Authors in (Xi et al., 2014) propose solving an approximate SDP in two phases. In the first phase, the optimal policies and objective function values for each state are found by solving a discretized SDP. In the second phase a mixed-integer program, in which the value function of the true SDP is approximated as piecewise-linear using the optimal objective function values from the descretized SDP, is solved to obtain a near-optimal policy. Study (Jiang and Powell, 2015) take advantage of monotonicity properties of the their problem and propose a monotone approximate dynamic programming that can be used to obtain a near-optimal bidding policy in real time market. In (Cheng and Powell, 2016) authors using approximate SDP, in order to store the value functions for all the states, they implemented a low-rank approximation using singular value decomposition.

- *deep and reinforcement learning(DL, RL):* Authors of (Wang and Zhang, 2018) obtain a Q-learning policy for temporal arbitrage in the real market with

a reward function that in addition to instant profit, incorporates historical information. In order to address the limitations of Q-learning such as inscalability, (Bui et al., 2019) proposes a distributed operation strategy using double deep Q-Network method which uses deep neural network as a function approximator to estimate Q-values. In (Cao et al., 2020) authors address the uncertainty of electricity price, by considering a hybrid convolutional neural network (CNN) and Long Short Term Memory (LSTM) model to predict electricity prices for the next day. (Shuai et al., 2021) investigates a branching dueling Q-Network (BDQ) based online operation strategy for a microgrid with distributed energy storage systems which improves scalability and applicability of deep reinforcement learning methods. (Li et al., 2022) proposes a dense skip attention based DL model for day-ahead price forecasting. In their model, a mechanism of dense skip attention is proposed to efficiently assign learnable weights on the features for training.

- *time series analysis:* Analysing the underlying patterns and structure of electricity prices, and making forecasts using different time series models are studied in (Muniain and Ziel, 2020). Reference (Klein et al., 2023) proposes two approaches for constructing deep time series probabilistic models based on a variant of RNN called an echo state network (ESN).

- *other machine learning methods:* Price spike forecasting is studied in (Stathakis et al., 2021) using support vector machine. In (Tschora et al., 2022), Authors study electricity price forecasting on the day-ahead market using machine learning techniques namely support vector regressor and random forest regressor.

- *Geometric Brownian motion (GBM):* In order to derive a bidding strategy (Mohamed et al., 2024) proposes methods that include price forecast and scenario generation using GBM. Their two methods involves forecasting one-point price profiles using SARIMA and depends on the price residuals between

each two successive days. In (Shin and Lee, 2024) the battery revenue with uncertainty is modeled as a stochastic process using GBM, and the optimal time to invest in an energy storage is determined using an Least Squares Monte Carlo simulation considering investment costs.

3.1.2 Preliminaries and notations

Following the classical textbook (Birge and Louveaux, 2011) let us review the basic concepts and standard notations that will be used in this chapter.

Two stage stochastic program: The following mathematical model provides a general formulation of a two stage stochastic program where a decision maker chooses a decision variable x to optimize expected outcomes:

$$\begin{aligned} RP = \max_x \ \mathbb{E}_{\boldsymbol{\xi}} z(x, \boldsymbol{\xi}) = \max_x \ & \langle c, x \rangle + \mathbb{E}_{\boldsymbol{\xi}}[Q(x, \boldsymbol{\xi})] \\ & \text{subject to} \ \ Ax \leq b \\ & x \geq 0 \end{aligned} \tag{3.1}$$

where the recourse function $Q(x, \xi)$ is the optimal objective value of

$$\begin{aligned} & \max_y \ \langle q(\xi), y(\xi) \rangle \\ & \text{subject to} \ \ T(\xi)x + W(\xi)y(\xi) \leq h(\xi) \\ & y(\xi) \geq 0. \end{aligned} \tag{3.2}$$

In above program known as recourse problem (RP), x is first stage decision variable, y is the second stage variable, and $\mathbb{E}_{\boldsymbol{\xi}}$ denotes the expectation with respect to a random variable $\boldsymbol{\xi}$. Suppose the uncertainty of problem (3.1) can be modeled using a set of scenarios. We let a realization of random variable $\boldsymbol{\xi}$ denoted by ξ corresponds to one

of these scenarios. The recourse function $Q(x, \xi)$ quantifies the benefits associated with taking corrective actions in response to the realized value ξ.

The solution of stage one variable x^* obtained by solving recourse problem, is called 'here and now' solution, and are then followed by the resolution of uncertainty. Second stage decisions or recourse decisions, are taken 'wait and see' as corrective action at the end of the stage two.

Mean value problem: Stochastic program (3.1) is typically difficult to solve in real-world applications. A common approach is to simplify this problem by using a deterministic program, known as expected value problem, mean value problem (MVP) where the random variable $\boldsymbol{\xi}$ is replaced with its expected value. The optimal objective of MVP is denoted by EV:

$$EV = \max_x z(x, \overline{\xi}) \tag{3.3}$$

where $\overline{\xi} = \mathbb{E}(\boldsymbol{\xi})$. We denote the optimal solution of MVP by $\overline{x}(\overline{\xi})$, and define the expected result of using the EV solution to be

$$EEV = \mathbb{E}_{\boldsymbol{\xi}}(z(\overline{x}(\overline{\xi})), \boldsymbol{\xi}). \tag{3.4}$$

Now the value of the stochastic solution is defined as

$$VSS = RP - EEV, \tag{3.5}$$

which measures the cost of ignoring uncertainty in choosing a decision.

Wait and see problem: Another problem which is associated with (3.1) is the wait and see problem where the random vector $\boldsymbol{\xi}$ is replaced with a realization ξ:

$$\max_x \; z(x, \xi) = \max_x \; \langle c, x \rangle + Q(x, \xi). \tag{3.6}$$

Suppose that we can solve problem (3.6) for each scenario and obtain a solution denoted by $\overline{x}(\xi)$ and its optimal objective $z(\overline{x}(\xi))$. The expected value of these optimal objectives known as wait-and-see solution is

$$WS = \mathbb{E}_{\boldsymbol{\xi}}\left[\max_{x} z(x, \boldsymbol{\xi})\right] = \mathbb{E}_{\boldsymbol{\xi}} z(\overline{x}(\boldsymbol{\xi}), \boldsymbol{\xi}) \tag{3.7}$$

The purpose of the wait and see problem is to determine the best possible outcome that could be achieved if all future uncertainties were known in advance. It serves as a benchmark against other solutions, particularly those obtained from the recourse problem and mean value problem.

Multi-stage stochastic program: To enhance our assessment of uncertainty, we can move beyond a two-stage stochastic model and adopt a multi-stage approach, leading to better approximations. Multi-stage similar to two stage is scenario based approach meaning that is often uses a scenario tree to represent different possible realizations of uncertainties over time. In a planning horizon of H stages, the realization of uncertainty is typically modeled as a stochastic process $(\xi^1, \xi^2, \ldots, \xi^H)$ such that, ξ_1 is deterministic, and for each $t = 2, 3, \ldots, T$, ξ^H is a random vector that will be realized in stage t. The nested form of a multi-stage stochastic program can be expressed as the following.

$$\begin{aligned}
\max \quad & \mathbf{c}^1\mathbf{x}^1 + \mathbb{E}_{\boldsymbol{\xi}^2}[\max \mathbf{c}^2(\xi^2)\mathbf{x}^2(\xi^2) + \ \ldots \ + \ \mathbb{E}_{\boldsymbol{\xi}^H}[\max \mathbf{c}^H(\xi^H)\mathbf{x}^H(\xi^H)]] \\
\text{subject to } & W^1\mathbf{x}^1 \leq \mathbf{h}^1 \\
& T^1(\xi^2)\mathbf{x}^1 + W^2\mathbf{x}^2(\xi^2) = \mathbf{h}^2(\xi^2) \\
& \vdots \\
& T^{H-1}(\xi^H)\mathbf{x}^{H-1}(\xi^{H-1}) + W^H\mathbf{x}^H(\xi^H) = \mathbf{h}^H(\xi^H) \\
& \mathbf{x}^1 \geq 0 \quad \mathbf{x}^t(\xi^t) \geq 0 \ t = 2, \ldots, H
\end{aligned} \tag{3.8}$$

In the above formulation: $\mathbf{x}^t$ represents the decision variables for period t, $\mathbf{c}^t$ is the cost vector incurred for stage t, $\mathbf{h}^t$ denotes the right-hand side vector for period t, ξ^t is the random variable for stage t, $\mathbf{x}^t(\xi^t)$ are the decision variables made at period t for the realization of ξ^t, $T^{t-1}(\xi^t)$ is the technology matrix for time $t-1$, and W^t is the fixed recourse matrix for period t.

stochastic dynamic programming: When uncertainty unfolds over multiple stages, an alternative to consider is stochastic dynamic programming (SDP) instead of multi-stage stochastic programming. SDP relies on the concept of states and transitions between states present in Markov decision processes. At the core of SDP is the Bellman equation, which recursively and backward defines the value function. The value function represents the expected optimal cost/reward from a given state onwards. To solve for optimal value function using SDP requires to visit every state which is computationally difficult.

Q-learning: Q-learning is a model-free (distribution-free) reinforcement learning algorithm used to find the optimal action-selection policy for any given finite Markov decision process (MDP). It aims to learn the value of taking a given action in a given state, which is referred to as the "Q-value" or "action-value". Q-learning approximates the optimal value function of SDP and it is used when the state space is large and/or the dynamics of the environment are unknown.

We organize this chapter as the following. In section 3.2 we list the models of prices that we study throughout the chapter and setup computational parameters. Section 3.3 is devoted to average case analysis, the so called mean value problem. In section 3.4 we present two stage stochastic program for a battery under price uncertainty with two different types of scenario generation algorithm. Section 3.5 is devoted to stochastic dynamic programming and the related computational results. In section 3.6 we study a model free approximation algorithm, the so called Q-learning. Section 3.7 concludes the chapter.

3.2 Models of prices

In this study we consider two different types of models for charge and discharge prices. First we simulate the prices by assuming that they follow a Markov process. A Markov process a sequence of stochastic events where the next state of a variable or system is independent of all the past states, except for the current state.

Definition 3.1. *(Markov process) A discrete-time Markov process is a sequence of random variables $X_1, X_2, X_3, \ldots$ with the Markov property:*

$$\mathbb{P}\left(X_{t+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_t = x_t\right) = \mathbb{P}\left(X_{t+1} = x \mid X_t = x_t\right) \tag{3.9}$$

There are two types of Markov processes, time-homogeneous and time-inhomogeneous.

Definition 3.2. *A Markov process is said to be time-homogeneous if the transition probabilities are the same at each time t, i.e. the probability of transition is independent of t.*

Definition 3.3. *A time-inhomogeneous Markov process refers to a process with different transition probabilities at each time t.*

In our problem the random variable X_t represents the real time price of electricity at time t, and $\{X_1, X_2, X_3, \ldots, X_T\}$ is the sequence of all real time prices for a given time horizon of length T. We consider both types of Markov processes in our study. For the time-homogeneous case we assume that the probability of going up/down is fixed throughout the time horizon, and for the time-inhomogeneous case for simplicity and without loss of generality, we consider two different probabilities for the first and the second half of the time horizon. Time-homogeneous Markov processes are simple and widely used in many contexts. However, in practice, the time independence assumption is violated for many data sets (Truquet, 2019) including energy prices. Comparing the two types will offer valuable insights into managing arbitrage optimization in the presence of high volatility and understanding the computational costs involved.

For simplicity we assume our Markov process is a simple random walk. In particular, for the entire time horizon T, we assume the prices at future time $t+1$ decrease or increase by 1 unit with specified probabilities with respect to the prices at time t. This is not a restrictive assumption. For a more complex random walk the amount of increase/decrease may vary at each time period in order to introduce more volatility into the model. For example modeling a spike of the prices at a given time is possible through a large price change and high probability of price growth. Another assumption is that the price of discharge is lower than the price of charge. We assume the gap between them at each time period to be one.

An additional consideration which is natural to take into account is whether the prices are bounded or unbounded. We will see in 3.5 for unbounded prices the value function of SDP is linear in terms of the prices which leads to an optimal policy that is prices independent. This is not the case when the prices are bounded. In one case, we suppose the prices are unbounded both from above and below and the in the second case we bound the prices from above in such a way that when the prices hit a prescribed threshold, we reflect them down by one unit. For instance, whenever the price of charge reaches at the threshold equal to 42, we set the prices of charge and discharge to 41 and 40 respectively.

To summarize in our simulation we have four models of prices:

1. Unbounded and time-homogeneous
2. Bounded and time-homogeneous
3. Unbounded and time-inhomogeneous
4. Bounded and time-inhomogeneous

We evaluate various optimization models, including the mean value problem, two-stage stochastic programming, stochastic dynamic programming, and Q-learning, against the above price models. We examine the solution properties of each optimization model and compare their quantitative performance in terms of the

resulting optimal operational policies and the (expected) arbitrage. This analysis provides insights into the practical advantages of more complex models compared to the simpler deterministic mean value problem.

The second approach to test the performance of the optimization models involves using historical data. By analyzing past price movements, historical data can reveal patterns, trends, and behaviors inherent in the market. This information helps in understanding the factors that influence price changes and provides a basis for forecasting future prices. Historical data can uncover cyclical patterns, seasonal variations, and reactions to specific events, offering valuable insights into the nature and dynamics of prices over time. In this study we use prices of June, July and August 2023 obtained from PJM independent operating system, zone APS available at website https://www.engieresources.com/historical-data#reports_anchor. This data is gathered from public and/or internal sources of the website and may not be accurate.

3.2.1 Experimental setup

Here we list all parameters and computational setup for our simulations in subsequent sections. We consider the time horizon $T = 24$ that corresponds to daily hours. The physical parameters of the battery which we use throughout this chapter are:

$$\overline{S} = 10,\ \overline{C} = 4,\ \overline{D} = 3,\ S_{\text{init}} = 4. \tag{3.10}$$

The sample size K is set to 100000. The initial prices of charge and discharge are set to 38 and 37 respectively. We recall that the gap between prices of charge and discharge is assumed to be one at each time period. To precisely define the four cases of price models explained above, we setup the transition probabilities as well as the bound on the prices. For the time-homogeneous Markov prices we assume a single transition probability that a price at any time increases by one unit with probability 0.7. For time-inhomogeneous case for the first half of the time horizon we

consider the probability of 0.4 for increasing by one unit. For the second half of the time horizon this probability is set to 0.8. The probability of decreasing prices is one minus the probability of increasing for both cases. The upper bound for the charge and discharge prices are set to 42 and 41 respectively.

3.3 Mean value problem

The two stage stochastic program (3.1) is nonlinear. A linear deterministic equivalent or extensive form of the problem for each realization ξ of the random vector can be formulated as the following:

$$\begin{aligned} \max\ & \langle c, x\rangle + \sum_{\boldsymbol{\xi}=\xi} \mathbb{P}(\xi)\langle q(\xi), y(\xi)\rangle \\ \text{subject to}\ & Ax \leq b \\ & T(\xi)x + W(\xi)y(\xi) \leq h(\xi) \\ & x \geq 0 \\ & y(\xi) \geq 0. \end{aligned} \tag{3.11}$$

The two stage problem (3.1) can be solved by solving the deterministic equivalent problem (3.11) directly using commercial solvers. Solving this deterministic program is computationally expensive when the number of scenarios and in our case when the time horizon is large. A simple approach is to consider the mean value problem (MVP), where a decision maker replaces the random vector $\boldsymbol{\xi}$ by its expected value $\overline{\xi}$ and solves the deterministic program (3.11).

In our problem the uncertainty vector ξ is daily price trajectory. The MVP approaches this uncertainty by sampling prices drawn from a price model or historical data and then computing the expectation of the prices for each time period. Figure 3.7 shows price sampling and averaging over the entire time horizon for MVP. A gray line represents a price trajectory i.e. a realization of the random price ξ and the bold

line is the average of all gray lines, i.e. $\mathbb{E}[\xi] = \overline{\xi}$. We restate the deterministic battery charging problem from the previous chapter.

$$\max \quad \sum_{t=1}^{T} (P_t^d d_t - P_t^c c_t) \tag{3.12a}$$

$$\text{s.t.} \quad s_t = \eta s_{t-1} + c_t - d_t \qquad \forall t \in \mathcal{T}, \tag{3.12b}$$

$$s_0 = S_{\text{init}}, \tag{3.12c}$$

$$\underline{S} \leq s_t \leq \overline{S} \qquad \forall t \in \mathcal{T}, \tag{3.12d}$$

$$0 \leq c_t \leq \overline{C} u_t \qquad \forall t \in \mathcal{T}, \tag{3.12e}$$

$$0 \leq d_t \leq \overline{D} v_t \qquad \forall t \in \mathcal{T}, \tag{3.12f}$$

$$u_t + v_t = 1 \qquad \forall t \in \mathcal{T}, \tag{3.12g}$$

$$u_t, v_t \in \{0, 1\} \qquad \forall t \in \mathcal{T}. \tag{3.12h}$$

We solve problem (3.12) for all four cases of price models described in previous section and later compare the optimal profit and solution with two stage stochastic program and and SDP. The figures 3.1, ..., 3.4 display the computational results of our simulation for the four price models. In each scenario, we sampled daily price trajectories based on the probabilistic model, with a sample size of 10,000. We then solved problem (3.12) using the average prices for charging and discharging derived from all sampled trajectories.

The MVP effectively captures the basic trends of the prices in all cases. In Case 1, the optimal strategy is to fully charge the battery at the beginning of the day, maintain full capacity throughout the day, and discharge completely during the final time periods. Case 2 follows a similar policy; however, due to the boundedness introduced in the second half, the optimal policy oscillates between discharging and doing nothing. Here, we observe that the MVP can capture the slight volatility introduced to prices. In case 3 and 4 with time-inhomogeneous prices, the prices decreases with probability of 0.4 until time $t = 11$ and then they increase with

probability of 0.8 till the end of the time horizon. The optimal policy is to fully discharge at the beginning and stay at zero level till time $t = 8$ and then start charging to full capacity when the prices reach to their local minimum. At the end of the day when the prices are high the battery discharges fully to make profit.

We are now utilizing historical data in our computational studies. We use hourly real time electricity prices for each day of the months of June, July and August 2023 collected from PJM independent system operator (ISO) zone APS. We consider this data as a price of charge. The price of discharge is obtained by subtracting one the price of charge at each time during a day. Figure 3.5 shows the hourly average of prices over 60 days of the months June and July. The optimal operation policy of the battery obtained by solving the MVP represented in the same figure indicates discharge at the first hour of a day, then charge to full capacity, maintain it until we reach a peak at 5 pm. At this time the battery will be discharged for profit.

We tested the optimal policy derived from the MVP using average prices from June and July against the prices in August. The resulting expected arbitrage is 385.11, with a standard deviation of 181.65. The high standard deviation indicates significant volatility in electricity prices. While the MVP policy successfully captures the basic daily price trends, it fails to account for the high volatility, especially the sharp spikes and brief fluctuations in prices. Figure presents a histogram of the optimal objectives achieved by applying the MVP optimal policy to all days in August. The histogram reveals significant fluctuations in the daily optimal objectives.

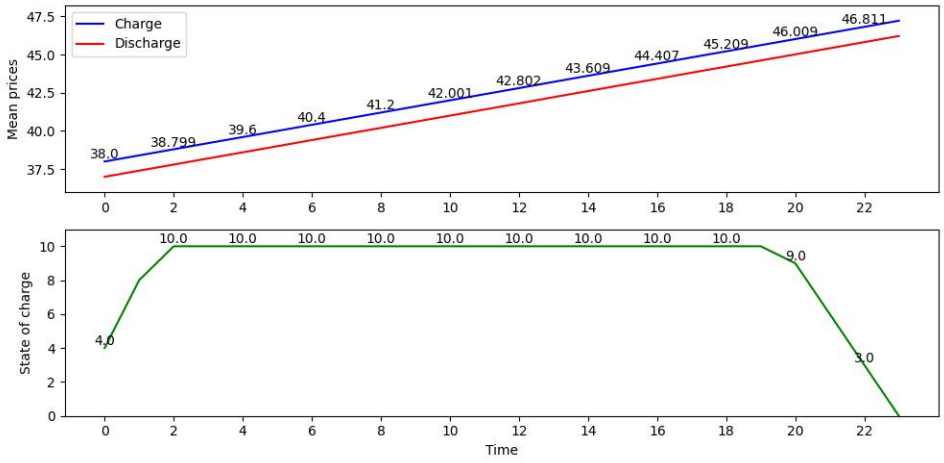

Figure 3.1: MVP solution for an unbounded time-homogeneous price trajectory. Total profit = 226.011.

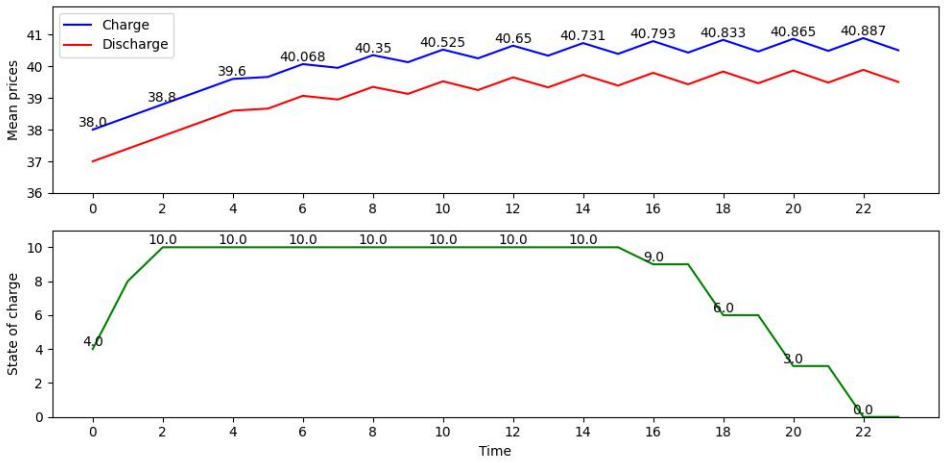

Figure 3.2: MVP solution for bounded time-homogeneous prices.Total profit = 167.302.

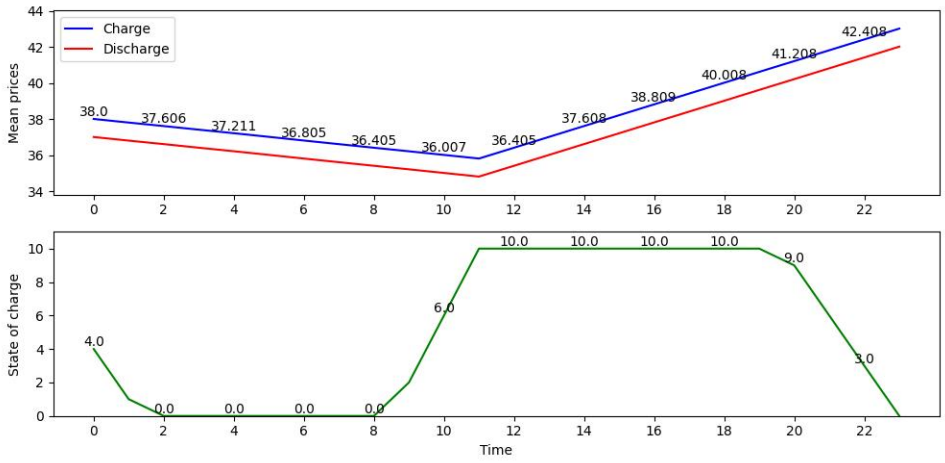

Figure 3.3: MVP solution for unbounded time-inhomogeneous prices. Total profit = 206.765.

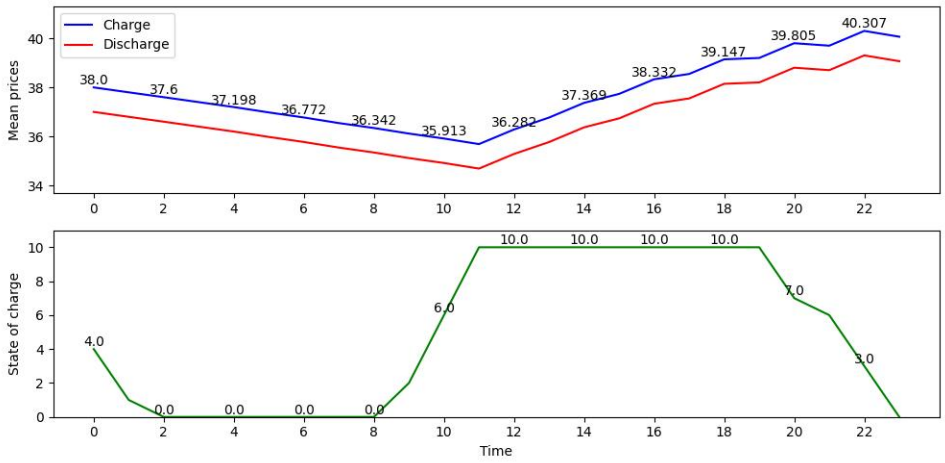

Figure 3.4: MVP solution for bounded time-inhomogeneous prices. Total profit = 170.745.

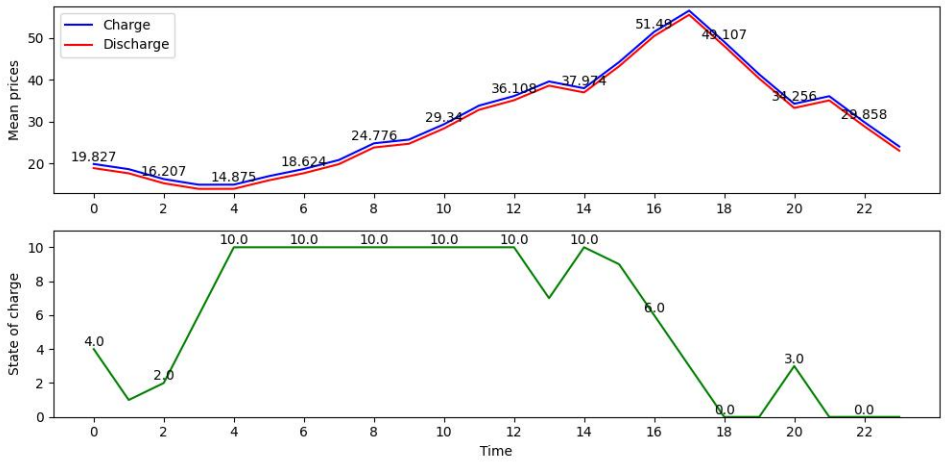

Figure 3.5: MVP solution for historical data collected from JPM ISO, June and July 2023, hourly real time, expected arbitrage = 427.69

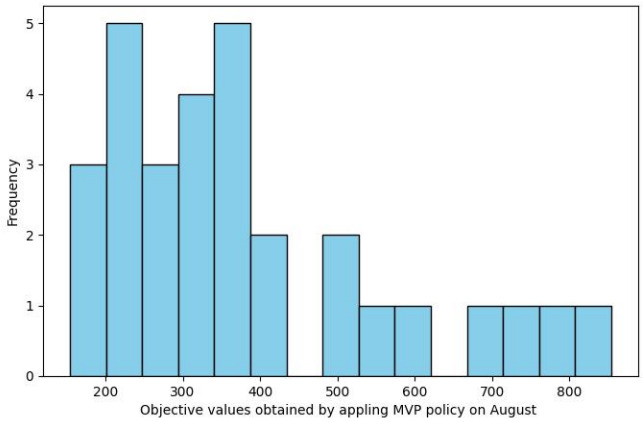

Figure 3.6: MVP policy averaged on June and July prices tested on August

3.4 Two stage stochastic model

An extensive form of a two stage stochastic linear program corresponding to the MILP model (2.1) under price uncertainty can be written as

$$\max \quad \left(\sum_{t=1}^{T_1} (P_t^d d_t - P_t^c c_t) + \sum_{s \in \mathcal{S}} \sum_{t=T_1+1}^{T_2} \mathbb{P}_s (P_{st}^d d_{st} - P_{st}^c c_{st}) \right) \tag{3.13a}$$

$$\text{s.t.} \quad s_t = \eta s_{t-1} + c_t - d_t \qquad \forall t \in \mathcal{T}_1, \tag{3.13b}$$

$$s_0 = S_{\text{init}}, \tag{3.13c}$$

$$\underline{S} \leq s_t \leq \overline{S} \qquad \forall t \in \mathcal{T}_1, \tag{3.13d}$$

$$0 \leq c_t \leq \overline{C} u_t \qquad \forall t \in \mathcal{T}_1, \tag{3.13e}$$

$$0 \leq d_t \leq \overline{D} v_t \qquad \forall t \in \mathcal{T}1, \tag{3.13f}$$

$$u_t + v_t = 1 \qquad \forall t \in \mathcal{T}_1, \tag{3.13g}$$

$$u_t, v_t \in \{0, 1\} \qquad \forall t \in \mathcal{T}_1, \tag{3.13h}$$

$$s_{st} = \eta s_{st-1} + c_{st} - d_{st} \qquad \forall s \in \mathcal{S}, \forall t \in \mathcal{T}_2, \tag{3.13i}$$

$$\underline{S} \leq s_{st} \leq \overline{S} \qquad \forall s \in \mathcal{S}, \forall t \in \mathcal{T}_2, \tag{3.13j}$$

$$0 \leq c_{st} \leq \overline{C} u_{st} \qquad \forall s \in \mathcal{S}, \forall t \in \mathcal{T}_2, \tag{3.13k}$$

$$0 \leq d_{st} \leq \overline{D} v_{st} \qquad \forall s \in \mathcal{S}, \forall t \in \mathcal{T}_2, \tag{3.13l}$$

$$u_{st} + v_{st} = 1 \qquad \forall s \in \mathcal{S}, \forall t \in \mathcal{T}_2, \tag{3.13m}$$

$$u_{st}, v_{st} \in \{0, 1\} \qquad \forall s \in \mathcal{S}, \forall t \in \mathcal{T}_2. \tag{3.13n}$$

In the above formulation $\mathcal{T}_1 = \{1, 2, \ldots, T_1\}$ is the set of all time periods within the first stage and $\mathcal{T}_2 = \{T_1 + 1, T_1 + 2, \ldots, T_2\}$ is the set of time periods in the second stage. The set $\mathcal{S}$ represents the set of price scenarios for the second stage. The probability of observing a scenario $s \in \mathcal{S}$ is denoted by $\mathbb{P}_s$. The variables and parameters with subscript st indicate the respective variable/parameter for scenario s at time period t.

The optimal objective of the stochastic program (3.13) and its corresponding optimal charge/discharge operational policy of the battery significantly depend on the scenario generation algorithms employed. Some methods rely on historical and current price data, while others assume some information about future prices. Naturally, assuming future price information results in a higher optimal objective compared to using only past and present data. We consider two different approaches for generating prices scenarios.

The first approach which we refer to as *lookahead* approach is the following. We make first stage decisions, then see all or some partial future information about second stage prices at time T_1, then make all second stage decisions with those future information. The lookahead approach allows us to anticipate future price scenarios at the end of the first stage.

The first approach allows us to look into the future by generating price scenarios at the end of the first stage. At time $t = T_1$, we have partial or complete information about the prices from $t = T_1 + 1$ to $t = T_2$, i.e. we know which scenarios for the second stage will occur and what the corresponding prices will be. In this approach, the probability of each scenario is uniformly distributed, and it is calculated as one divided by the sample size. Figure 3.8 illustrates the price trajectories for the first and second stage time periods, with each line in the second stage representing a different price scenario.

The second approach is by averaging over all sampled price trajectories conditioned on observed prices at the end of first stage times. In this case at time T_1 we don't have complete information about future prices, we can only predict second stage prices in expectation based on the information that are revealed up time T_1. To explain this approach clearly we define conditional expectation.

Definition 3.4. *(Conditional expectation) Let X and Y be discrete random variables and $Y = y$ be an event with nonzero probability. The conditional expectation of X*

given event $Y = y$ *is*

$$\mathbb{E}[X|Y = y] = \sum_x x \ \mathbb{P}(X = x|Y = y).$$

In our problem X represents possible price trajectories in the second stage and Y represents possible prices observed at the end of the first stage, i.e. at time T_1. We generate scenarios for the second stage prices based on the conditional expectation.

For the first stage we sample prices and compute the empirical mean at each time period similar to sampling and averaging in the case of MVP. The second stage scenarios are parameterized by the observed prices at time T_1 while sampling first stage price trajectories. The observed prices at this time is a random variable which we denoted by Y. Suppose we observed a particular price P_{T_1}. So the event of interest is $Y = P_{T_1}$. Conditioned on this event meaning starting with this price, we generate a price trajectory for the second stage time periods according to our price models. We denote this price trajectory by $X = x$. While sampling first stage price trajectories we keep track of the number of times that we observe the price P_{T_1} in order to compute the likelihood of the scenario that corresponds to the event $Y = P_{T_1}$. Now the scenario parameterized by this price is the expectation over all such sampled trajectories

$$\mathbb{E}[X|Y = P_{T_1})]$$

with probability

$$\frac{\#P_{T_1}}{K} \tag{3.14}$$

where $\#P_{T_1}$ denotes the number of times we observed price P_{T_1} and K is the sample size. The Figure 3.9 represents this idea of conditional averaging. A gray line is a sampled price trajectory and a bold line is the average of the respective gray lines. As the figure suggests the number of sampled trajectories starting from an observed price at time T_1 are different. To have a meaningful averaging we remove rare events.

If a price at time T_1 is not observed or rarely observed we ignore it. Algorithm 1 describes the procedure of the parameterized scenario generation explained above. This approach which we refer to it as *conditional expected value* (CEV), is an interpolation between the price generation for the MVP and SDP. We investigate SDP in the next section. We will see later in this section that the optimal objective of the two stage program solved with CEV price generation is larger than the one obtained from MVP and smaller than the SDP optimal objective.

Algorithm 1: CEV scenario generation algorithm for the two stage stochastic program; an interpolation between MVP and SDP

Input: K: Sample size
Output: Price trajectories for first and second stages with the corresponding probabilities

for $k \leftarrow 1$ *to* K **do**
  Sample price trajectory from $t = 1$ to $t = T_1$;
  $P_{T_1} \leftarrow$ observed price at $t = T_1$;
  Add 1 to the count of price P_{T_1} ;
  Sample a price trajectory from $t = T_1 + 1$ to $t = T_2$ conditioned on the observed price P_{T_1} ;
  Store P_{T_1} and the corresponding sampled trajectory ;
end
Compute the empirical mean of the price trajectories for first stage time periods over the sample size K;
Generate scenario: compute the empirical mean of all sampled trajectories parameterized by P_{T_1} for second stage over the number of its occurrence (remove rare events);
Compute the probability of a scenario parameterized by P_{T_1} according to 3.14 ;
Return the mean price trajectory for first stage and the price scenarios for the second stage with their corresponding probabilities ;

3.4.1 Computational studies

Assume the initial price of charge is 38. In the case where our model of prices is unbounded the possible prices that we can observe at time T_1 are all integers within the range of $38-T_1$ and $38+T_1$. So for $T_1 = 12$ there would be only 24 possible prices

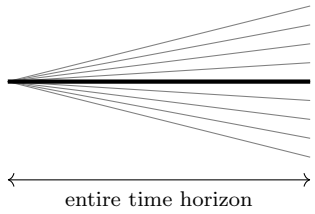

Figure 3.7: MVP price generation

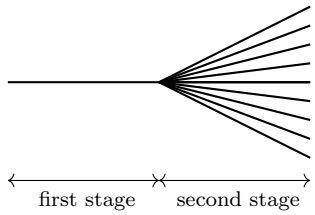

Figure 3.8: Look ahead price generation for the two stage program.

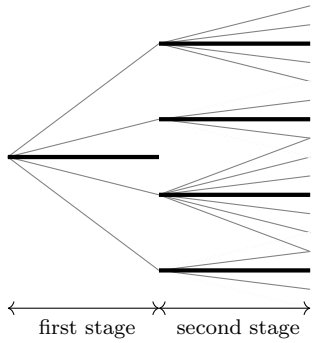

Figure 3.9: CEV scenario generation for the two stage program.

at time T_1. This means that the number of scenarios is at most 24. This number is significantly smaller than the sample size which is 10000 in our computations, however, the obtained profit from it is a number between the MVP optimal objective and the DP. Also, as a result of small number of scenarios the computational in this case is much lower than the computational time in the case of look into future. Figures 3.10, ..., 3.13 represent the computational results of the scenario generation and optimization of the two stage stochastic program in the four cases of our models of prices. In order to have a meaningful averaging we removed rare scenarios. Specifically for a sample size of 100000 we removed scenarios with less than 50 total counts.

The table 3.1 shows the optimal objective and the solution time of two stage stochastic program for the two models of price generations: look into future and not look into future. The optimal objectives when we are allowed to look into future are higher and overestimated due to the information that we are incorporating into our model. When we don't allow incorporating the exact prices in each scenario and not look into future the optimal objective are naturally and realistically smaller. It seems more realistic when we only know the prices of the first stage at time T_1 based on the historical data. The observed data at time T_1 only allows us to construct a conditional mean value trajectory for the second stage and does not provide a complete information about the price scenarios.

Since the number of scenarios that the CEV algorithm generates, significantly smaller than the number of scenarios in the lookahead case, the computational time of solving the two stage stochastic program with the former scenarios is also significantly smaller than the latter one. This is desirable since with less computations we produce more realistic approximation of the optimal expected arbitrage. The CEV algorithm can be seen as a scenario generation and scenario reduction at the same time.

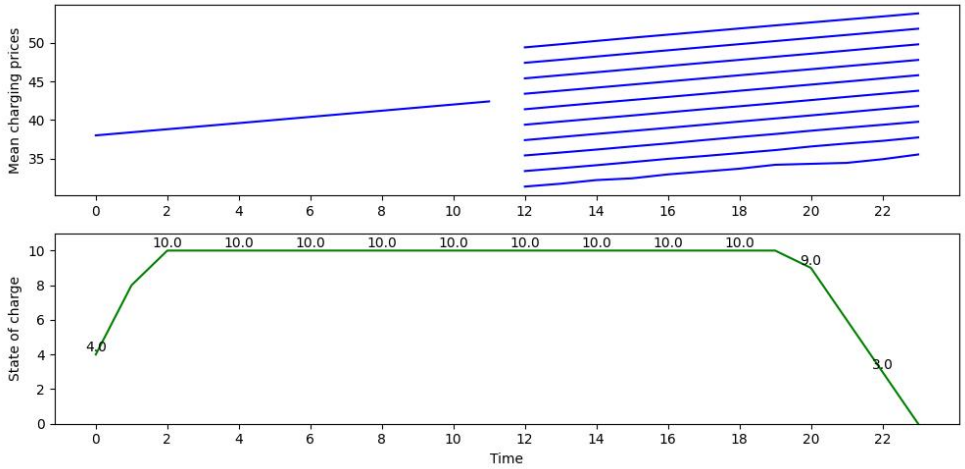

Figure 3.10: Two stage stochastic solution for unbounded prices with 0.7 probability of going up. Total profit = 225.930, number of scenarios = 9, sample size = 100000.

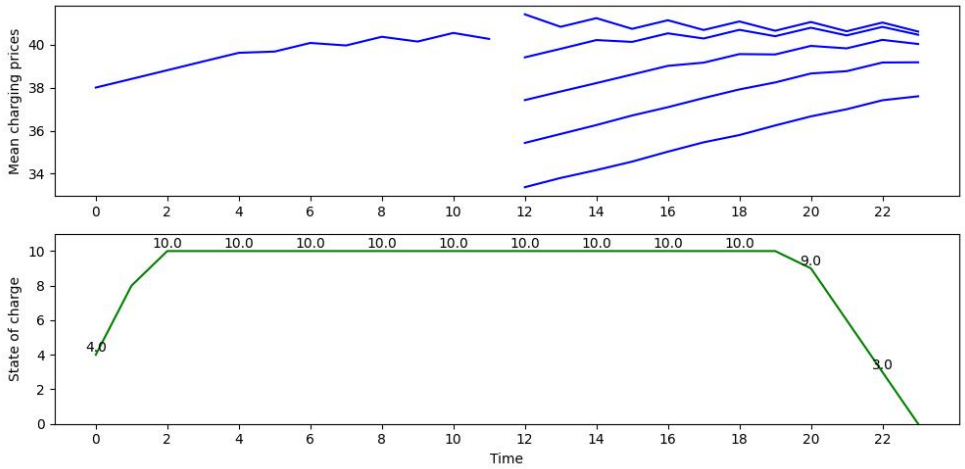

Figure 3.11: Two stage stochastic solution for unbounded prices with 0.7 probability of going up. Total profit = 168.791, number of scenarios = 5, sample size = 100000.

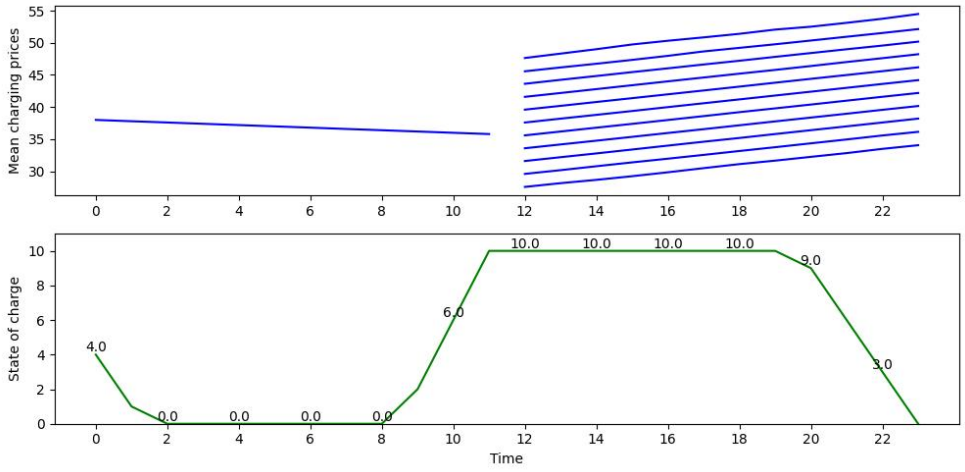

Figure 3.12: Two stage stochastic solution for unbounded prices with 0.45 and 0.8 probability of going up. Total profit = 206.645, number of scenarios = 11, sample size = 100000.

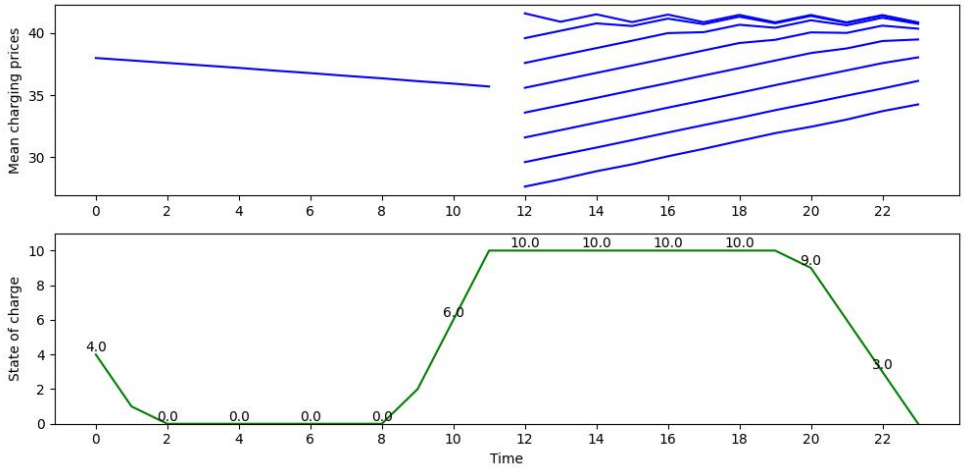

Figure 3.13: Two stage stochastic solution for unbounded prices with 0.45 and 0.8 probability of going up. Total profit = 172.220, number of scenarios = 7, sample size = 100000.

Table 3.1: Comparison of price scenarios: look into future and CMV. The number of scenarios for look into future = 10000, for CMV < 12 averaged over a sample of size 100000.

Price model	RP(LKAD)		RP(CEV)	
	obj	time(s)	obj	time(s)
Unbounded time-homogeneous	272.559	17.17	225.897	2.60
Bounded time-homogeneous	180.787	16.38	168.735	2.75
Unbounded time-inhomogeneous	208.912	17.05	200.160	2.61
Bounded time-inhomogeneous	201.524	16.89	179.544	2.77

3.5 Stochastic dynamic programming

We formulate the battery charging problem as a finite horizon Markov decision process (MDP) and solve it using the method of stochastic dynamic programming (SDP). An MDP provides a framework for modeling our problem where outcomes are partly random; prices in our study; and partly under the control of the decision-maker; charge/discharge decision variables. The key assumption is the Markov property, which states that the future state depends only on the current state and action, not on the history of past states and actions.

Dynamic programming is a general solution method for problems that have optimal substructure, i.e. the optimal solution can be decomposed into subproblems; and the subproblems overlap in the sense that subproblems recur many times so solutions can be reused. An MDP satisfies these properties as the Bellman equation provides recursive decomposition and the value function stores and reuses the solutions. An SDP extends Dynamic programming to handle problems involving uncertainty. It is used for decision-making under uncertainty where outcomes are probabilistic. An SPD assumes a perfect model of the environment dynamics as a MDP and using it one can compute sequentially optimal policies and predict the expected value of such policies. In the following we define formally the terms mentioned above.

Definition 3.5. *(Markov decision process) A Markov decision process is defined as a tuple* $(X, A, \mathbb{P}, r, \gamma)$ *where*

- X *is the state space*
- A *is the action space*
- $\mathbb{P}(y \mid x, a)$ *is the probability of transitioning (i.e., environment dynamics) from state* $x \in X$ *to state* $y \in X$ *performing action* $a \in A$
- $r(x, a, y)$ *is the reward obtained when taking action* a *during the transition from state* x *to state* y

- γ *is the discount factor.*

A policy in the context of an MDP is a strategy or a rule that defines the actions that we will take in each state to achieve our goals. Formally,

Definition 3.6. *(Policy) A policy is a function from a state to a set of actions.*

If the policy is deterministic, it specifies a single action for each state. If the policy is stochastic, it provides a probability distribution over possible actions for each state. An optimal policy is a policy that maximizes the expected cumulative reward.

Next we state Bellman optimality conditions for a general dynamic programming problem and then explain the specifics about the battery charging problem under price uncertainty. The Bellman optimality conditions are a set of necessary conditions for optimality. For a given dynamic programming problem, the Bellman optimality principle is stated as follows.

Bellman principle of optimality: An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. Informally the statement means that the optimization of the future does not depend on what we did in the past. If we have a policy that is optimal overall, then no matter where you are in the process (whatever state we are in, after making an initial decision), the remaining strategy should also be optimal from that point onward.

Bellman equation: The Bellman equation is a recursive equation that characterizes the optimal (expected) value function in terms of smaller subproblems. For a deterministic problem, the equation is typically expressed as:

$$V^*(x) = \max_a \{r_x(a) + \gamma V^*(y)\} \tag{3.15}$$

where $V^*(x)$ is the optimal value at state x, $r_x(a)$ denotes the instance reward of taking action a in the state x, $\gamma \in [0, 1]$ is the discount factor and $\mathbb{P}_{xy}[a]$ is the transition probability from state x to state y.

The Bellman equation is modified to account for the uncertainty the problem. Instead of a single deterministic transition function and reward function, An SDP considers probability distributions over possible transitions and rewards. Consequently, the Bellman equation becomes an expectation over these distributions. In the stochastic setting, the Bellman equation is modified to account for the uncertainties, incorporating the expected value over possible outcomes of the stochastic transitions and rewards. The typical form of the Bellman equation for a stochastic problem is expressed as an expectation over possible future states and rewards and it reads as follows:

$$V^*(x) = \max_a \left\{ R_x(a) + \gamma \sum_y \mathbb{P}_{xy}[a]\ V^*(y) \right\} \tag{3.16}$$

where $R_x(a)$ is the expected reward during stage x performing action a. The maximization problem on the right hand side of the Bellman equation provides optimal value function given state x. The argmax function over all possible actions of the right hand side provide an optimal policy given state x.

In the context of the battery charging problem the goal is to find optimal value function at initial time which is the maximum total expected arbitrage. We are also looking for an optimal policy i.e. charge/discharge decisions at each time period that maximizes the expected value function over future times. In the following, we describe a MDP framework adapted to our problem.

State space: In general construction of state space is a subtle concept because it involves carefully defining the states in such a way that they capture all the relevant information needed to make optimal decisions. This often requires a deeper understanding of the problem and the dynamics of the system being modeled (Powell and Meisel, 2015). The state space of a battery can be described by the time period t, the state of charge variable s_t and the prices of charge and discharge P_t^c and P_t^d.

$$\mathcal{S} = \{(t, s_t, P_t^c, P_t^d)\ :\ t \in \mathcal{T}\}$$

The state description consists of a controllable component s_t, and information components P_t^c and P_t^d. The stochasticity of our problem arises from the information component of the defined states. For simplicity of the state space we assume the following deterministic relation between prices of charge and discharge:

$$P_t^d = P_t^c - 1.$$

We discretize the state variables into equi-distance intervals. Without loss of generality we let minimum state of charge $\underline{S} = 0$. Also we assume the maximum state of charge $\overline{S}$ and the initial prices P_0^c and P_0^d to be integers. Thus at each time t we have

$$\begin{aligned} s_t &\in \{0,\ 1,\ \ldots,\ \overline{S}\} \\ P_t^c &\in \{P_0^c,\ P_0^c \pm 1,\ \ldots,\ P_0^c \pm t\} \\ P_t^d &\in \{P_0^d,\ P_0^d \pm 1,\ \ldots,\ P_0^d \pm t\}. \end{aligned}$$

If the $\overline{S}$ or initial prices are not integer by rescaling them we can assume integrality. We remind that according to our model of prices with Markov property we have $P_{t+1}^c - P_t^c = \pm 1$ and $P_{t+1}^d - P_t^d = \pm 1$. Notice that the state space is non-stationary since the state variables change over time.

Action space: The action space consist of all possible amounts of charges and discharges at each time period. There are three types of actions each time period in the action space: charge, discharge, and do nothing meaning zero charge and zero discharge. A charge or discharge action can be full or partial with respect to the maximum capacity rates. In general partial charge/discharge can be any number between 0 and the maximum charge/discharge capacities. However, with the assumption of integrality of $\overline{C}$ and $\overline{D}$ without loss of generality, the set of possible

and optimal partial charge and discharge can fall into the following finite sets .

$$\text{the amount of partial charge } \in \{1, 2, \ \ldots, \ \overline{C} - 1\} \tag{3.17}$$

$$\text{the amount of partial discharge } \in \{1, 2, \ \ldots, \ \overline{D} - 1\} \tag{3.18}$$

Therefore the action space is set of size $\overline{C} + \overline{D} + 1$ containing the actions: full charge/discharge, partial charges/discharges given in (3.17) and (3.18) and do nothing.

$$\mathcal{A} = \{c_1 = 1, c_2 = 2, \ldots, c_{\overline{C}} = \overline{C}, d_1 = 1, d_2 = 2, \ldots, d_{\overline{D}} = \overline{D}, \text{ do nothing}\} \tag{3.19}$$

This is a coarse but optimal discretization of the action space given the integrality assumptions. By optimality of the discretization we mean finer discretization will not effect the solution quality. The optimality of the above partial charge/discharges is due to the formula 2.6 in the previous chapter which we restate it here.

$$\text{the optimal amount of partial charge/discharge} = S_{\text{final}} + \overline{D}n^d - \overline{C}n^c - S_{\text{init}} \tag{3.20}$$

In the above formula, n^d, n^c the number of full discharge and charge are integers. If the parameters $\overline{C}$ and $\overline{D}$ are assumed to be integers then the amount of partial charge/discharge will be integers as well.

It is worth mentioning that other democratization's of action space are possible. For instance $\mathcal{A} = \{\min\{\overline{C}, \overline{S} - s_{t-1}\}, \ \min\{\overline{D}, s_{t-1} - \underline{S}\}, \ 0\}$ is considered in (Wang and Zhang, 2018). The size of this discretization is smaller, however it is suboptimal since it does not provide flexibility to populate the partial charge/discharge according to the prices. For example according to above action space, if the battery is at full capacity at time t, in the optimal policy we will never observe a partial discharge of 1 at time $t + 1$. This situation happens in the optimal policy of MVP, see for example Figure 3.1 time $t = 19$.

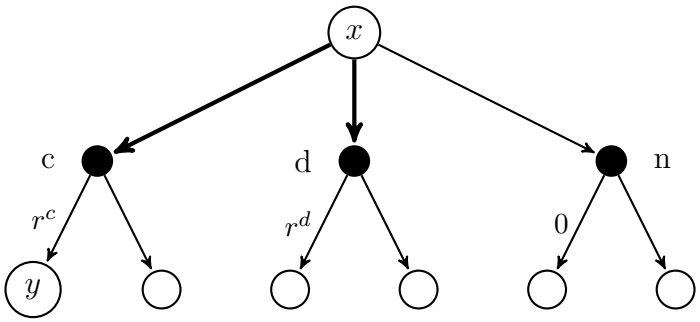

Figure 3.14: State transition diagram in stochastic dynamic programming. White circles represent states and black circles represent actions

Figure 3.14 shows the state transition diagram for our SDP. State x at time t with given probabilities associated to each action transitions to a possible new state y at time $t+1$. The bold arcs indicate that there are sets of actions of type charge and discharge. As shown in the Figure we list all possible new states with their probabilities to be able to use Bellman equation to compute the optimal value of the state x.

Remark. Regardless of positive or negative prices we choose only one action at each time period. That means we avoid simultaneous charge and discharge at a time period and the integrality constraint is satisfied.

State transition probabilities: The transition probabilities define the likelihood of moving from one state to another, given a particular action. In our problem we assume the state variables evolve randomly and according to our Markovian model of prices. The state variables s_t, P_t^c and P_t^d evolve according to the following transition equations:

$$
\begin{aligned}
s_{t+1} &= s_t \pm a_t \quad a_t \in \mathcal{A} \\
\mathbb{P}\{P_{t+1}^c = P_t^c + 1\} &= \mathbb{P}\{P_{t+1}^d = P_t^d + 1\} = \mathbb{P}_t(H) \\
\mathbb{P}\{P_{t+1}^c = P_t^c - 1\} &= \mathbb{P}\{P_{t+1}^d = P_t^d - 1\} = \mathbb{P}_t(L)
\end{aligned}
$$

In above equations $\mathbb{P}_t(H)$ and $\mathbb{P}_t(L)$ indicate probability of price increase and decrease at time period t, respectively. In time-homogeneous case, these probabilities are fixed for all $t \in \mathcal{T}$.

Reward: There are three different types of instance reward at time t which correspond to the three types of actions described above. At each time period the

instance rewards can be collected as the following.

$$r_t = \begin{cases} r_t^c = -P_t^c c_t & \text{if charge} \\ r_t^d = P_t^d d_t & \text{if discharge} \\ 0 & \text{if do nothing} \end{cases}$$

The amount of charge c_t and the amount of discharge d_t can be full or partial with respect to the maximum capacity rates. The instance reward of charging and discharging at time t are denoted by r_t^c and r_t^d respectively. As shown in Figure 3.14 instant rewards are computed after taking a possible action. Then according to the Bellman equation the expected reward associated to the performed action is computed.

Policy: As we defined in 3.5 a policy denoted by π is a function from state space to action space. The policy that the Bellman equation follows while navigating the environment is the optimal policy. We seek to determine the optimal policy π^* for charging and discharging decisions at each time period, aiming to maximize the expected value function over future time periods. We identify this optimal policy by applying the argmax function over all possible actions for a given state.

$$\pi^*(x) = \arg\max_a \left\{ R_x(a) + \gamma \sum_y \mathbb{P}_{xy}[a]\ V^*(y) \right\} \tag{3.21}$$

Computational studies: Here we present our computational experiments for the battery charging problem under price uncertainty for our price models defined in section 3.2. We use the same parameters defined in subsection 3.2.1. For $T = 24$ and $\overline{S} = 10$, the size of state space is $24 \times 11 \times 2(24)$. Since the maximum charge and discharge rates are set to 4 and 3 respectively, and their minimums are set to 0, according to our discussion for defining the action space, we have a set of actions of size 8. Four actions of type charge, three actions of type discharge plus a do nothing

action.

$$\mathcal{A} = \{c_1 = 1, c_2 = 2, c_3 = 3, c_4 = 4, d_1 = 1, d_2 = 2, d_3 = 3, \text{ do nothing}\} \tag{3.22}$$

Figures 3.15, ..., 3.18 illustrate the result of our experiments for the four models of prices. Time $t = 0$ correspond to initial state of the battery with state of charge of 4. Each of these figures for a given and unseen price trajectory show the optimal policy by evolution of state of charge of the battery, the optimal value function and the expected arbitrage starting time t onward, conditioned on the realized price trajectory up to time t. In each case, the price trajectory is generated randomly but follows to the same corresponding distribution for that case. For example Figure 3.15 corresponds to unbounded time-homogeneous model which the transition probability is fixed to 0.7 for the growth of the prices. To test the result of value function computation we generate a random price trajectory that follows the exact distribution. The optimal policy clearly involves charging the battery to full capacity at the beginning and maintaining it at full charge until the end of the time horizon when prices are high, at which point the battery is fully discharged.

The value function at the end of time period, $t = 23$ in the figure initialized with value of 0 for all possible states at this time. Then at each time going backward, the value function is updated according to Bellman equation. Notice that value function for a given state at time t computes the expected income starting from time t onward. According to the Bellman optimality principle, the value function maximizes future rewards independent of past events. To calculate profit rather than income, we account for past expenses based on realized prices and incorporate these into the value function. This approach explains how expected arbitrage is computed for each time period.

We are mainly interested in the value function at time 0, when we want to know with the initial state of charge what is the expected arbitrage. However, following the optimal policy it might of interest to know the amount of expected arbitrage at

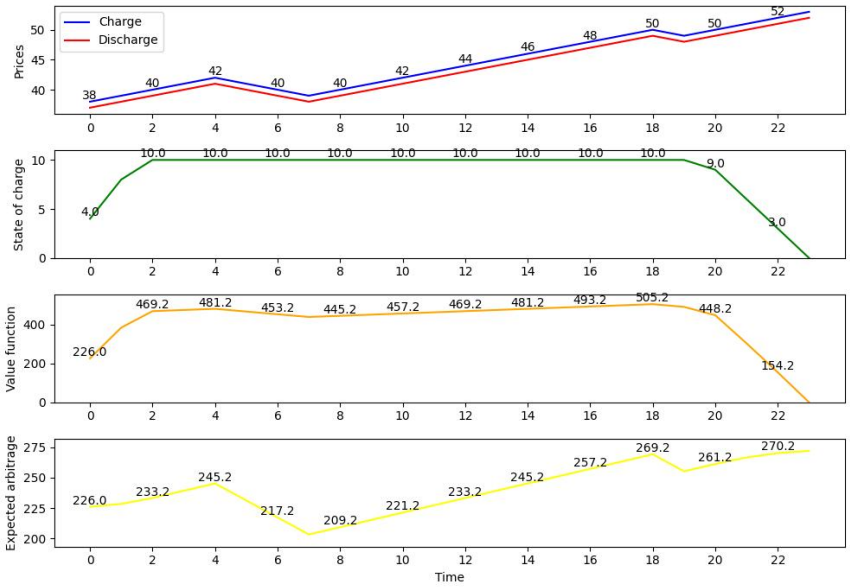

Figure 3.15: SDP solution for unbounded time-homogeneous prices with 0.7 probability of going up. Total profit = 226.0, sample size = 10000.

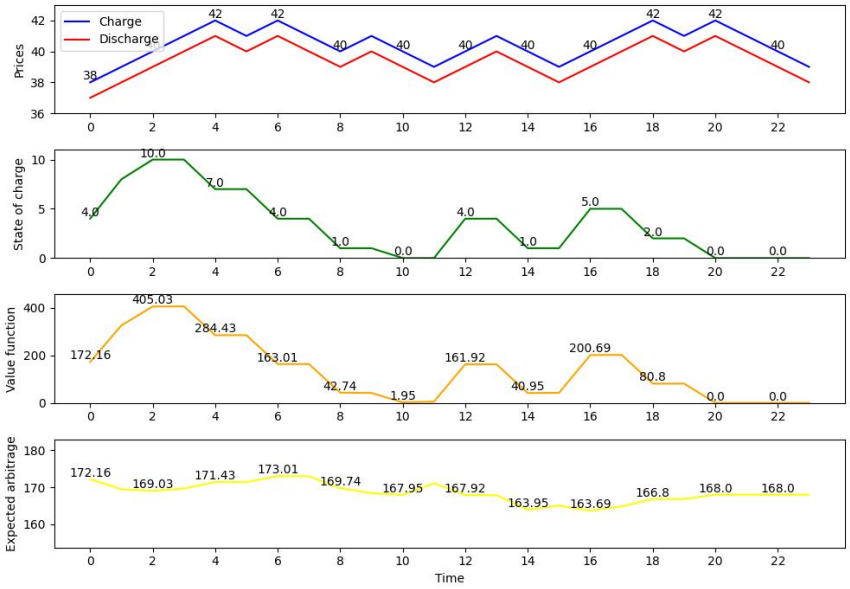

Figure 3.16: SDP solution for bounded time-homogeneous prices with 0.7 probability of going up. Total profit = 172.16, sample size = 10000.

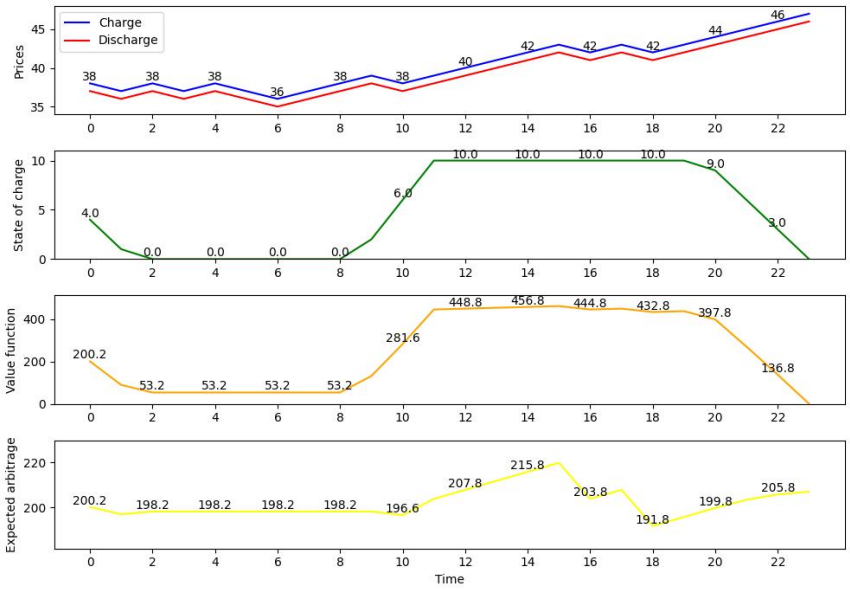

Figure 3.17: SDP solution for unbounded time-inhomogeneous prices with 0.4 and 0.8.

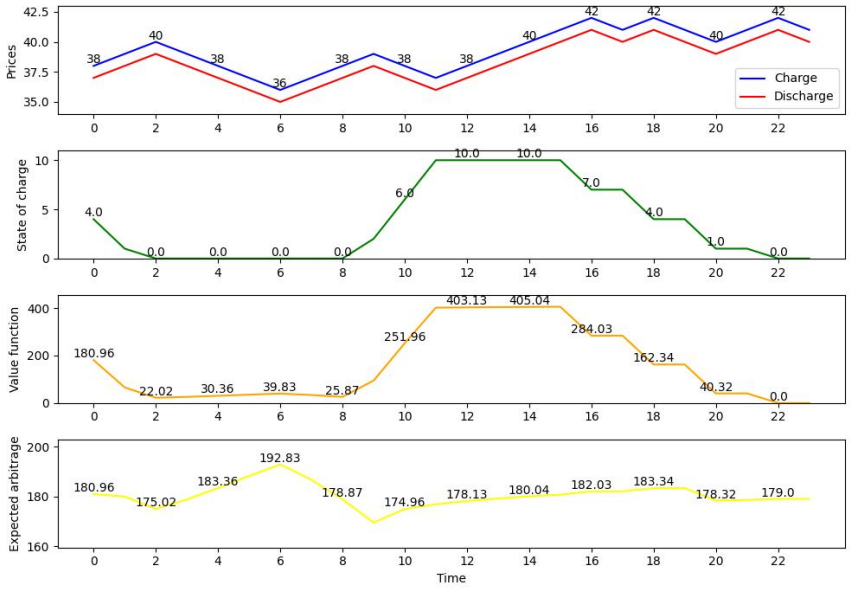

Figure 3.18: SDP solution for bounded time-inhomogeneous prices with 0.4 and 0.8.

other time periods. Also notice that in all four cases the expected arbitrage is more concentrated over the entire time horizon compared to value function.

Figure 3.16 corresponds to bounded time-homogeneous prices with the same probability transition as in unbounded case. The optimal policy in this case is more complex than the unbounded case. Here the upper bound for the charging price is set to 42. Whenever the price of charge reaches to this threshold the price reflect down by one unit.

Figure 3.17 corresponds to unbounded time-inhomogeneous prices with probability transition of growing by 0.4 for the first half and 0.8 for the second half. The random prices decrease slightly in the first half and start growing in the second half. For the second half the optimal policy is similar to Unbounded time-homogeneous case. Figure 3.18 illustrates bounded time-inhomogeneous prices with the same probability transitions. The introduced upper bound on the prices leads to more complex optimal policy for the second half of the time horizon compared to the unbounded case.

The difference between bounded and unbounded cases: In our experiments, we observed that SDP for unbounded price produces a price independent and deterministic policy, meaning that when SDP algorithm is applied to different price trajectories, the resulting policies remain identical. One explanation is that the charge/discharge decisions depends on the distribution of the difference between current and future prices. The unbounded prices have the following property.

Definition 3.7. *(PII property) We say that a price trajectory has the price independent increment (PII) property if the distribution of the difference* $P_t - P_{t-1}$ *does not depend on* P_{t-1}.

Figure 3.19 shows the distribution of the price difference at each time period t in unbounded time-homogeneous case where the probability of price increase is 0.7. This distribution remains identical throughout the time horizon, irrespective to the current price. Consequently, using the Bellman equation it can be shown that there

exists a price independent function f such that

$$\begin{aligned}\pi^*(t, s_t, P_t) &= \arg\max_a \{f(t, s_t, a) + s_t P_t\} \\ &= s_t P_t + \arg\max_a \{f(t, s_t, a)\}\end{aligned}$$

As a result, the optimal policy does not change when the price trajectories are different.

Algorithm 2: An algorithm to check the similarity of optimal policy for unbounded price trajectories

Input: The optimal policy π^* obtained by SDP.
Output: The Truth value of price independence property of the optimal policy of unbounded price trajectories

```
for t ∈ 𝒯 do
    for s ∈ {0, 1, ..., S̄} do
        A = {} ;
        for P ∈ {P_0, P_0 ± 1, ..., P_0 ± T} do
            Add π*(t, s, P) to the set A ;
        end
        if size A != 1 then
            Return False
        end
    end
end
Return True ;
```

We can also confirm its correctness empirically utilizing the algorithm 2. In Appendix A, we include more figures that show distinct price trajectories with identical optimal policy.

For bounded price trajectories the above property is no longer valid. By defining an upper bound on the prices we introduce price dependency for the optimal policy. For instance when we reach a state where the prices are at upper bound, in the next state the prices will decrease with probability 1 and the only optimal action is to discharge the battery. Hence the distribution of the price difference will change. As shown in Figure 3.20 in this case there are two distinct distributions for the price

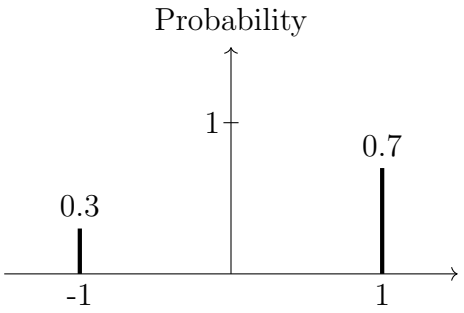

Figure 3.19: The distribution of price difference $P_t - P_{t-1}$ for all $t \in \mathcal{T}$ for unbounded time-homogeneous prices

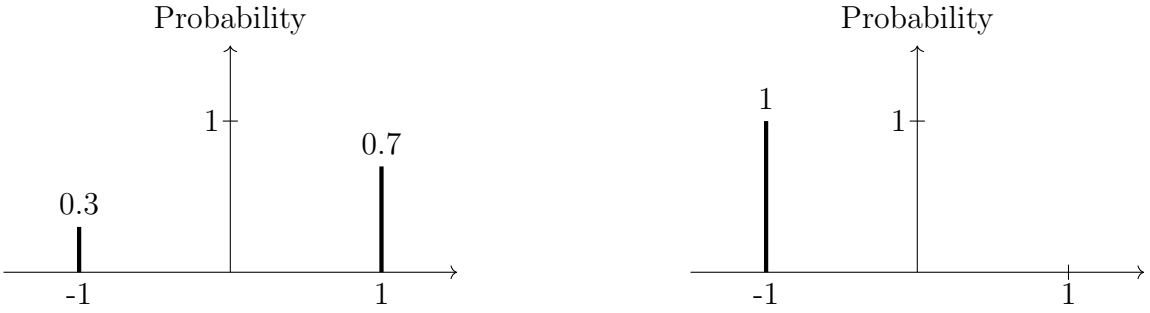

Figure 3.20: The distributions of price difference $P_t - P_{t-1}$ for all $t \in \mathcal{T}$ for bounded time-homogeneous prices. Left: the price is strictly smaller than upper bound. Right: the price is at upper bound.

changes. In Appendix A we include figures with different bounded price trajectories and distinct optimal policies.

Solution quality of MVP, RP, and SDP: Tables 3.2 and 3.3 show the optimal expected arbitrage obtained by solving MVP, two stage stochastic with CMV scenario generation algorithm and SDP. The MVP provides a lower bound for the arbitrage and SDP provides an upper bound. The objective of the two stage stochastic program, RP(CMV), by construction, is an interpolation between the MVP and SDP.

Remark. Recall that backcasting technique in the context of battery charge problem consists in deriving hourly charge and discharge operations by solving MVP, and then estimating the arbitrage value using an unseen hourly price trajectory. We do not apply backcasting since in our mathematical models there is no distinction between the distributions of past and current price trajectories.

When prices are unbounded, the expected arbitrage in all three models, ignoring the round of error, is essentially the same. As previously discussed, the optimal policy of unbounded price trajectories in SDP is price independent. Similarly, the MVP, as a deterministic program, produces price independent policy. Moreover, the optimal policies of SDP and MVP are identical at each time period. Given that RP(CEV) interpolates between the MVP and SDP, then optimal policy and optimal objective of all three methods are identical. Note that in this case the value of stochastic solution, VSS, defined in preliminaries section is zero.

However, the optimal policy of bounded prices are different when the price trajectories are distinct. Therefore the optimal objective of SDP is different and larger than the optimal objective of MVP. For our choice of parameters, the relative gap between the MVP and SDP optimal objectives computed by formula

$$\frac{|SDP - MVP|}{MVP} \times 100\% \tag{3.23}$$

Table 3.2: Comparison of optimal expected arbitrage obtained by MVP, two-stage stochastic with CMV scenario generation, and SDP. Initial price of charge = 38, upper bound = 42, probabilities of price increase = $0.4, 0.8$ for first half and second half of the time horizon respectively.

Price model	MVP	RP(CEV)	SDP	relative gap
Unbounded time-homogeneous	226.106	225.897	226.085	0.01
Bounded time-homogeneous	167.355	168.735	172.160	2.87
Unbounded time-inhomogeneous	200.225	200.160	200.20	0.01
Bounded time-inhomogeneous	178.624	179.544	180.960	1.31

Table 3.3: Comparison of optimal expected arbitrage obtained by MVP, two-stage stochastic with CMV scenario generation, and SDP. Initial price of charge = 5, upper bound = 9, probabilities of price increase = $0.45, 0.7$ for first half and second half of the time horizon respectively.

Price model	MVP	RP(CEV)	SDP	relative gap
Unbounded time-homogeneous	94.01	94	94.03	0.02
Bounded time-homogeneous	35.31	36.86	40.19	13.8
Unbounded time-inhomogeneous	48.72	49.03	49.56	1.72
Bounded time-inhomogeneous	31.08	32.67	35.29	13.5

is within 3%. However, higher gaps can be achieved by experimenting on probability transitions and the other parameters such as T and the upper/lower bound on the prices. Introducing more volatility on the prices leads to larger gap between MVP and SDP. One way to do this is to change the parameters the initial prices and the parameters of the Markov model of prices. For instance if we set the initial price of charge to 5 and the upper bound to 9 and the probabilities of price increase to 0.45 and 0.7 for the first and second half of the time horizon, then the gap between MVP and SDP reaches to 13%.

Another way to introduce volatility is by applying larger incremental changes to the current price to predict future prices. Instead of adjusting the current price by a small amount such as ± 1, using a larger variation like ± 10 can create significant fluctuations. This approach not only amplifies the volatility in price trajectories but also highlights the limitations of the MVP approach, which smooths out such variations and fails to capitalize on sudden price changes.

It is important to note that the uncertainty in our problem is confined only to the objective function, making it challenging to achieve substantial gaps between MVP and SDP. This limitation arises because the variability in the objective function alone may not be sufficient to create large discrepancies in performance between the two approaches. It is also not clear how to estimate and setup an upper bound on the gap between MVP and SDP.

The choice between MVP and SDP involves a trade-off between solution quality and computational expenses. MVP offers computational efficiency and faster solution but at the cost of less accurate and robust decisions. SDP, on the other hand, provides high-quality solutions by explicitly modeling uncertainty, but requires significantly more computational effort and resources.

It is worth mentioning that if a price trajectory is a martingale stochastic process then the expected arbitrage is zero at any time $t \in \mathcal{T}$.

Definition 3.8. *(Martingale) A discrete-time martingale is a discrete-time stochastic process (i.e., a sequence of random variables) X_1, X_2, X_3, ... that satisfies for any time t*

$$\mathbb{E}(X_{t+1} \mid X_1, X_2, \ldots, X_t) = X_t$$

That is, the conditional expected value of the next observation, given all the past observations, is equal to the current observation.

In probability theory there is a theorem called the optional stopping theorem saying that under certain conditions, the expected value of a martingale at a stopping time is equal to its initial expected value. This implies that on average, for martingale random processes nothing can be gained by stopping play based on the information obtainable so far i.e., without looking into the future.

In our models of prices, an example of a martingale is the following. Assume the prices are unbounded and the Markov process is time-homogeneous. If the probabilities of the price increasing or decreasing are both 0.5, then the price trajectory forms a martingale.

$$\mathbb{E}(P_{t+1} \mid P_0, P_1, \ldots, P_t) = \mathbb{E}(P_{t+1} \mid P_t) = 0.5(P_t + 1) + 0.5(P_t - 1) = P_t$$

The first equality above is due to the Markov property of the prices. As a result of the optional stopping theorem the expected arbitrage in this case is zero, meaning that no matter which charge and discharge policy we use, in expectation there will be no gain and no loss of profit.

3.6 Q-learning

As mentioned in previous section an SDP assumes full description of environment dynamics as part of an MDP. The probability transitions from one state to another state for all possible action should be known in advance. But what if we don't

know the dynamics of the environment which is usually the case. In our problem we don't know the exact probabilities of price fluctuations in the future. When we don't have access to full description of the MDP framework for a problem we use model-free approximation methods to estimate a solution. Monte Carlo and Q-learning algorithms are among popular methods for estimating the value function of each state and an optimal policy. Another reason for using these algorithms is that the state space and/or action space is large and SDP can not handle expensive computations to navigate through the entire space. Instead they sample trajectories inside the state space and evaluate and update the value function based on the observations. In this study we investigate Q-learning to solve the battery problem.

Q-learning is a popular model-free reinforcement learning algorithm , where an agent explores the environment without assuming a full description of MDP framework and finds the optimal way to maximize the cumulative reward. It is called model-free because the agent doesn't know the probability transitions between states. In other words, the agent does not have a predefined MDP model that describes the likelihood of moving from one state to another. Instead, the agent learns to make decisions based solely on the experiences and rewards it accumulates through interactions with the environment, without needing to understand the underlying transition dynamics.

In Q-learning, the agent only needs to know its state space and possible actions. Each state-action pair is assigned by a Q-value, which represents the quality of the action for the given state. When the agent carries out an action from the given state, it receives a reward and moves to a new state. The reward is used to update the Q-value using Bellman equation.

$$Q(x,a) \longleftarrow (1-\alpha)Q(x,a) + \alpha\left(r + \gamma \max_b Q(y,b)\right) \tag{3.24}$$

where $Q(x, a)$ is the Q-value for current state x and action a, α is the learning rate for convergence, γ discount factor for future rewards, and y is a possible future state reachable from state x.

The agent explores the environment across numerous episodes, a process known as episodic learning. In each episode, the agent starts from an initial state and performs several actions until reaching the goal state and updates its knowledge (Q-values). Choosing an action at each state within an episode is based on the ϵ-greedy policy, which is a way of selecting random actions with uniform distribution from a set of available actions. Using this policy, the agent can select an action with the current optimal policy with probability $1 - \epsilon$, (exploitation) or try other actions with probability ϵ (exploration) in a hope for a better reward resulting in improvement of Q-values and thereby, deriving better policy.

Algorithm 3: Q-learning algorithm

Input: Parameters: $\epsilon = 1$, $\alpha = 0.01$, $\gamma = 1$
Output: Q-values for each state-action pair

```
Initialize Q with 0 for all state-action pairs;
for each episode do
    for t ∈ 𝒯 do
        Choose state of charge s_t at random from the set {0, 1, ..., S̄};
        Choose a feasible action from the set 𝒜 at random (ϵ-greedy);
        Choose a day at random from the training data;
        Move to the next state and collect the reward;
        Update the current Q according to (3.24);
    end
end
Return Q;
```

In the context of battery charging problem under price uncertainty, each episode contains states, actions and rewards as well as a price trajectory over the time horizon T which is used for computing the reward at each time.

$$S_0, a_1, r_1, S_1, a_2, r_2, S_2, \ldots, a_T, r_T, S_T \tag{3.25}$$

The initial state S_0 remains fixed at the beginning of all episodes. The possible actions are the same as we saw in SDP assuming the integrality of $\overline{C}$ and $\overline{D}$.

As mentioned before construction of state variables is subtle concept. We experiment with the definition of state variables. One simple way to define the state space for our problem is to include only time and state of charge of the battery.

$$\mathcal{S} = \{(t, s_t) \ : \ t \in \mathcal{T}, \ s_t \in \{0, 1, \ldots, \overline{S}\}\} \tag{3.26}$$

where $\overline{S}$ is assumed to be integer. In the definition above, the states are price-independent, meaning that the stochastic component is ignored, allowing us to have full control over the state space. This choice of state variables impacts the optimal policy derived from running the Q-learning algorithm. Consequently, this policy will also be price-independent, leading to a case where the maximum achievable profit aligns with the optimal objective of the MVP.

Suppose we want to evaluate the optimal policy obtained by Q-learning, denoted by π^*_{Q} across a set of price trajectories $\mathcal{S}$. Assume that π^*_{Q} corresponds to $c^*_t(\text{Q})$ and $d^*_t(\text{Q})$ the optimal amount of charge and discharge at each time t. For each price trajectory $s \in \mathcal{S}$ we compute the inner product of the prices and the optimal charge and discharge decisions obtained by Q-learning algorithm. Then we compute the average according to the formula on the left hand side of the inequality (3.27).

$$\frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{T}} \left(P^d_{st} d^*_t(\text{Q}) - P^c_{st} c^*_t(\text{Q}) \right) \leq \sum_{t \in \mathcal{T}} \left(\overline{P^d_t} d^*_t(\text{MVP}) - \overline{P^c_t} c^*_t(\text{MVP}) \right) \tag{3.27}$$

In this inequality $\overline{P^d_t} = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} P^d_{st}$. The right hand side of the inequality uses the average price trajectories and the optimal charge and discharge decisions obtained by MVP, denoted by $c^*_t(\text{MVP})$ and $d^*_t(\text{MVP})$. In Q-learning algorithm as the number of episodes grows toward infinity, by law of large numbers, the optimal policy approaches to the optimal policy of MVP. In conclusion the state space defined in (3.26) does not improve the arbitrage achieved by MVP. To design a better policy than MVP

policy we have to reconsider the definition of the state variables. Any policy better than MVP policy should somehow be dependent on the prices.

To move beyond the price independent deterministic policy we define two alternatives for constructing state space. To maintain the Markov setting, we consider predefined price intervals and add them to the description of the state.

$$\mathcal{S} = \left\{(t, s_t, I_t) \ : \ t \in \mathcal{T}, \ s_t \in \{0, 1, \ldots, \overline{S}\}, \ P_t^c \in I_t\right\} \tag{3.28}$$

In the state space described above, I_t denotes the interval that contains the price of charge at time t.

Another way to define the state space is to use second order Markov process (Diaconis and Miclo, 2013) where the next state depends on the previous two states, rather than just the previous one.

Definition 3.9. *(Second order Markov process) A discrete-time second order Markov process is a sequence of random variables $X_1, X_2, X_3, \ldots$ with property:*

$$\mathbb{P}\left(X_{t+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_t = x_t\right) = \mathbb{P}\left(X_{t+1} = x \mid X_{t-1} = x_{t-1}, X_t = x_t\right) \tag{3.29}$$

$$\mathcal{S} = \left\{(t, s_t, I_{\Delta_t}) \ : \ t \in \mathcal{T}, \ s_t \in \{0, 1, \ldots, \overline{S}\}, \ \Delta_t = P_t^c - P_{t-1}^c \in I_{\Delta_t}\right\} \tag{3.30}$$

In the state space above, I_{Δ_t} is the interval that contains the price difference $\Delta_t = P_t^c - P_{t-1}^c$ at time t.

In cases where the transition probabilities are known, the optimal policy derived from Q-learning converges to the SDP optimal policy as the number of episodes approaches infinity (Watkins, 1989). Therefore, we anticipate that, given a sufficient number of episodes, the optimal objective and the optimal charge/discharge decisions for the battery across our four price models will be identical to those obtained through

SDP. Our computations confirm this limiting behavior of the Q-learning algorithm, validating its convergence properties.

Now we investigate the performance of Q-learning on historical data where the transition probabilities are unknown. On average, Q-learning outperforms the mean value problem (MVP) on the training data. For instance, in the months of June and July, the optimal objective achieved by MVP is 427 (see Figure 3.5). In contrast, the Q-learning algorithm with (3.30) state description, yields an average objective of 570 over the same period. This represents a substantial improvement of approximately 33%.

Figure 3.21 illustrates a case where Q-learning outperforms MVP. On June 22nd, a day within the training set, we observe a price trajectory with a significant spike at hour 13, which deviates notably from the average prices of June and July. The price surge occurs over two consecutive time periods, allowing Q-learning to capitalize on this spike more effectively than the MVP policy. Consequently, Q-learning achieves higher profits by responding to the sudden price increase. The MVP policy, relying on averaged values, fails to anticipate and react to such transient spikes effectively. This limitation is inherent to the static nature of MVP, which smooths out the peaks, thus missing out on potential profit opportunities that arise from short-term price volatility.

The computational results of the two methods are different on test data. We train Q-learning algorithm on month of June and July and test the optimal policy on days of August. Tables 3.4 and 3.5 present the result of testing Q-learning policy and MVP policy obtained from the historical data, June and July 2023, on the month of August 2023. In the tables $\overline{Q^*}$ indicates the empirical mean of optimal objectives achieved by applying Q-learning policy over all daily price trajectories of August. The average objective of applying MVP policy on the same data is 385.11.

From table 3.4 we see that when price intervals are considered in the state description, (3.28), Q-learning policy improves the arbitrage less than 3%. From table 3.5, it is evident that the most significant advantage of arbitrage, in the case of

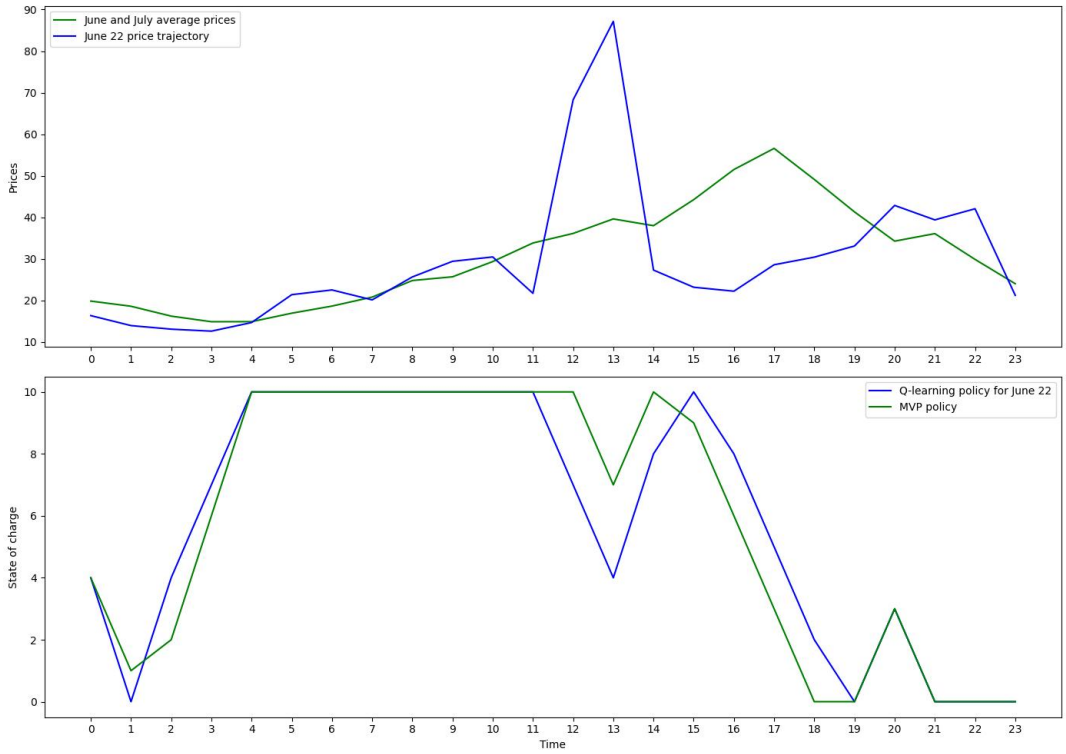

Figure 3.21: The Q-learning and MVP optimal policy tested on June 22nd. Q-learning optimal objective = 434.74, MVP optimal objective = 385.11.

considering price difference intervals, (3.30), is achieved with a 5% improvement when using the Q-learning policy compared to the MVP policy. In both cases, fine-tuning is necessary to select the optimal interval.

The primary reason Q-learning does not significantly enhance arbitrage on test data, is that price spikes are short-lived. For instance, consider the scenario where $P_{t-2} = 30$, $P_{t-1} = 80$ and $P_t = 30$. In this case, the Q-learning policy may not effectively capture and exploit the brief price surge at P_{t-1}, leading to suboptimal arbitrage performance. This limitation arises because the Q-learning algorithm relies on historical data to update its policy, and short-lived spikes do not provide enough consistent information to significantly adjust the learning process.

To further improve the Q-learning policy's effectiveness, advanced techniques such as adaptive learning rates, more sophisticated state representations, or integrating external predictive models for price forecasting could be employed. These enhancements could help the policy better respond to transient price changes and improve arbitrage outcomes.

Moreover, incorporating additional features into the state space, such as moving averages or volatility measures, might provide the Q-learning algorithm with more contextual information, allowing it to make more informed decisions during periods of rapid price fluctuations.

3.7 Conclusion

In this chapter, we analyze the quantitative performance of various optimization methods applied to our price models and historical data in both Markov and near-Markov settings in order to maximize energy arbitrage of a battery under price uncertainty. Our analysis demonstrates the hierarchy of optimal objectives achievable by the different optimization methods, reflecting their performance under varying information availability. Our findings can be summarized along a spectrum of information, ranging from zero information to perfect information about past and

Table 3.4: Comparing the average objective obtained by Q-learning optimal policy and MVP optimal policy over the month of August. At each time t a day trajectory is randomly chosen. The initial soc at each episode is also randomly chosen. MVP obj = 385.11

price intervals	$\overline{Q^*}$	relative gap
(0,50) [50, ∞)	388.42	0.85
(0,60) [60, ∞)	391.99	1.79
(0,70) [70, ∞)	386.96	0.49
(0,40) [40, 70) [70, ∞)	393.52	2.18

Table 3.5: Comparing the average objective obtained by Q-learning optimal policy and MVP optimal policy over the month of August. At each time t a day trajectory is randomly chosen. The initial soc at each episode is also randomly chosen. MVP obj = 385.11

price difference intervals	$\overline{Q^*}$	relative gap
($-\infty$, -3) [-3, 0) [0, 5) [5, ∞)	402.79	4.59
($-\infty$, -14) [-14, 14) [14, ∞)	390.92	1.50
($-\infty$, -15) [-15, 15) [15, ∞)	396.89	3.05
($-\infty$, -16) [-16, 16) [16, ∞)	394.38	2.40

future states, see Figure 3.22. The optimal objectives achieved by policies derived from methods MVP denoted by EV, recourse problem denoted by $RP(CEV)$ and $RP(LKAD)$, Q-learning indicated by Q_0^*, SDP represented by V_0^*, and the theoretical wait and see approach, WS, are ordered accordingly.

While the wait and see solution WS is theoretical and not achievable in real world, its value lies in providing a target for decision-making performance and understanding the potential benefits of improved information. On the other hand, the MVP is entirely price-independent and requires no information from the past or future. This characteristic makes MVP a useful benchmark for evaluating basic trends and the performance of other methods that incorporate varying degrees of information in their optimization processes. By comparing the outcomes of MVP with those of more information-dependent methods, we can assess how much additional value is derived from utilizing historical and predictive data. This comparison helps in understanding the trade-offs between complexity, information requirements, and performance gains. Selecting a policy will ultimately involve finding a balance between the quality of the solution and the computational complexity.

The availability of past and future information significantly impacts the optimal objectives and expected arbitrage opportunities of various optimization methods. To quantify the effects of past and future information, we introduce the following definitions:

$$\text{value of past information} = V_0^* - EV \tag{3.31}$$

$$\text{value of past and future information} = WS - EV \tag{3.32}$$

The value of past information helps in understanding the extent to which historical data can enhance the optimization performance. This measure highlights the benefits of learning from previous patterns, trends, and anomalies in price movements or other relevant metrics. Among the various optimization methods, SDP fully leverage past information. SDP's comprehensive use of historical data allows it to achieve

an upper bound on the optimal objective value while only using past information. Other optimization methods that only utilize past information tend to underperform when compared to SDP, indicating their relative inefficiency in capturing the full potential of historical data. If an optimization method outperform SDP, it means that it uses some future information in the optimization process. The two stage stochastic program with lookahead scenario generation ($RP(LKAD)$) is an example of such methods.

The value of past and future information provides insights into the maximum possible benefits that could be obtained if decision-makers had perfect information about future events. It sets an upper bound on the performance of any practical optimization method.

Another key conclusion from our empirical computational study is the clear necessity for advanced and sophisticated methods to improve expected arbitrage, surpassing the results achieved by MVP. Our findings indicate that there is a significant potential for enhancing arbitrage compared to MVP. In particular, high volatility and price spikes provide substantial arbitrage opportunity. However, Taking advantage of short-lived price spikes is challenging in an hourly setup. Any price spike occurring within an hour, by the time the spike is detected and a response is implemented, the spike may have already subsided, resulting in missed opportunities for optimization. By observing historical data, we find that another challenge is that price spikes occur randomly, typically between the hours of 10:00 and 19:00. This unpredictability makes it difficult to anticipate their occurrence and capitalize on them effectively.

To address these challenges and improve expected arbitrage, future research and development can focus on the following directions. Increasing temporal resolution i.e. moving from hourly setups to finer time intervals, such as 15-minute or 5-minute intervals, can capture and respond to short-lived price spikes more effectively. Developing and deploy advanced forecasting models that can predict short-term price spikes with higher accuracy can be another future direction. Advanced techniques in

machine learning and deep learning or other methods listed in literature review could be utilized to identify patterns and predict price movements more accurately.

It is essential that any attempt to design an algorithm for computing arbitrage be evaluated against the simple deterministic MVP, which produces a price-independent policy. This comparison is crucial for assessing the value of explicitly handling uncertainty. While some researchers have adopted this strategy, it is surprising how many studies neglect to develop an average case analysis for such evaluations. This oversight limits the ability to fully understand and demonstrate the benefits of more sophisticated approaches.

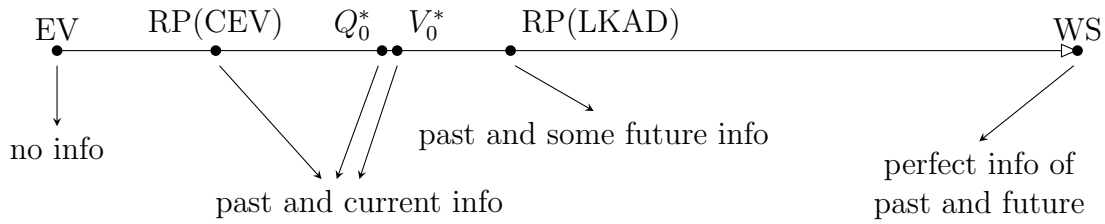

Figure 3.22: Information line and optimal objectives of various optimization models

Chapter 4

On linear programs with random coefficient matrix

The first section of this chapter is based on the following paper:

On the optimal objective value of random linear programs, Marzieh Bakhshi, James Ostrowski and Konstantin Tikhomirov (2024). Submitted.

A preprint of this paper is available at https://arxiv.org/pdf/2401.17530

The second section is based on the following paper:

A decomposition algorithm for two-stage stochastic programs with approximate rotational invariance, Marzieh Bakhshi and Konstantin Tikhomirov.

A preprint of this paper is available at https://arxiv.org/pdf/2407.21215

4.1 Introduction

In this chapter, we explore linear programs with a random coefficient matrix for two main purposes. First, we aim to estimate the optimal objective of these programs. Second, we examine a decoupling algorithm for two-stage stochastic programs where the technology matrix is randomly generated.

Linear programs with either the coefficient matrix or the right hand side generated randomly, have been actively studied in various contexts:

- *Average-case analysis.* As we mentioned above, the setting of random LPs with the Gaussian coefficient matrix (or, more generally, a matrix with independent rows having spherically symmetric distributions) was considered by Borgwardt (Borgwardt, 1982, 1987, 1999) in the context of estimating the average-case complexity of the shadow simplex method. The average-case complexity of the Dantzig algorithm was addressed by Smale in (Smale, 1983) (see also (Blair, 1986)).

- *Smoothed analysis.* The smoothed analysis of the simplex algorithm, when the coefficient matrix is a sum of an arbitrary deterministic matrix and a small Gaussian perturbation, was first carried out by Spielman and Teng in (Spielman and Teng, 2004), with improved estimates obtained in later works (Vershynin, 2009; Dadush and Huiberts, 2018; Huiberts et al., 2023).

- *Distribution of the optimal objective value under small random perturbations.* The setting when the coefficient matrix, the RHS or the cost vector is subject to small random perturbations was considered, in particular, in (Babbar, 1955; Prékopa, 1966; Klatt et al., 2022; Liu et al., 2023).

- *Random integer feasibility problems.* We refer to (Chandrasekaran and Vempala, 2014) for the problem of integer feasibility in the randomized setting.

- *The optimal objective value of LPs with random cost vectors (Dyer et al., 1986).*

4.2 Asymptotics of the optimal objective

4.2.1 Introduction

Consider the linear program (LP)

$$\begin{aligned} &\text{maximize} \quad \langle c, x \rangle \\ &\text{subject to} \quad Ax \leq \mathbf{1} \\ &x \in \mathbb{R}^n \end{aligned} \tag{4.1}$$

where the entries of the coefficient matrix A are mutually independent random variables, and $\mathbf{1}$ denotes the vector of all ones. Note that x is a vector of free variables. In this note we deal with two settings: either c is a non-random unit vector or c is random uniformly distributed on the unit Euclidean sphere. For any realization of A the above LP is feasible since the zero vector satisfies all the constraints. We denote the value of the objective function of (4.1) by $z = z(x) = \langle c, x \rangle$, the optimal objective value by z^*, and any optimal solution by x^*.

The main problem we would like to address here is the following: *Under the assumption that the number of constraints is significantly larger than the number of variables, what is the magnitude of the optimal objective value?*

The LP (4.1) with i.i.d random coefficients can be viewed as the simplest model of an "average-case" linear program. LP of this form has been considered in the mathematical literature in the context of estimating the average-case complexity of the simplex method (Borgwardt, 1982, 1987, 1999). Despite significant progress in that direction, fundamental questions regarding properties of random LPs remain open (Spielman and Teng, 2004, Section 6). The results obtained in this note contribute to better understanding of the geometry of random feasible regions defined as intersections of independent random half-spaces. We view our work as a step towards investigating random LPs allowing for sparse, inhomogeneous and correlated

constraints, which would serve as more realistic models of linear programs occurring in practice.

The problem of constructing or approximating a vector in the non-empty intersection of a given collection of convex sets has been actively studied in optimization literature. The special setting where the convex sets are affine hyperplanes or halfspaces, corresponding to solving systems of linear equations and inequalities, was first addressed by Kaczmarz, Agmon (Agmon, 1954), Motzkin and Schoenberg (Motzkin and Schoenberg, 1954). We refer, among others, to papers (Aharoni and Censor, 1989; Chubanov, 2015; Gower and Richtárik, 2015; Liu and Wright, 2016; Jiao et al., 2017; De Loera et al., 2017; Bai and Wu, 2018; Gower et al., 2021; Morshed et al., 2022) for more recent developments and further references. In particular, the work (Aharoni and Censor, 1989) introduced the block-iterative projection methods for solving the feasibility problems, when each update is computed as a function of projections onto a subset of constraints.

4.2.2 Theoretical and computational results

We state the following theorem as the main result of the study of the linear program (4.1) with Gaussian and subGaussian random matrices. Examples of subGaussian random variables include Rademacher (i.e symmetric ± 1) random variables, and more generally, any bounded random variables. The Rademacher distribution is of interest to us since a matrix with i.i.d ± 1 entries produces a simplest model of a normalized system of constraints with discrete bounded coefficients. For the proof of the theorem we refer the reader to (Bakhshi et al., 2024).

Theorem 4.1 (Asymptotics of z^* in (sub)Gaussian setting)**.** *Assume that* $\lim_{n\to\infty} \frac{m}{n} = \infty$ *and that for each* n*, the entries of the* $m \times n$ *coefficient matrix* $A = A(n)$ *are mutually independent standard (sub)Gaussian variables. For each* n*, let* $c(n)$ *be a non-random unit cost vector. Then* $\lim_{n\to\infty} \sqrt{2\log(m/n)}\, z^* = 1$ *almost surely.*

The above result naturally give rise to the questions: (a) *How close to each other are the asymptotic estimates of the optimal objective value and empirical observations?*, (b) *What is the asymptotic distribution and standard deviation of z^*?*, and (c) *How does the algorithm in the proof of Theorem 4.1 perform in practice?* We discuss these questions in the following, and make conjectures while providing numerical evidence.

Magnitude of the optimal objective value: We investigate the quality of approximation of the optimal objective value z^* by the asymptotic bound $ab := (2\log(m/n))^{-\frac{1}{2}}$ given in Theorem and 4.1. The empirical outcomes are obtained through multiple runs of the LP (4.1) under various choices of parameters. We consider two distributions of the entries of A: either A is standard Gaussian or its entries are i.i.d Rademacher (± 1) random variables. In either case and for different choices of m, we sample the random LP (4.1) taking the sample size 50 and letting c be the vector of i.i.d Rademacher variables rescaled to have unit Euclidean norm. The cost vector is generated once and remains the same for each of the 50 instances of the LP within a sample.

As is shown in Tables 4.1 and 4.2, for a fixed value of $n = 50$, as the number of constraints m progressively increases in magnitude, the *relative gap* between ab and the sample mean $\hat{\mu}$ of the optimal objective values, defined as

$$\left(\frac{|ab - \hat{\mu}|}{ab} \times 100\right)\%,$$

decreases.

Limiting distribution of the optimal objective value: Recall that, given mutually independent standard Gaussian variables, their maximum converges (up to appropriate rescaling and translation) to the Gumbel distribution as the number of variables tends to infinity. If the vertices of the random polyhedron $P = \{x \in \mathbb{R}^n : \ Ax \leq \mathbf{1}\}$ were behaving like independent Gaussian vectors (with a proper rescaling) then the limiting distribution of z^* for any given fixed cost vector would

Table 4.1: Asymptotic versus empirically observed optimal objective value in the Gaussian setting (the sample size = 50)

m	n	ab	$\hat{\mu}$	relative gap(%)
1000	50	0.40853	0.50626	23.92
2000	50	0.36816	0.44119	19.83
6000	50	0.32317	0.37256	15.28
10000	50	0.30719	0.35176	14.50
20000	50	0.28888	0.32473	12.41

Table 4.2: Asymptotic versus empirically observed optimal objective value in the Rademacher setting (the sample size = 50)

m	n	ab	$\hat{\mu}$	relative gap(%)
1000	50	0.40853	0.50369	23.29
2000	50	0.36816	0.43801	18.97
6000	50	0.32317	0.37200	15.11
10000	50	0.30719	0.35220	14.65
20000	50	0.28888	0.32856	13.73

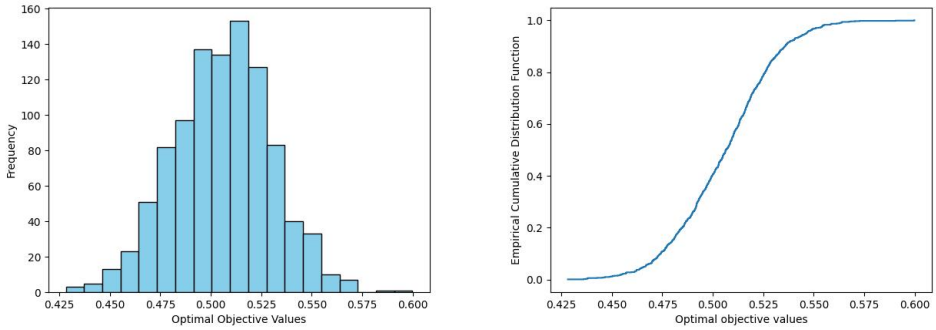

Figure 4.3: The frequency histogram and the empirical cumulative distribution function of the optimal objective values for the Gaussian matrix A with $m = 1000$, $n = 50$ (the sample size = 1000). The KS test results: KS statistic = 0.0232, p-value = 0.6453.

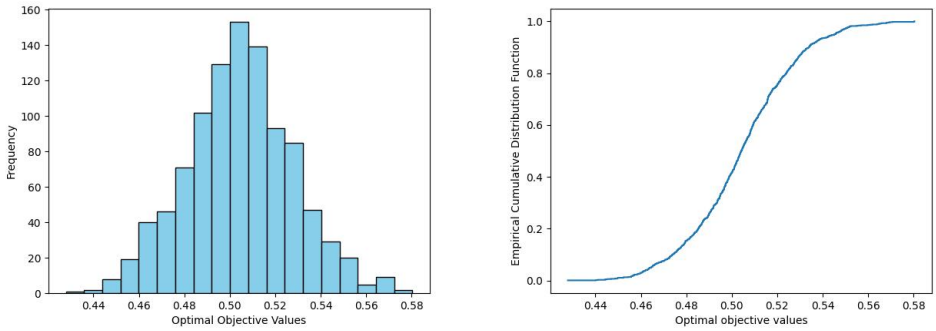

Figure 4.6: The frequency histogram and the empirical cumulative distribution function of the optimal objective values for the Rademacher matrix A with $m = 1000$, $n = 50$ (the sample size = 1000). The KS test results: KS statistic = 0.0219, p-value = 0.7161.

be asymptotically Gumbel. Our computations suggest that in fact the limiting distribution is Gaussian.

In our numerical experiments, we consider Gaussian and Rademacher random coefficient matrices of dimensions $m = 1000$ and $n = 50$. The sample size in either case is taken to be 1000. As in the previous experiment, we take c to be a random vector with i.i.d Rademacher components rescaled to have the Euclidean norm one. We generate the cost vector once so that it is the same for every LP from a sample. We employ the Kolmogorov–Smirnov (KS) test to compare the sample distribution of z^* to Gaussian of a same mean and variance. In either case, the obtained p-value exceeds the conventional significance level of 0.05, indicating lack of substantial evidence to refute the null hypothesis that the data is normally distributed. Further, visual representation of the frequency histogram and the empirical cumulative distribution function of the optimal values shown in Figures 4.3 and 4.6 validate this conjecture.

The next conjecture summarizes the experimental findings:

Conjecture 4.2 (Limiting distribution of z^*)**.** *Let m, n, $A(n)$ be as in Theorem 4.1, and for each n let $c = c(n)$ be uniform random unit cost vector independent from $A(n)$. Then the sequence of normalized random variables*

$$\frac{z^* - \mathbb{E}\, z^*}{\sqrt{\operatorname{Var} z^*}}$$

converges in distribution to the standard Gaussian.

Standard deviation of the optimal objective value: As in the previous two numerical experiments, in this one we consider two types of the entries distributions: Gaussian and Rademacher. The cost vector is generated according to the same procedure as above. We sample the LP (4.1) taking the sample size to be 50, and compute the sample standard deviation $\hat{\sigma}$ (the square root of the sample variance) of the optimal objective value. Based on our numerical studies we speculate that

Table 4.3: Sample standard deviation of the optimal objective value with Gaussian matrix A (the sample size = 50)

m	n	ab	$\hat{\sigma}$	$\hat{\sigma} \times \sqrt{m}$
1000	50	0.408539	0.02420	0.765
2000	50	0.368161	0.01728	0.773
4000	50	0.337791	0.01333	0.843
6000	50	0.323170	0.01142	0.885
8000	50	0.313877	0.01059	0.947
10000	50	0.307196	0.01027	1.027
1000	100	0.465991	0.03014	0.953
2000	100	0.408539	0.01592	0.712
4000	100	0.368161	0.01172	0.742
6000	100	0.349456	0.00990	0.767
1000	150	0.528257	0.03192	1.010
2000	150	0.441515	0.015066	0.674
4000	150	0.391745	0.01155	0.731
6000	150	0.368161	0.011850	0.918

Table 4.4: The sample standard deviation of the optimal objective value with Rademacher matrix A (the sample size = 50)

m	n	ab	$\hat{\sigma}$	$\hat{\sigma} \times \sqrt{m}$
1000	50	0.408539	0.020420	0.645
2000	50	0.368161	0.017393	0.777
4000	50	0.337791	0.011962	0.756
6000	50	0.323170	0.010207	0.790
8000	50	0.313877	0.010333	0.924
10000	50	0.307196	0.008997	0.899
1000	100	0.465991	0.023097	0.730
2000	100	0.408539	0.014648	0.655
4000	100	0.368161	0.010436	0.660
6000	100	0.349456	0.011791	0.913
1000	150	0.528257	0.032600	1.030
2000	150	0.441515	0.015466	0.691
4000	150	0.391745	0.011030	0.697
6000	150	0.368161	0.009153	0.708

standard deviation of z^* is roughly of the order $1/\sqrt{m}$, at least in the setting where m is very large compared to n. Indeed, upon examination of the last column in Tables 4.3 and 4.4, we observe a consistent phenomenon: the quantities are of order 1 for various choices of m and n.

An algorithm for finding a near-optimal feasible solution: As previously discussed, the proof of Theorem 4.1 involves an iterative projection process to restore feasibility from an initial point. In the process, we discretize the continuous space spanned by violated constraints at each iteration (i.e use a covering argument) as a means to apply probabilistic union bound. Recall that the goal of the algorithm described in the proof of Theorem 4.1 is to provide theoretical guarantees for z^*. An exact software implementation of that algorithm is heavy in terms of both the amount of code and computations. On the other hand, it is of considerable interest to measure performance of block-iterative projection methods in the context of finding near optimal solution to instances of the random LP (4.1). To approach this task, we consider a 'light' version of the algorithm from the proof of Theorem 4.1, which does not involve any covering arguments (see Algorithm 5 below). The algorithm is a version of the classical Kaczmarz block-iterative projection method as presented in (Elfving, 1980) and its randomized counterparts in (Needell and Tropp, 2014) and (Zhang and Li, 2023). In our context, the blocks correspond to the rows of the matrix A that are violated by the initial point and its updates at each iteration. Table 4.5 shows the results of running the algorithm for various parameters m and n. For each instance of the LP, r is the the number of iterations required to restore feasibility, $x_0 = (2\log(m/n))^{-\frac{1}{2}}c$ is the initial point and x is the output of the Algorithm 5. Note that the obtained perturbations to x_0 are in the orthogonal complement of the cost vector c and as a result $\|x_0\|_2 = z(x)$. The numbers $|I_0|$ and $|I_1|$ are the number of violated constraints by x_0 at iteration 1 and $x_0 + x_1$ at iteration 2, respectively. Note that these numbers are rather small compared to the size of the LP instance. We note that in our experiment, the algorithm did not converge for the 20000×50

instance of the LP, which we attribute to the random nature of the problem. At the same time, the Algorithm converged fast due to small number of constraint violations for all other given instances.

Algorithm 4: Algorithm for finding near-optimal feasible solution to LP (4.1), an instance of block-iterative Kaczmarz projection method

Input: $A_{m\times n}$: standard Gaussian random matrix,

c: random cost vector uniformly distributed on the sphere

Output: x: near-optimal feasible solution to LP (4.1)

$x \leftarrow (2\log(m/n))^{-\frac{1}{2}}c$;

$j \leftarrow 0$;

$\epsilon \leftarrow 0.1$;

while x *is not feasible* **do**

  $j \leftarrow j+1$;

  Find the set of constraints I_{j-1} that are violated or nearly violated by x, i.e all indices i with $\langle \text{row}_i(A), x\rangle > 1-\epsilon$;

  Compute vector b where $b_i = 1-\epsilon-\langle \text{row}_i(A), x\rangle$ for all $i \in I_{j-1}$;

  Project the constraints into the space $c^{\perp}$ (i.e define $v_i = \text{row}_i(A) - \langle \text{row}_i(A), c\rangle c$ for all $i \in I_{j-1}$) ;

  Find x_j as a linear combination of v_i's, $i \in I_{j-1}$, so that the constraints violated by x are repaired (specifically, solve the linear system $M^TMu = b$ for u, then solve $x_j = Mu$ where M is the matrix with columns v_i);

  $\epsilon \leftarrow \epsilon * 0.1$;

  $x \leftarrow x + x_j$;

end

Return x

Table 4.5: Numerical results for the Algorithm 5 for various m and n (r = the number of iterations to restore the feasibility).

m	n	r	$\|x_0\|_2 = z(x)$	$\|I_0\|$	$\|I_1\|$
1000	50	2	0.408539	11	1
1000	100	2	0.465991	28	4
2000	50	2	0.368161	6	1
2000	100	2	0.408539	23	1
4000	50	2	0.337791	13	2
4000	100	2	0.368161	30	1
6000	50	2	0.323170	22	21
6000	100	2	0.349456	31	8
8000	50	2	0.313877	22	6
8000	100	2	0.337791	21	5
10000	50	2	0.307196	19	1
10000	100	1	0.329505	34	0
20000	100	1	0.307196	29	0
40000	50	1	0.273493	12	0
40000	100	2	0.288881	34	3
60000	50	2	0.265558	16	2
60000	100	1	0.279576	36	0
80000	50	1	0.260329	21	0
80000	100	2	0.273493	36	1
100000	50	2	0.256479	28	5
100000	100	2	0.269040	35	1

4.2.3 Conclusion

In this section, we study the optimal objective value z^* of a random linear program where the entries of the $m \times n$ coefficient matrix A have subgaussian distributions, the variables $x_1, \ldots, x_n$ are free and the right hand side of each constraint is 1. We establish sharp asymptotics $z^* = (1 \pm o(1))(2\log(m/n))^{-\frac{1}{2}}$ in the regime $n \to \infty$, $\frac{m}{n} \to \infty$ and some additional assumptions on the cost vectors. This asymptotic provides quantitative information about the geometry of the random polyhedron defined by the feasible region of the random LP (4.1), specifically about its spherical mean width. The computational experiments support our theoretical guarantees and give insights into several open questions regarding the asymptotic behaviour of the random LP. The connection between the proof of the main result and convex feasibility problems allows us to provide a fast algorithm for obtaining a near optimal solution in our randomized setting. The random LP (4.1) is a step towards studying models with inhomogeneous coefficient matrices and establishing stronger connections to practical applications.

4.3 Decoupling two stage stochastic programs

4.3.1 Introduction

Consider the general formulation of two stage stochastic program

$$\begin{aligned} &\text{maximize} \quad \langle c, x \rangle + \mathbb{E}_{\boldsymbol{\xi}}[Q(x, \boldsymbol{\xi})] \\ &\text{subject to} \quad Ax \leq b \\ &x \geq 0 \end{aligned} \tag{4.2}$$

where for each realization ξ of random vector $\boldsymbol{\xi}$, $Q(x, \xi)$ is the optimal objective value of the recourse problem:

$$\begin{aligned} &\text{maximize} \quad \langle q(\xi), y \rangle \\ &\text{subject to} \quad T(\xi)x + W(\xi)y \leq h(\xi) \\ &y \geq 0. \end{aligned} \tag{4.3}$$

In program (4.2), $c, x \in \mathbb{R}^{n_1}$ are the first stage cost and variable vectors, $A \in \mathbb{R}^{m_1 \times n_1}$ encodes first stage constraints, and $b \in \mathbb{R}^{m_1}$ denotes the right hand side of the constraints. In the recourse problem (4.3), $q(\xi), y(\xi) \in \mathbb{R}^{n_2}$ are the second stage cost and variable vectors, and the technology matrix $T(\xi) \in \mathbb{R}^{m_2 \times n_1}$ along with the recourse matrix $W(\xi) \in \mathbb{R}^{m_2 \times n_2}$ determine the quantitative relationship between the first and the second stage variables. The vector $h(\xi) \in \mathbb{R}^{m_2}$ denotes the right hand side of the second stage constraints.

Combining the recourse problem together with the first stage constraints, and considering all realizations of $\boldsymbol{\xi}$, we obtain the so-called extensive form or deterministic equivalent of problem (4.2):

$$\begin{aligned} &\text{maximize } \langle c, x \rangle + \sum_{\xi} \mathbb{P}(\xi) \langle q(\xi), y(\xi) \rangle \\ &\text{subject to } Ax \leq b \\ &\qquad\qquad T(\xi)x + W(\xi)y(\xi) \leq h(\xi) \\ &\qquad\qquad x \geq 0 \\ &\qquad\qquad y(\xi) \geq 0. \end{aligned} \tag{4.4}$$

The two-stage problem (4.2) can be approached by considering the equivalent deterministic problem (4.4) directly using optimization solvers. However, problem (4.4) can be computationally expensive when the number of scenarios (i.e distinct realizations of $\boldsymbol{\xi}$) is large.

To resolve the issue of computational inefficiency, various decomposition algorithms have been proposed and employed in practice. The L-shaped method (Van Slyke and Wets, 1969) is the most widely used decomposition algorithm for solving two stage stochastic programs. This method employs Benders decomposition (Benders, 1962; Laporte and Louveaux, 1993) to address the problem. The algorithm starts by defining a master problem with only the first-stage decision variables. After the master problem is solved, the first-stage variables are fixed at an optimal solution of the master problem. Then the second stage problem can be decomposed into $|\mathcal{S}|$ subproblems where $\mathcal{S}$ denotes the set of scenarios. The subproblems can be solved in parallel, and valid inequalities can be derived and added to master problem. The master problem is then solved again and the algorithm iterates. The algorithm is known to converge in a finite number of steps when the second-stage decision variables are continuous and the number of scenarios is finite.

A related decomposition method is Lagrangian method (Guignard, 2003) where non-anticipativity constraints are typically enforced through the use of Lagrangian multipliers (dual variables) associated with these constraints. The Lagrangian relaxation of the problem incorporates these constraints along with the original objective function and other constraints. The dual variables associated with the non-anticipativity constraints capture the price of violating these constraints, and they are updated iteratively as part of the decomposition process.

Along the line of research related to Benders decomposition, a substantial body of literature has emerged, presenting a variety of decomposition techniques tailored for two stage stochastic programs. In particular, one can mention stochastic decomposition (Higle and Sen, 1991), subgradient decomposition (Sen, 1993), level decomposition (Zverovich et al., 2012), and partition-based algorithms (Song and Luedtke, 2015; van Ackooij et al., 2018).

Existing decomposition algorithms are typically designed for generic technology matrices, which define the relationship between first and second stage variables.

However, what if the technology matrix possesses a unique structure? Can we exploit this structure for decoupling the stochastic programs? The answer is yes when dealing with Gaussian technology matrices.

In the present work, we specifically focus on two stage stochastic programs with standard Gaussian technology matrices. This focus allows us to harness the unique properties of Gaussian matrices to develop a decomposition algorithm. Specifically, we introduce a method that uses the rotational invariance of standard Gaussian matrices. This characteristic enables us to decouple the first and second stages of the stochastic programs.

Problem setup: We propose a decomposition algorithm for solving the stochastic program (4.2) under the following assumptions.

- (Independence) The random matrices and vectors $h(\xi), q(\xi), T(\xi), W(\xi)$ are mutually independent;
- (Centered Gaussian) Each row of $T(\xi)$ has i.i.d Gaussian components with mean 0 (arbitrary variance)
- (Feasibility and boundedness) There is $R > 0$ such that for every realization of ξ and for every choice of vector $x \in \mathbb{R}^n$ with $\|x\| \leq \max\{\|x'\| : \ Ax' \leq b, \ x' \geq 0\}$, the recourse program is feasible, and the optimal objective value of the program lies in the interval $[-R, R]$;

Suppose that our goal is to estimate the optimal objective value of the program (4.2) up to a small additive error, i.e to compute a number $\hat{z}^*$ such that

$$\hat{z}^* - z^* \approx 0.$$

The main purpose of this note is to show that in the setting when the distribution of the entries of the technology matrix is close to Gaussian (normal), and certain additional technical conditions are satisfied, the estimation can be performed by *decoupling* the first and second stage of the program into separate computations.

Remark. The algorithm described below is based on the following observation. If x_1 and x_2 are any pair of fixed vectors in $\mathbb{R}^{n_1}$ having the same Euclidean norm, then the sets

$$\{T(\xi)x_1\}_\xi \text{ and } \{T(\xi)x_2\}_\xi$$

are very similar (in fact equi-distributed). This suggests that the expected objective value of the recourse problem is influenced essentially by the norm of vector x and not by its direction. Therefore, we can "replace" the first stage solution x^* in the recourse problem by the first standard basis vector times the norm $||x^*||_2$, without significantly altering the optimal objective of the entire two stage program.

4.3.2 The decoupling algorithm

The decoupling procedure for the first and second stages involves the following steps.

1. *Discretizing the norm of the first stage vector x:* We begin by discretizing the norm of the first stage decision vector x. Specifically, we consider the constraint $||x||_2 \leq \tau$ together with other first stage constraints, where $\tau = k\Delta$, for some discretization step size Δ and integer k. This constraint ensures that the decision space is manageable and structured for optimization.

$$\begin{aligned} &\text{maximize} \quad \langle c, x \rangle \\ &\text{subject to} \quad Ax \leq b \\ &||x||_2 \leq \tau \\ &x \in \mathbb{R}^n \end{aligned} \tag{4.5}$$

2. *Solving the first stage problem:* With the constraint $||x||_2 \leq \tau$ where $\tau = k\Delta$, we solve the first stage problem above to find an optimal solution x^*. This step involves optimizing a convex nonlinear program.

3. *Fixing the first stage variable in the recourse problem:* In the recourse problem, we fix the variable x to the optimal solution x^* obtained from the first stage. This step decouples the first stage and second stage variables in the recourse problem.

$$\begin{aligned} &\text{maximize} \quad \mathbb{E}_\xi \left[\langle\langle q(\xi), y(\xi)\rangle\right] \\ &\text{subject to} \quad W(\xi)y(\xi) \leq h(\xi) - T(\xi)x^* \\ &y \in \mathbb{R}^m \end{aligned} \tag{4.6}$$

4. *Reformulating the recourse problem:* The recourse problem now has $T(\xi)x^*$ on the right hand side, where $T(\xi)$ is a random matrix and x^* is a fixed vector. Given the assumption that T is a standard Gaussian matrix, $T(\xi)x^*$ becomes a random vector which is a multiple ($||x^*||_2$) of standard Gaussian distribution. Since standard Gaussian distribution is rotational invariant, $T(\xi)x^* \overset{d}{\sim} T(\xi)\overline{x}$ where $\overline{x}$ is any vector with the condition that $||\overline{x}||_2 = ||x^*||_2$. For simplicity of the computation we replace x^* by the extreme vector $\overline{x} = (||x^*||_2, 0, 0, \ldots, 0)$. As a result problem (4.6) reads as follows

$$\begin{aligned} &\text{maximize} \quad \mathbb{E}_\xi \left[\langle q(\xi), y(\xi)\rangle\right] \\ &\text{subject to} \quad W(\xi)y(\xi) \leq h(\xi) - \text{col}_1(T(\xi))||x^*||_2 \\ &y \in \mathbb{R}^m \end{aligned} \tag{4.7}$$

where $\text{col}_1(T(\xi))$ is the first column of the matrix $T(\xi)$.

The decoupling procedure is presented in Algorithm 5. In this algorithm, K is an upper bound for k which is initialized with a sufficiently large number and is updated at the beginning of the algorithm according to the norm of optimal objectives of (**??**). Further, S is the set of scenarios, Z_1 and Z_2 are vectors for storing first and second stage optimal objectives, respectively, and X is a vector indexed over $[0, \ldots, K]$ that stores the norms of $\tilde{x}_{\Delta k}$, for each admissible k.

Algorithm 5: The decoupling algorithm based on the structure of technology matrix

Input: A, W, T, b, h, c, q; Δ (a discretization step); K (a sufficiently large parameter)

Output: An approximation to the optimal objective value of (4.2)

for $k \leftarrow 0$ *to* K **do**
  Solve problem (4.5) with $\tau := \Delta\, k$;
  $Z_1[k] \leftarrow$ optimal objective value of (4.5);
  $X[k] \leftarrow$ norm of the optimal solution of (4.5);
end
$a \leftarrow \max_k X[k]$;
$K \leftarrow a/\Delta + 2$ (updating the upper bound K to optimize the execution time);
for $s \leftarrow 0$ *to* S **do**
  for $k \leftarrow 0$ *to* K **do**
    Compute the RHS of (4.7) taking the value of $\|\tilde{x}_{\Delta\, k}\|_2$ from $X[k]$;
    Solve problem (4.7) ;
    $Z_2[s, k] \leftarrow$ optimal objective value of (4.7);
  end
end
Compute $\mathbb{E}_{\boldsymbol{\xi}} Z_2[\boldsymbol{\xi}, k]$ for every k;
for $k \leftarrow 0$ *to* K **do**
  $Z[k] \leftarrow \mathbb{E}_{\boldsymbol{\xi}} Z_2[\boldsymbol{\xi}, k] + Z_1[k]$;
end
Return $\max_k Z(k)$;

4.3.3 Computational Studies

To evaluate the performance of the proposed algorithm, we conduct a series of tests with various sizes of matrices A, T and W, and compare it with both the extensive form and Benders' decomposition. Recall that the extensive form combines all possible scenarios into a single large scale linear problem. The optimal objective of this problem serves as a benchmark to measure the accuracy of our algorithm. Benders' decomposition, a widely used decomposition algorithm for two stage stochastic programs, is implemented as a second benchmark. We evaluate the Benders decomposition in terms of computational time. We solve the extensive form and the subproblems of Benders' decomposition and our decoupling algorithm with help of the optimization solver Gurobi 10.1.1.

The third benchmark, which we refer to as the *naive decoupling*, consists in fixing the first stage variables to the optimal solution of problem

$$\begin{aligned} &\text{maximize } \langle c, x\rangle \\ &\text{subject to } Ax \leq b \\ &\qquad x \geq 0, \end{aligned} \tag{4.8}$$

denoted as $\tilde{x}$, and solving the recourse problems for that choice of the first-stage variables:

$$\begin{aligned} &\text{maximize } \langle c, \tilde{x}\rangle + \sum_{\xi \in \mathcal{S}} \mathbb{P}(\xi)\langle q(\xi), y(\xi)\rangle \\ &\text{subject to } \; T(\xi)\tilde{x} + W(\xi)y(\xi) \leq h(\xi) \\ &\qquad y(\xi) \geq 0. \end{aligned} \tag{4.9}$$

By comparing the optimal objective value of (4.9) with that of the extensive form, we gain insights into the parameter configurations where our algorithm proves to be effective.

These metrics are useful for assessing the relative efficiency of our proposed algorithm. The computational results are presented in Table 4.6. In the table, m_1 and n_1 are the number of rows and columns of the matrix A, and m_2 and n_2 are the number of rows and columns of the matrix W. We consider constant vectors h and vary the magnitude of the components for different choices of the dimension of the matrices. *Ngap* is the relative gap in percentage between the optimal objective of problem (4.9) and the optimal objective of the extensive form, while *Dgap* is the relative gap in percentage between the optimal objective obtained by our algorithm and the optimal objective of the extensive form. Columns t_e, t_d and t_b denote the computational times for solving the extensive form, executing our algorithm and Benders decomposition algorithm respectively.

Input data for the experiments in Table 4.6 are as follows:

- $b = \mathbf{1}$, $\Delta = 0.01$, $K = 100$
- A is a realization of a standard Gaussian matrix. The realization is fixed and remains the same in all experiments.
- c, q are realizations of Gaussian random vectors normalized to have $\|c\|_2 = 0.5$ and $\|q\|_2 = 1$. The realizations are fixed and remain the same in all experiments.
- $T(\xi)$, $W(\xi)$ are realizations of independent standard Gaussian matrices.
- Number of scenarios = sample size = 50
- Each number in the 6th to 10th column of the table is the empirical average over 50 runs for the given parameters.
- Compared to the description in Algorithm 5, the range of k in the main cycle for the second stage problem is optimized to reduce the computational time.

From the table, we observe that Dgap is of order 2%, indicating that the optimal objective of our decoupling algorithm is in close proximity to that of the extensive form solution. We must note that in certain parameter settings, the approximation accuracy of our algorithm matches that of the naive decoupling. For example, when the number of the first stage variables is small, both the naive approach and our decoupling algorithm yield similar objective values. Determining the range of parameters where our algorithm significantly outperforms the naive decoupling remains an open question.

Although solving the extensive form exhibits better performance in terms of the computational time, our decoupling algorithm outperforms Benders' decomposition for some choice of parameters. For fair comparison we run Benders' decomposition algorithm up to 2% gap between the objective obtained from the master problem and the objective of current incumbent.

Table 4.6: Performance of our algorithm compared with extensive form, Benders' decomposition, and naive decoupling. Columns t_e, t_d and t_b are computational times for solving the extensive form, our decoupled model and Benders decomposition algorithm respectively.

m_1	n_1	m_2	n_2	h	Ngap	Dgap	t_e	t_d	t_b
100	5	100	5	2	8.1	2.1	0.05	0.41	0.41
100	10	100	10	2	11.7	2.3	0.18	0.75	2.1
100	15	100	15	2	34.3	2.1	0.2	1.2	5.18
100	20	100	20	2	20.7	2.8	0.45	1.6	10.4
100	5	100	5	3	3.3	1.8	0.06	0.43	0.24
100	10	100	10	3	5.6	2.0	0.17	0.78	1.1
100	15	100	15	3	13.6	2.1	0.21	1.3	3.2
100	20	100	20	3	9.3	2.5	0.45	1.7	6.1
100	5	100	5	4	2.1	2.1	0.06	0.46	0.2
100	10	100	10	4	3.3	2.0	0.16	0.73	0.8
100	15	100	15	4	7.7	2.2	0.22	1.2	1.8
100	20	100	20	4	5.6	2.1	0.41	1.6	2.4
100	5	100	5	5	1.3	2.1	0.06	0.48	0.2
100	10	100	10	5	2.2	1.9	0.17	0.76	0.6
100	15	100	15	5	5.4	2.2	0.2	1.1	1.1
100	20	100	20	5	4.1	2.8	0.4	1.5	1.5

4.3.4 Conclusion

In this section, we exploit the structure of the technology matrix to effectively decouple the first and second stages of two stage stochastic programs. Through empirical analysis, we demonstrate that when the technology matrix follows a random standard Gaussian distribution, it is possible to decouple the two stage stochastic program efficiently. The optimal objectives of our proposed algorithm are in close proximity to the optimal objectives of the extensive form solution. Further studies could involve testing the algorithm on larger problem instances. The question of choosing parameters when our decoupling algorithm significantly outperforms the naive decoupling remains open. We hope that our findings will inspire the discovery of additional decomposition techniques leveraging the structure of the technology matrix.

Chapter 5

Conclusions and future work

In this chapter we draw conclusions about our work in previous chapters and suggest directions for further research.

5.1 Optimization of a battery energy storage for arbitrage

Chapter 2 reformulates the battery charging problem in a deterministic setting as a shortest path problem, resulting in a polynomial-time algorithm for its solution. By assuming that storage and round-trip efficiencies are negligible, we analyzed different types of vertices within the convex hull of the feasible region, and conducted optimization across all subintervals of the entire time horizon.

While this approach is innovative, the computational results have not been as promising as those achieved by solving Mixed-Integer Linear Programming (MILP) problems using commercial solvers.

Future research could focus on generating cuts in the direction of the objective function to enhance the performance of our algorithm. This could potentially improve computational time efficiency. The optimization process for each subinterval can be performed independently, suggesting that the shortest path calculations can be

parallelized. Implementing parallel processing techniques could significantly reduce computation time and improve overall performance.

Chapter 3 addresses the optimization of battery charging strategies under price uncertainty. Various optimization methods have been employed to develop operational policies aimed at maximizing expected arbitrage. However, the challenge of designing a policy that significantly improves energy arbitrage compared to the MVP remains open. Additionally, when applying real historical data and using model-free approximation methods in the absence of known price dynamics, the construction of an effective state space continues to pose a significant challenge.

Future research could focus on exploring more advanced optimization techniques beyond those currently used. This could include approaches such as function approximation in deep reinforcement learning, evolutionary algorithms, or hybrid methods that combine different optimization methods. Creativity can play a role at every stage of the optimization process including state construction, reward function definition, dimentionality reduction, etc.

5.2 Random coefficient matrices in optimization problems

Chapter 4 explores the potential applications of random matrices in optimization problems. Linear programs with random coefficient matrices are employed in average-case analysis of algorithms such as the simplex method. By examining the expected performance of these algorithms, we can gain valuable insights into their behavior in practice as apposed to in theory. In our model, we consider linear programs with variables that are free, without any constraints. Future research could explore the case where variables are non-negative. Additionally, while our current analysis focuses on cases with tall coefficient matrices, it would be interesting to investigate situations where the coefficient matrix is square or nearly square.

Two-stage stochastic programs can be effectively decoupled by exploiting the structure of the technology matrix. This study demonstrates that, under the assumption that the technology matrix follows a standard Gaussian distribution, the first and second stages of the problem can be decomposed efficiently.

Future research should investigate the potential applications of problems involving technology matrices with standard Gaussian distributions. This exploration could reveal additional insights into how such matrices can facilitate more effective decompositions in practice. Furthermore, exploring other possible structures of technology matrices could uncover new opportunities for decomposition of stochastic program.

Bibliography

Aghahosseini, A., Bogdanov, D., and Breyer, C. (2020). Towards sustainable development in the mena region: Analysing the feasibility of a 100% renewable electricity system in 2030. *Energy Strategy Reviews*, 28:100466. 1

Agmon, S. (1954). The relaxation method for linear inequalities. *Canad. J. Math.*, 6:382–392. 87

Aharoni, R. and Censor, Y. (1989). Block-iterative projection methods for parallel computation of solutions to convex feasibility problems. *Linear Algebra and Its Applications*, 120:165–175. 87

Arteaga, J. and Zareipour, H. (2019). A price-maker/price-taker model for the operation of battery storage systems in electricity markets. *IEEE Transactions on Smart Grid*, 10(6):6912–6920. 20

Babbar, M. M. (1955). Distributions of solutions of a set of linear equations (with an application to linear programming). *Journal of the American Statistical Association*, 50(271):854–869. 85

Bai, Z.-Z. and Wu, W.-T. (2018). On greedy randomized Kaczmarz method for solving large sparse linear systems. *SIAM J. Sci. Comput.*, 40(1):A592–A606. 87

Bakhshi, M., Ostrowski, J., and Tikhomirov, K. (2024). On the optimal objective value of random linear programs. *arXiv preprint arXiv:2401.17530*. 87

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.*, 4:238–252. 98

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media. 32

Blair, C. (1986). Random linear programs with many variables and few constraints. *Math. Programming*, 34(1):62–71. 85

Borgwardt, K.-H. (1982). The average number of pivot steps required by the simplex-method is polynomial. *Z. Oper. Res. Ser. A-B*, 26(5):A157–A177. 85, 86

Borgwardt, K.-H. (1987). *The simplex method*, volume 1 of *Algorithms and Combinatorics: Study and Research Texts*. Springer-Verlag, Berlin. A probabilistic analysis. 85, 86

Borgwardt, K. H. (1999). A sharp upper bound for the expected number of shadow vertices in LP-polyhedra under orthogonal projection on two-dimensional planes. *Math. Oper. Res.*, 24(3):544–603. 85, 86

Brijs, T., Geth, F., Siddiqui, S., Hobbs, B. F., and Belmans, R. (2016). Price-based unit commitment electricity storage arbitrage with piecewise linear price-effects. *Journal of Energy Storage*, 7:52–62. 8

Bui, V.-H., Hussain, A., and Kim, H.-M. (2019). Double deep q-learning-based distributed operation of battery energy storage system considering uncertainties. *IEEE Transactions on Smart Grid*, 11(1):457–469. 31

Cao, J., Harrold, D., Fan, Z., Morstyn, T., Healey, D., and Li, K. (2020). Deep reinforcement learning-based energy storage arbitrage with accurate lithium-ion battery degradation model. *IEEE Transactions on Smart Grid*, 11(5):4513–4521. 30, 31

Chandrasekaran, K. and Vempala, S. S. (2014). Integer feasibility of random polytopes: random integer programs. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 449–458. 85

Chazarra, M., Pérez-Díaz, J. I., and García-González, J. (2017). Value of perfect information of spot prices in the joint energy and reserve hourly scheduling of pumped storage plants. *Electric Power Systems Research*, 148:303–310. 29

Cheng, B. and Powell, W. B. (2016). Co-optimizing battery storage for the frequency regulation and energy arbitrage using multi-scale dynamic programming. *IEEE Transactions on Smart Grid*, 9(3):1997–2005. 29, 30

Child, M., Bogdanov, D., and Breyer, C. (2018). The role of storage technologies for the transition to a 100% renewable energy system in europe. *Energy Procedia*, 155:44–60. 1

Chubanov, S. (2015). A polynomial projection algorithm for linear feasibility problems. *Math. Program.*, 153(2, Ser. A):687–713. 87

Dadush, D. and Huiberts, S. (2018). A friendly smoothed analysis of the simplex method. In *STOC'18—Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 390–403. ACM, New York. 85

De Loera, J. A., Haddock, J., and Needell, D. (2017). A sampling Kaczmarz-Motzkin algorithm for linear feasibility. *SIAM J. Sci. Comput.*, 39(5):S66–S87. 87

Diaconis, P. and Miclo, L. (2013). On the spectral analysis of second-order markov chains. In *Annales de la Faculté des sciences de Toulouse: Mathématiques*, volume 22, pages 573–621. 75

Dicorato, M., Forte, G., Pisani, M., and Trovato, M. (2012). Planning and operating combined wind-storage system in electricity market. *IEEE Transactions on Sustainable Energy*, 3(2):209–217. 28

Ding, H., Hu, Z., and Song, Y. (2012). Stochastic optimization of the daily operation of wind farm and pumped-hydro-storage plant. *Renewable Energy*, 48:571–578. 29

Dyer, M. E., Frieze, A. M., and McDiarmid, C. J. (1986). On linear programs with random costs. *Mathematical Programming*, 35:3–16. 85

Elfving, T. (1980). Block-iterative methods for consistent and inconsistent linear equations. *Numerische Mathematik*, 35:1–12. 93

Feng, L., Zhang, X., Li, C., Li, X., Li, B., Ding, J., Zhang, C., Qiu, H., Xu, Y., and Chen, H. (2022). Optimization analysis of energy storage application based on electricity price arbitrage and ancillary services. *Journal of Energy Storage*, 55:105508. 29

García-Miguel, P. L. C., Alonso-Martínez, J., Arnaltes Gómez, S., García Plaza, M., and Asensio, A. P. (2022). A review on the degradation implementation for the operation of battery energy storage systems. *Batteries*, 8(9):110. 29

Gower, R. M., Molitor, D., Moorman, J., and Needell, D. (2021). On adaptive sketch-and-project for solving linear systems. *SIAM J. Matrix Anal. Appl.*, 42(2):954–989. 87

Gower, R. M. and Richtárik, P. (2015). Randomized iterative methods for linear systems. *SIAM J. Matrix Anal. Appl.*, 36(4):1660–1690. 87

Greenblatt, J. B., Succar, S., Denkenberger, D. C., Williams, R. H., and Socolow, R. H. (2007). Baseload wind energy: modeling the competition between gas turbines and compressed air energy storage for supplemental generation. *Energy policy*, 35(3):1474–1492. 28

Guignard, M. (2003). Lagrangean relaxation. *Top*, 11:151–200. 98

Higle, J. L. and Sen, S. (1991). Statistical verification of optimality conditions for stochastic programs with recourse. *Annals of Operations Research*, 30(1):215–239. 98

Huiberts, S., Lee, Y. T., and Zhang, X. ([2023] ©2023). Upper and lower bounds on the smoothed complexity of the simplex method. In *STOC'23—Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1904–1917. ACM, New York. 85

Jiang, D. R. and Powell, W. B. (2015). Optimal hour-ahead bidding in the real-time electricity market with battery storage using approximate dynamic programming. *INFORMS Journal on Computing*, 27(3):525–543. 30

Jiao, Y., Jin, B., and Lu, X. (2017). Preasymptotic convergence of randomized Kaczmarz method. *Inverse Problems*, 33(12):125012, 21. 87

Joshi, A., Kebriaei, H., Mariani, V., and Glielmo, L. (2021). A sufficient condition to guarantee non-simultaneous charging and discharging of household battery energy storage. *arXiv preprint arXiv:2104.06267*. 8

Kim, J. H. and Powell, W. B. (2011). Optimal energy commitments with storage and intermittent supply. *Operations research*, 59(6):1347–1360. 28

Klatt, M., Munk, A., and Zemel, Y. (2022). Limit laws for empirical optimal solutions in random linear programs. *Annals of Operations Research*, 315(1):251–278. 85

Klein, N., Smith, M. S., and Nott, D. J. (2023). Deep distributional time series models and the probabilistic forecasting of intraday electricity prices. *Journal of Applied Econometrics*, 38(4):493–511. 31

Laporte, G. and Louveaux, F. V. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142. 98

Li, Y., Ding, Y., Liu, Y., Yang, T., Wang, P., Wang, J., and Yao, W. (2022). Dense skip attention based deep learning for day-ahead electricity price forecasting. *IEEE Transactions on Power Systems*, 38(5):4308–4327. 31

Liu, J. and Wright, S. J. (2016). An accelerated randomized Kaczmarz algorithm. *Math. Comp.*, 85(297):153–178. 87

Liu, S., Bunea, F., and Niles-Weed, J. (2023). Asymptotic confidence sets for random linear programs. *arXiv preprint arXiv:2302.12364*. 85

Mercier, T., Olivier, M., and De Jaeger, E. (2023). The value of electricity storage arbitrage on day-ahead markets across europe. *Energy Economics*, 123:106721. 2

Mohamed, A., Rigo-Mariani, R., Debusschere, V., and Pin, L. (2024). Operational planning strategies to mitigate price uncertainty in day-ahead market for a battery energy system. *IEEE Access*, 12:85388–85399. 31

Morshed, M. S., Islam, M. S., and Noor-E-Alam, M. (2022). Sampling kaczmarz-motzkin method for linear feasibility problems: generalization and acceleration. *Mathematical Programming*, 194(1-2):719–779. 87

Motzkin, T. S. and Schoenberg, I. J. (1954). The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6:393–404. 87

Muniain, P. and Ziel, F. (2020). Probabilistic forecasting in day-ahead electricity markets: Simulating peak and off-peak prices. *International Journal of Forecasting*, 36(4):1193–1210. 31

Needell, D. and Tropp, J. A. (2014). Paved with good intentions: analysis of a randomized block Kaczmarz method. *Linear Algebra Appl.*, 441:199–221. 93

Nowotarski, J. and Weron, R. (2018). Recent advances in electricity price forecasting: A review of probabilistic forecasting. *Renewable and Sustainable Energy Reviews*, 81:1548–1568. 30

Pan, K., Zhao, M., Li, C.-L., and Qiu, F. (2022). A polyhedral study on fuel-constrained unit commitment. *INFORMS Journal on Computing*. 9, 26

Penaranda, A. F., Romero-Quete, D., and Cortés, C. A. (2021). Grid-scale battery energy storage for arbitrage purposes: A colombian case. *Batteries*, 7(3):59. 29

Perez, A., Moreno, R., Moreira, R., Orchard, M., and Strbac, G. (2016). Effect of battery degradation on multi-service portfolios of energy storage. *IEEE Transactions on Sustainable Energy*, 7(4):1718–1729. 29

Powell, W. B. and Meisel, S. (2015). Tutorial on stochastic optimization in energy—part ii: An energy storage illustration. *IEEE Transactions on Power Systems*, 31(2):1468–1475. 3, 56

Pozo, D., Contreras, J., and Sauma, E. E. (2014). Unit commitment with ideal and generic energy storage units. *IEEE Transactions on Power Systems*, 29(6):2974–2984. 8

Prékopa, A. (1966). On the probability distribution of the optimum of a random linear program. *SIAM Journal on Control*, 4(1):211–222. 85

Seel, J., Millstein, D., Mills, A., Bolinger, M., and Wiser, R. (2021). Plentiful electricity turns wholesale prices negative. *Advances in Applied Energy*, 4:100073. 9

Sen, S. (1993). Subgradient decomposition and differentiability of the recourse function of a two stage stochastic linear program. *Operations research letters*, 13(3):143–148. 98

Shen, Z., Wei, W., Wu, D., Ding, T., and Mei, S. (2020). Modeling arbitrage of an energy storage unit without binary variables. *CSEE Journal of Power and Energy Systems*, 7(1):156–161. 8

Shin, K. and Lee, J. (2024). Investment decision for long-term battery energy storage system using least squares monte carlo. *Energies*, 17(9):2019. 32

Shuai, H., Li, F., Pulgar-Painemal, H., and Xue, Y. (2021). Branching dueling q-network-based online scheduling of a microgrid with distributed energy storage systems. *IEEE Transactions on Smart Grid*, 12(6):5479–5482. 31

Sioshansi, R., Denholm, P., Jenkin, T., and Weiss, J. (2009). Estimating the value of electricity storage in pjm: Arbitrage and some welfare effects. *Energy economics*, 31(2):269–277. 28

Smale, S. (1983). On the average number of steps of the simplex method of linear programming. *Math. Programming*, 27(3):241–262. 85

Song, Y. and Luedtke, J. (2015). An adaptive partition-based approach for solving two-stage stochastic programs with fixed recourse. *SIAM Journal on Optimization*, 25(3):1344–1367. 98

Spielman, D. A. and Teng, S.-H. (2004). Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463. 85, 86

Stathakis, E., Papadimitriou, T., and Gogas, P. (2021). Forecasting price spikes in electricity markets. *Review of Economic Analysis*, 13(1):65–87. 31

Toufani, P., Karakoyun, E. C., Nadar, E., Fosso, O. B., and Kocaman, A. S. (2023). Optimization of pumped hydro energy storage systems under uncertainty: A review. *Journal of Energy Storage*, 73:109306. 29

Truquet, L. (2019). Local stationarity and time-inhomogeneous markov chains. 36

Tschora, L., Pierre, E., Plantevit, M., and Robardet, C. (2022). Electricity price forecasting on the day-ahead market using machine learning. *Applied Energy*, 313:118752. 31

van Ackooij, W., de Oliveira, W., and Song, Y. (2018). Adaptive partition-based level decomposition methods for solving two-stage stochastic programs with fixed recourse. *Informs Journal on Computing*, 30(1):57–70. 98

Van Slyke, R. M. and Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM journal on applied mathematics*, 17(4):638–663. 98

Vedullapalli, D. T., Hadidi, R., and Schroeder, B. (2019). Combined hvac and battery scheduling for demand response in a building. *IEEE Transactions on Industry Applications*, 55(6):7008–7014. 9

Vejdan, S. and Grijalva, S. (2018). The value of real-time energy arbitrage with energy storage systems. In *2018 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE. 28

Vershynin, R. (2009). Beyond Hirsch conjecture: walks on random polytopes and smoothed complexity of the simplex method. *SIAM J. Comput.*, 39(2):646–678. 85

Walawalkar, R., Apt, J., and Mancini, R. (2007). Economics of electric energy storage for energy arbitrage and regulation in new york. *Energy Policy*, 35(4):2558–2568. 28

Wang, H. and Zhang, B. (2018). Energy storage arbitrage in real-time markets via reinforcement learning. In *2018 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE. 30, 58

Wankmüller, F., Thimmapuram, P. R., Gallagher, K. G., and Botterud, A. (2017). Impact of battery degradation on energy arbitrage revenue of grid-level energy storage. *Journal of Energy Storage*, 10:56–66. 29

Watkins, C. J. C. H. (1989). Learning from delayed rewards. 75

Xi, X., Sioshansi, R., and Marano, V. (2014). A stochastic dynamic programming model for co-optimization of distributed energy storage. *Energy Systems*, 5:475–505. 28, 29, 30

Zhang, Y. and Li, H. (2023). Randomized block subsampling Kaczmarz-Motzkin method. *Linear Algebra Appl.*, 667:133–150. 93

Zhou, Y., Scheller-Wolf, A., Secomandi, N., and Smith, S. (2019). Managing wind-based electricity generation in the presence of storage and transmission capacity. *Production and Operations Management*, 28(4):970–989. 28

Zverovich, V., Fábián, C. I., Ellison, E. F., and Mitra, G. (2012). A computational study of a solver system for processing two-stage stochastic lps with enhanced benders decomposition. *Mathematical Programming Computation*, 4:211–238. 98

Appendix

Appendix A

Computational results for chapter 3

A.1 Unbounded price trajectories and similar optimal policy

Figures A.1 and A.2 show the solution of SDP for two different price trajectories of type unbounded time-homogeneous. Although the prices are different, the optimal policies produced by SDP are exactly the same.

We can see the correctness of equation **??** from the figures. For instance, at time $t = 4$ in both figures the state of charge is at full capacity. However, the price of charge at this time in the trajectory given in Figure A.1 is 42 and in Figure A.2 is 38. The value function at this time are 481.2 and 441.2. The difference of value function for these two states is the price difference multiplied by the state of charge.

A.2 Bounded price trajectories and distinct optimal policies

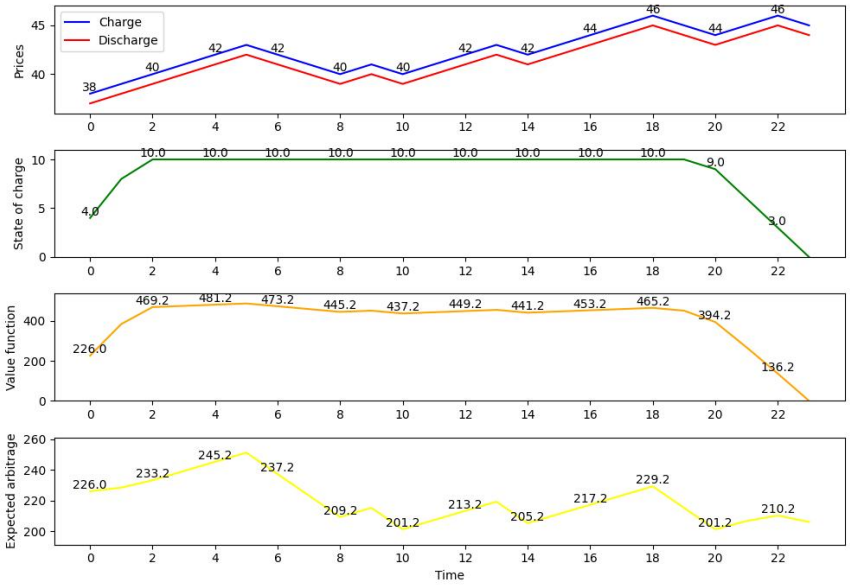

Figure A.1: SDP solution for unbounded time-homogeneous prices with 0.7 probability of going up. Total profit = 226.0, sample size = 10000.

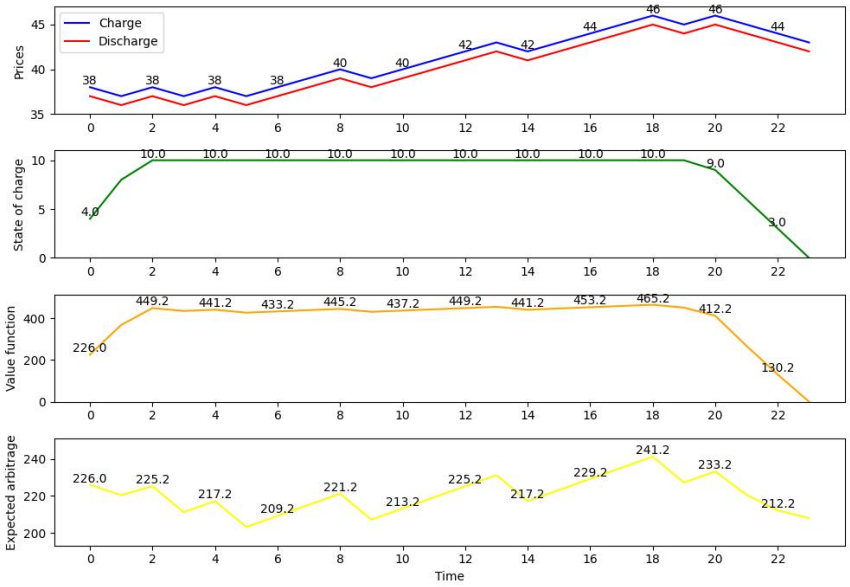

Figure A.2: SDP solution for unbounded time-homogeneous prices with 0.7 probability of going up. Total profit = 226.0, sample size = 10000.

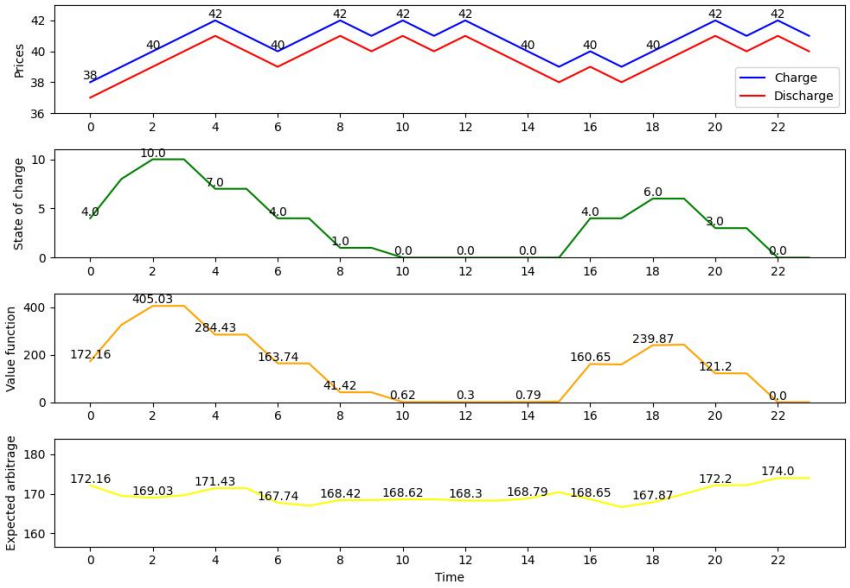

Figure A.3: SDP solution for bounded time-homogeneous prices with 0.7 probability of going up. Total profit = 172.16, sample size = 10000.

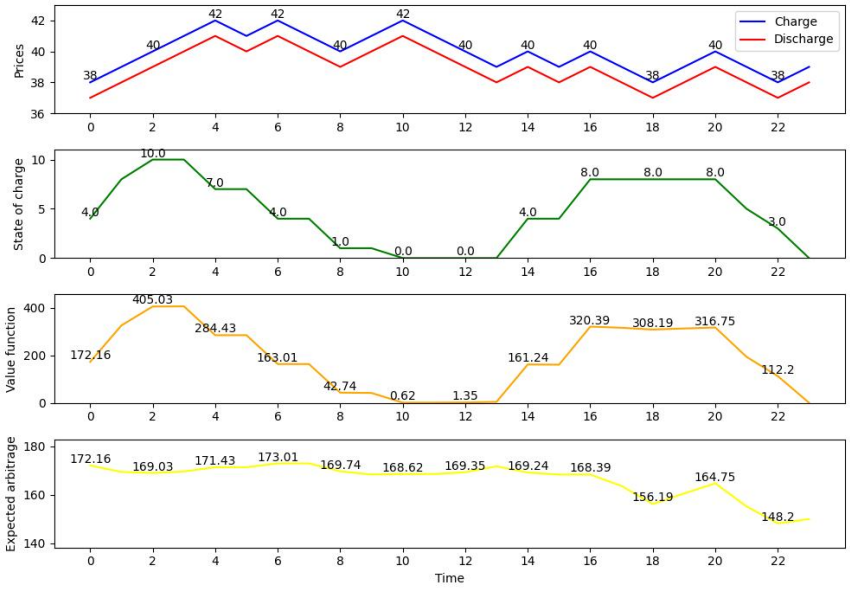

Figure A.4: SDP solution for bounded time-homogeneous prices with 0.7 probability of going up. Total profit = 172.16, sample size = 10000.

Vita

Marzieh Bakhshi was born in Mashhad, Iran. She earned a Bachelor's degree in Mathematics from Ferdowsi University of Mashhad in 2009. After graduation, she taught mathematics at a high school for four years. In 2017, she pursued a Master's degree in Applied Mathematics at Miami University in Oxford, Ohio.

During her master's program, Marzieh worked with Dr. Wards on a project in variational optimization and collaborated with Dr. Anna Ghazarian on a project related to traveling wave solutions, a system of partial differential equations in the context of social sciences.

In the Fall of 2020, she began her Ph.D. in the Department of Industrial and Systems Engineering at the University of Tennessee, Knoxville. Under the supervision of Dr. James Ostrowski, she focused her research on the optimization of battery energy storage for arbitrage.

With a background in mathematics and valuable experience in programming and computation during her PhD, Marzieh enjoys engaging and collaborating in interdisciplinary research, where theory and application come together.