2015

Power optimization in electricity networks

Fang Chen
Lehigh University

Power Optimization in Electricity Networks

by

Fang Chen

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Industrial Engineering

Lehigh University

August, 2015

Approved and recommended for acceptance as a Ph.D. thesis in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

_____
Date

_____
Dissertation Advisor

Committee Members:

_____
(Lawrence V. Snyder), Committee Chair

_____
(Binayak Roy)

_____
(Boris Defourny)

_____
(Shalinee Kishore)

Acknowledgements

First and foremost I would like to thank my advisor Professor Larry Snyder for supporting me for the past five years. It has been an honor to be one of his Ph.D. students. He has shared with me many brilliant ideas and suggestions about my research. I appreciate all his contributions of time and effort to make my Ph.D. experience productive. I am also thankful for the excellent example he has provided as a very innovative and enthusiastic researcher, which is motivational for me. He always encourages me to explore new ideas, supports me to overcome difficulties, and guide me through unknown domain of knowledge. His patience, motivation, immense knowledge toward the energy area, taken together, made him a great mentor to me.

I would also like to thank the members of my PhD committee, Dr. Binayak Roy, Professors Boris Defourny and Professor Shalinee Kishore. Dr. Roy was my supervisor during my internship with Bosch Research Center. Working with him is always a joyful experience. He is very enthusiastic and skillful toward applying his domain knowledge to solve real world problems. He has provided many insights toward the the energy industry as well as valuable suggestions toward my research and career. Professor Shalinee Kishore introduced me to smart grid and renewable system, which is extremely useful for my research. Professor Boris Defourny provided many theoretical foundation toward robust optimization, which is very inspiring. This dissertation would not have been possible without their persistent help.

I also received help from many of my colleagues and friends. I'm grateful to Professor Tamas Terlaky, Professor. Alberto J. Lamadrid, Dr. Lin He, Dr. Chen Chen, Yunfei Song, Zheng Han for their time and inspiring questions toward my research. I'm also thankful

to Dr. Maksim Subbotion, Dr. Dagmar Becker, Dr. Yusef Shafi, Dr. Jessie Huang, Asma Hamzaoui, Dr. Tolga Seyhan, Dr. Elcin Cetinkaya, Dr. Jiadong Wang, Dr. Alper Uygur, Dan Li, Hao Wang, Tengjiao Xiao, Xi Bai, Choat Inthawongse, Xiao Meng, Xiaocun Que for their support, companionship, and wonderful suggestions toward my career and life in general. I appreciate the wonderful moments we had together.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. I was honored to be a Presidential Fellow for the first year; my second and third year was funded by the Lehigh ISE department. During the last two years my work was supported by Bosch Battery Storage Solutions (BESS) team. I especially want to thank Dr. Jasim Ahmed from BESS who makes the funding possible.

Lastly, I would like to thank my family for all their love and encouragement. For my parents who raised me with a love of science and supported me in all my pursuits. And most of all for my husband Junyu Guo, who is a true and great supporter at all times, and always been by my side during the entire time. He has faith in me and my intellect which made me became more brave and confidence. Thank you.

Contents

Acknowledgements **iv**

List of Tables **ix**

List of Figures **x**

Abstract **1**

1 Introduction **3**

2 Efficient Algorithms and Policies for Demand Response Scheduling **5**

2.1 Introduction 5

2.2 Literature Review 7

2.3 Formulation and DP Models 8

2.3.1 Problem Formulation 8

2.3.2 Exact DP Model 10

2.3.3 ADP Algorithm 11

2.4 Fast EMC Policies 14

2.4.1 General Scheme 14

2.4.2 Policies 15

2.5 Computational Study 16

2.5.1 Experimental Design 16

2.5.2 Performance of ADP Algorithm 18

2.5.3 Performance of Policies 20

2.5.4 Comparison to No-EMC Case 23
2.5.5 Relationship between Cost and Power Limit 24
2.6 Conclusions 26
2.7 Future Work 26

3 Power Consumption Optimal Control with a Battery Energy Storage System (BESS) under Demand Charge Tariff 28

3.1 Introduction 28
3.2 Literature Review 30
3.3 Notation 34
3.4 Deterministic Load Profile 35
3.4.1 Battery Constraints 35
3.4.2 Minimax Route Dynamic Programming Algorithm 36
3.4.3 Deterministic Case Study 39
3.4.4 Computational Performance of Deterministic Case Study 40
3.5 Stochastic Load Profile 41
3.6 Dynamic Programming Algorithm 42
3.6.1 Introduction to Dynamic Programming Algorithm 42
3.6.2 Stochastic Load Case Study 46
3.6.3 Computational Performance of Demand Charge Algorithm - A posteriori 47
3.7 Sample Average Approximation Algorithm for Stochastic Loads 55
3.7.1 Introduction to SAA Algorithm 55
3.7.2 Computational Performance of SAA Algorithm 56
3.8 Naive Algorithms 57
3.8.1 Naive Algorithm 1 60
3.8.2 Naive Algorithm 2 63
3.8.3 Naive Algorithm 3 71
3.9 Summary of Results 73
3.10 Performance Analysis 74

3.11 Real Time Planning 78
3.11.1 Introduction to Real Time Planning Algorithm 78
3.11.2 Computational Performance of Real Time Planning Algorithm . . . 80
3.12 Conclusions and Future Work 86

4 A PV (Photovoltaic) Plant Power Feed-in Problem with Battery Storage System under Profile Constraints 89

4.1 Problem Introduction 89
4.2 Literature Review 90
4.3 Deterministic PV Feed-in Problem 92
4.3.1 Problem Formulation for the Deterministic Case 95
4.3.2 Analysis of the Problem 99
4.3.3 Illustrative Examples 100
4.3.4 Starting with Nonempty Battery 105
4.3.5 Case Study 106
4.4 Stochastic PV Feed-in Problem 110
4.4.1 Dynamic Programming Model of Energy Sold Maximization 111
4.4.2 Dynamic Programming Algorithm 113
4.4.3 Incorporating Battery Efficiency 114
4.4.4 Case Study 116
4.4.5 Incorporating the Three Phase Phase Constraints 120
4.5 The Search of Structured Solutions 123
4.5.1 Deterministic Case 124
4.5.2 Stochastic Case 127
4.6 Conclusions and Future Work 138

5 Conclusions and Future Work 141

Biography 152

List of Tables

2.1 Average peak loads 24

3.1 Result summary of different algorithms 74
3.2 Result summary of DP and SAA algorithms on different instances 76
3.3 Result summary of Naive Algorithms on different instances 77

4.1 Comparison between sunny and cloudy day 104
4.2 Energy feed-in (kWh) for different systems 109
4.3 Data and solution for example with small battery 127
4.4 Data and solution for example with large battery 128
4.5 A summary of the threshold values 135

List of Figures

2.1 State transition diagram of an appliance 9

2.2 Objective function value vs. M 19

2.3 CPU time vs. M 19

2.4 Comparison of policies over 160 instances 21

2.5 Loads generated during middle 3 days of horizon 22

2.6 Average hourly load for middle 3 days of horizon 23

2.7 Average total cost vs. power limit L_t 25

2.8 Penalty cost vs electricity cost 26

3.1 A demonstration of deterministic case 38

3.2 Net load values 40

3.3 Result of case study 41

3.4 CPU time vs. relative error of standard deviation 49

3.5 CPU time vs. relative error of mean 49

3.6 CPU time vs. relative error - fixed δ_s, δ_u 50

3.7 CPU time vs. relative error - fixed δ_p, δ_u 51

3.8 Maximum grid purchase for fist 100 instances, DP algorithm. 51

3.9 Grid purchase for instance 11, DP algorithm 53

3.10 Battery SOE for instance 11, DP algorithm 53

3.11 Grid purchase for instance 81, DP algorithm 54

3.12 Battery SOE for instance 81, DP algorithm 54

3.13 Maximum grid purchase for first 100 instances, SAA algorithm 58

3.14 Grid purchase for instance 11, SAA algorithm 58
3.15 Battery SOE for instance 11, SAA algorithm 59
3.16 Grid purchase for instance 81, SAA algorithm 59
3.17 Battery SOE for instance 81, SAA algorithm 60
3.18 Maximum grid purchase for first 100 instances, Naive Algorithm 1 63
3.19 Grid purchase for instance 11, Naive Algorithm 1 64
3.20 Battery SOE for instance 11, Naive Algorithm 1 64
3.21 Grid purchase for instance 81, Naive Algorithm 1 65
3.22 Battery SOE for instance 81, Naive Algorithm 1 65
3.23 Grid purchase for instance 22, Naive Algorithm 1 66
3.24 Battery SOE for instance 22, Naive Algorithm 1 66
3.25 Maximum grid purchase for first 100 instances, Naive Algorithm 2 68
3.26 Grid purchase for instance 11, Naive Algorithm 2 68
3.27 Battery SOE for instance 11, Naive Algorithm 2 69
3.28 Grid purchase for instance 81, Naive Algorithm 2 69
3.29 Battery SOE for instance 81, Naive Algorithm 2 70
3.30 Grid purchase for instance 22, Naive Algorithm 2 70
3.31 Battery SOE for instance 22, Naive Algorithm 2 71
3.32 μ_r vs. σ_r for $r \in \{0.5 : 0.01 : 2\}$ 73
3.33 Comparison of policies over 20 instances 75
3.34 Comparison of policies over 20 instances 75
3.35 Comparison of policies over 20 instances 78
3.36 Facility historical load 81
3.37 Weekday and weekend load 82
3.38 Perfect knowledge for day 26 84
3.39 Performance summary for one year example 86
3.40 Real time planning for day 26 87
3.41 Evolving of battery SOE level 87

4.1 System layout 90

4.2 PV power generation, sunny day 93
4.3 PV power generation, sunny day 95
4.4 Power feed-in, sunny day 102
4.5 Battery SOE, sunny day 102
4.6 Power feed-in, cloudy day 103
4.7 Battery SOE, cloudy day 104
4.8 Power feed-in and battery actions, cloudy day 105
4.9 Power feed-in and battery actions, cloudy day 106
4.10 Performance profile for models A and B 108
4.11 The Mean PV output for Small Scale Instance 114
4.12 Feed-in comparison 115
4.13 Feed-in comparison 117
4.14 Revenue increase vs. battery efficiency 118
4.15 Performance of different batteries 119
4.16 The revenue function at time T 129
4.17 Future cost at time $T-1$ for fixed $\overline{PV}$ 130
4.18 The revenue function at time T 131
4.19 The current cost and future cost $T-1$, b=60 kWh 132
4.20 The revenue function at time $T-1$, b= 60 kWh 132
4.21 The price variation 134
4.22 Demonstration of single threshold policy, $\gamma = 0.9$ 136
4.23 Demonstration of single threshold policy-con't. 137
4.24 Demonstration of dual threshold structure 138
4.25 Demonstration of dual threshold policy, $\gamma = 0.9$ 139
4.26 Demonstration of threshold policy-con't. 140

Abstract

Today, one main challenge that the energy industry faces is the ability to increase energy efficiency. This requires effort from two entities within the energy system the rule/policy maker from the upper level who creates standards to properly regulate energy usage or energy-trading processes. Another entity is the rule follower, who reacts to the rules or polices wisely to maximize its own benefits taking into consideration economic and quality-of-life issues. This thesis studies three problems to help the rule follower increase its benefit in the electricity market. In the first problem, a power consumer aims to dispatch its power usage over time given different electricity price rates, appliance characteristics, and power limit constraints. We propose an approximate dynamic programming (ADP) algorithm, which works well for small instances. We also propose several scheduling policies for very fast solutions of large scale problems. We show that sorting the requested appliances according to their operating urgency improves the cost. This result allows the power consumer to dispatch the power usage properly and quickly. In the second problem, a demand charge cost is incurred according to the peak load for some large power consumers. A battery system is introduced for peak shaving. Under load uncertainty, we develop a stochastic DP model to solve the problem optimally as well as a Sample Average Approximation (SAA) algorithm for real time implementation. We also introduce several Naive Algorithms for comparison with the DP and SAA algorithm. Finally, we introduce a real time SAA algorithm and test its performance on a data set consisting of 365 days. This algorithm is very effective in terms of generating real time power dispatch plans. In the third problem, we look at the solar power trading problem between a PV farm and the grid operator who imposes complex constraints on the power profile. We propose a Mixed

Integer Programming model assuming perfect knowledge of the PV output. Then we relax some of the constraints, and develop a dynamic programming model for the stochastic load problem. We show that a threshold structure battery inventory solution exists for the relaxed problem. We also propose a dynamic programming model with respect to the key constraints from the grid operator.

Chapter 1

Introduction

The world now faces increasing environmental and economic challenges related to energy consumption. According to the Energy Information Administration (EIA), energy demand has almost doubled over the past twenty years [EIA, 2014]. Because of the impending shortage of traditional energy resources, improving energy efficiency has become a worldwide topic. Energy efficiency, "doing more with the same amount of energy," is one of the easiest and most cost effective ways to improve the environment and reduce energy costs for consumers. In an electricity network, volatile energy costs, surging demands, and unpredictable generation from renewable sources are all factors that threaten grid energy efficiency and therefore need to be taken care of properly.

In our first research topic we study efficient mechanisms to optimize power consumption for end users in an electricity network. In a smart grid network, the end users are informed of changes in electricity prices, and then react to it. In order to properly schedule tasks for the end user, then we propose a mathematical programming algorithm for this problem. Moreover, to solve the problem efficiently, we then propose several heuristic scheduling policies that provide efficient solutions. The work in this chapter is forthcoming in the *Journal of Energy Engineering*. See Chen et al. [2014].

In our second research topic, we study the power consumption optimization problem with a battery energy storage system (BESS) under a demand charge tariff, assuming the load is stochastic. We first propose a mathematical programming model for this problem.

Then we develop several algorithms for the model. The performance of these algorithms is compared. After that, we develop an algorithm to solve this problem in real time using the load forecast as input.

In our third research topic, we introduce a PV (photovoltaic) plant power feed-in problem. To regulate power received from the PV generator, the grid operator has incurred complex constraints on the feed-in profile. A battery storage system is deployed to increase revenue during the renewable trading process. We develop algorithms for both the deterministic case and the stochastic case.

Chapter 2

Efficient Algorithms and Policies for Demand Response Scheduling

2.1 Introduction

The Smart Grid's communication capabilities allow electricity providers to change prices many times per day and to communicate these prices to consumers [Conejo et al., 2005]. This presents opportunities for consumers to reduce their electricity bills, as well as pitfalls for consumers who fail to react to pricing signals. Given this situation, customers will constantly face decisions about when (or whether) to operate their appliances. These decisions will be too frequent and too combinatorially complex to be made manually, and therefore consumers need automated methods for making smart consumption decisions in order to react optimally to price patterns. Moreover, overloaded electricity systems increase consumers' electricity bills and harm electricity service providers by reducing grid stability and causing blackouts and other serious problems. If consumers are better able to shift their consumption in response to pricing (which is in turn a response to anticipated supply and demand), service providers can reduce peak demands and therefore overloads. Thus, appliance scheduling has benefits both for system operators and for consumers. The scheduling problem we discuss here may be considered as a type of *demand response* (DR).

We consider the optimization problem faced by an *energy management controller*

(EMC), a piece of hardware and/or software that makes timing decisions about the operation of appliances or devices within a home, industrial facility, college campus, or other facility or set of facilities. The EMC receives scheduling requests from the user for individual devices and makes decisions regarding when to begin operation of each. Unfortunately, this scheduling problem is difficult to solve to optimality in the short time scales required for practical use. For example, Kishore and Snyder [2010] propose a dynamic programming (DP) algorithm to solve it, but their DP can solve only very small instances due to the "curse of dimensionality." Therefore, we propose an approximate dynamic programming (ADP) [Powell, 2001] algorithm to solve the problem. The ADP is significantly faster than the exact DP algorithm but is still unable to handle instances with more than a moderate number of appliances. Therefore, we also propose efficient policies that can handle large instances and provide near-optimal results extremely quickly.

Following Kishore and Snyder [2010] and Xiong et al. [2011], we assume that the total power consumption in each time period cannot exceed a pre-specified threshold. Although such constraints are not common in today's electricity market, we believe they will become increasingly applicable in the future. For example, the power limit may be set up by the electricity provider, either implicitly (through pricing mechanisms) or explicitly (as through interruptible load contracts [Baldick et al., 2006]). Alternately, it might arise from the consumer's budget constraint in order to reduce demand charges or monthly bills. (If there is no power limit, the problem we formulate is easy to solve; see Kishore and Snyder [2010].)

We also assume that the consumer indicates his or her preferences regarding the delay of each appliance by specifying a *delay cost* for each period of delay. (The delay cost can be set to a large number to model appliances that must be turned on as soon as requested.) The consumer also specifies the *maximum preferred delay*, and an appliance that must be delayed beyond its maximum preferred delay (due to scheduling constraints) is assumed to incur a *penalty cost* that is higher than the delay cost. For example, when the user requests operation of the dishwasher, he or she might specify that each hour of delay incurs the equivalent of \$0.25 of inconvenience, that the dishwasher is preferred to

begin operation by 10:00 PM, and that each hour of delay past 10:00 PM incurs $2.50 worth of inconvenience. We stress that the model does not assume a strict maximum delay constraint for the appliances, since such constraints may lead to infeasibilities, especially when the power consumption limit is low. On the other hand, the user can implicitly impose a hard constraint by setting the penalty cost to be very high.

The request and operating times of many appliances are random. For example, air conditioner compressors turn on and off based on air temperature, the user's laundry schedule may be unpredictable, etc. We model this randomness by using discrete-time stochastic processes. Moreover, we assume that, once on, an appliance must stay on without interruption until it completes its operation.

The remainder of this chapter is organized as follows. In Section 2.2, we briefly review the literature on EMC optimization problems. In Section 2.3, we formulate the problem and discuss DP-based approaches to solving it, first reviewing the exact DP formulation proposed by Kishore and Snyder [2010] and then introducing our ADP approach. In Section 2.4, we introduce our efficient scheduling policies. We perform a computational experiment to test the ADP and policies in Section 2.5 and briefly summarize our conclusions and propose future research directions in Section 2.6.

2.2 Literature Review

There is a growing literature on novel approaches to formulating and solving demand response problems. Common strategies include the design of pricing schemes and demand-side control and management. In general, pricing schemes encourage reduction of consumption during peak hours through proper choices of electricity prices. For example, Shao et al. [2010] study the impact of time-of-use (TOU) electricity rates on consumer behavior, while Li et al. [2011a] show the existence of time-varying prices that can align individual optimality with social optimality. Adelman and Uçkun [2013] investigate dynamic pricing approaches and explore the resulting improvement in social welfare, using air conditioning as the central focus.

Demand-side management makes use of smart meters to control power usage. Some

researchers [Mohsenian-Rad et al., 2010a, Chang et al., 1988] consider coordinated demand-side management architectures, which assume the exchange of information among consumers. Others [O'Neill et al., 2005, Liang and Shen, 2011] apply learning techniques to reduce electricity costs. Subramanian et al. [2012] develop heuristic causal scheduling policies to allocate power resources efficiently. Most of the literature makes strong assumptions regarding the predictability of device timing. For example, Mohsenian-Rad et al. [2010b] assume that the operating duration of each task is deterministic, while Mohsenian-Rad and Leon-Garcia [2011] and Danandeh et al. [2012] assume that request time windows are deterministic. In reality, consumer requirements are largely random, as are operation durations, e.g., for air conditioning.

Moreover, to the best of our knowledge, other than a few previous papers [Kishore and Snyder, 2010, Xiong et al., 2011], none of the existing literature considers appliance scheduling under power usage constraints. Kishore and Snyder [2010] propose a DP for this problem, as well as a simple and extremely efficient method for the problem without power usage constraints. Xiong et al. [2011] introduce simple scheduling policies (which they call admission control methods), focusing primarily on the communication scheme that coordinates appliance scheduling and operation.

Our chapter contributes to the literature on device scheduling for demand response by considering random timing for appliance requests and operating durations, as well as power limit constraints; and it improves upon the work of Kishore and Snyder [2010] and Xiong et al. [2011] by introducing more accurate and computationally efficient solution procedures.

2.3 Formulation and DP Models

2.3.1 Problem Formulation

Following Kishore and Snyder [2010], we consider a set of N appliances within a home. The planning horizon consists of T time periods. We assume that the electricity price C_t in period t is known, i.e., the home receives day-ahead prices from the service provider, an

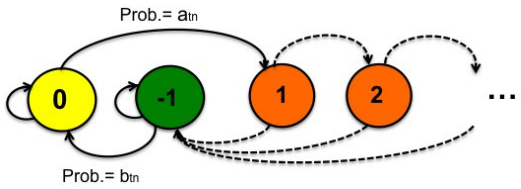


Figure 2.1: State transition diagram of an appliance

assumption that is common in the literature [see, e.g., Conejo et al., 2005].

An appliance can be in one of three states: *idle* means the appliance is off and has not been requested; *on* means the appliance is currently operating; and *requested* means the appliance is off but has been requested, e.g., when the user presses the start button on the washing machine but the EMC has not yet turned the machine on. Let S_{tn} denote the state of appliance n at time t, where $S_{tn} = 0$ indicates that the appliance is idle; $S_{tn} = -1$ indicates that the appliance is on; and $S_{tn} = r$, $(r = 1, 2, \ldots)$ indicates that the appliance has been in the requested state for r periods, including the current period. We denote the state vector at time t as $S_t = (S_{tn})_{n=1}^{N}$.

Appliances always transition among states in the following order: idle $\rightarrow$ requested $\rightarrow$ on $\rightarrow$ idle. Transitions from idle to requested, and from on to idle, follow a stochastic process. In particular, if an appliance is idle at time t, it will be requested during the next period $t + 1$ with probability a_{tn}; and if the appliance is operating at time t, it will finish operating at the end of period t with probability b_{tn}. (In the special case in which $a_{tn} = a_n$ and $b_{tn} = b_n$ for all t, the idle and on times are geometrically distributed and the process is Markovian.) On the other hand, appliances only transition from requested to on through actions taken by the EMC, i.e., decisions made by the optimization model.

Figure 2.1 depicts the state transition process of appliance n. The solid arrows represent random state transitions and the dashed arrows represent the state transitions brought about by actions taken by the EMC. In each time period, the action is taken first, and then we observe the realization of the random transitions.

We let A_t denote the *active set*, i.e., the set of appliances that are on at the start of period t. R_t is the *requested set*, i.e., the set of appliances that have been requested prior to the start of period t but have not yet been turned on. $\bar{R}_t \subseteq R_t$ is the subset of requested

appliances that have reached their maximum preferred delay. O_t is the *admission set*, i.e., the set of appliances that are chosen to be turned on in time t. O_t is the EMC's decision variable.

Appliance n uses P_n kWh of electricity per period when on (and 0 kWh otherwise). The total power usage in period t is constrained to be less than or equal to the power limit, L_t kWh.

The user specifies a delay cost d_{nt}, a maximum preferred delay ℓ_n, and a penalty cost $\bar{d}_{nt}$. Thus, the appliance incurs a cost of d_{nt} for each period of delay up to and including the ℓ_nth period and then incurs a cost of $\bar{d}_{nt}$ for each additional period of delay.

2.3.2 Exact DP Model

Let $f_t(S_t)$ be the optimal cost in periods $t, \ldots, T$ assuming that the system begins period t in state S_t. The DP recursion is as follows [Kishore and Snyder, 2010]:

$$f_t(S_t) = \min_{O_t \subseteq R_t} \left\{ \sum_{n \in O_t \cup A_t} C_t P_n + \sum_{n \in R_t \setminus (\bar{R}_t \cup O_t)} d_{nt} + \sum_{n \in \bar{R}_t \setminus O_t} \bar{d}_{nt} + \mathbb{E}[f_{t+1}(S_{t+1}|O_t)] \,\middle|\, \sum_{O_t \cup A_t} P_n \leq L_t \right\} \quad (2.1)$$

The DP minimizes the cost subject to the power limit constraint. For a given choice of the admission set O_t, the cost consists of the current costs (power consumption cost, delay cost, and penalty cost), and the expected future cost. The expectation inside the minimization is over all possible states S_{t+1} in period $t+1$ that result from transitions from the state at the end of period t, i.e., the state that results from modifying S_t to account for the appliances that are selected to be turned on in period t. Among all possible choices of O_t, the one that produces the smallest cost value will be chosen.

It is clear that this problem suffers from the curse of dimensionality. First, its state space grows exponentially in N. Second, the decision space is exponential in $|R_t|$ since we must make a yes/no decision about whether to turn on each appliance. Moreover, it is not straightforward to identify the optimal subset among the $2^{|R_t|}$ subsets of R_t since the

choice affects the cost in future periods as well as the current period. Kishore and Snyder [2010] suggest that the DP can be solved only for small values of N and T. Therefore, we propose both an ADP algorithm that is significantly more efficient than the (exact) dynamic programming approach above and a variety of scheduling policies that execute extremely quickly and are more forward-looking than the admission control policies proposed by Xiong et al. [2011].

2.3.3 ADP Algorithm

In the classical DP approach, (2.1) would be solved through (backward) recursion: Assume first that, for given $t+1$, the values of $f_{t+1}(S_{t+1})$ are known for all possible states S_{t+1}. Then, for each possible S_t, we calculate $f_t(S_t)$ by finding the O_t that minimizes the term inside the braces in (2.1). Each candidate choice of O_t implies a certain state at the end of period t that results from turning on the appliances in O_t in addition to those in A_t. We calculate the expectation using the transition probabilities from this ending state to all possible states S_{t+1}, and using the already-computed values of $f_{t+1}(S_{t+1})$. The recursion is initialized by assuming that the values for a dummy period $T+1$ are known (perhaps equal to 0).

As noted above, for all but the smallest instances, this approach is not practical for (2.1), since there are a combinatorial number of states S_t for each t and since the optimization problem for a given t and S_t is difficult to solve. Instead, we propose here an ADP algorithm [Powell, 2001] to solve larger instances of the appliance scheduling problem.

The basic idea behind ADP is to estimate values for $f_t(S_t)$, refining the estimates as the algorithm progresses. These estimates are denoted $\bar{f}_t^m(S_t)$, where m is an iteration counter, and they function like a "lookup table" that stores estimates for each t and S_t. Rather than enumerating the entire state space, the ADP algorithm approximates it via sampling, by simulating the system multiple times. Each time a state is visited, the algorithm updates the estimates for the cost function at that state.

Whereas the exact DP algorithm moves backward in time (beginning with period $T+1$ and working backwards to period 0), the ADP algorithm does the opposite. We

first choose an initial value for each t and S_t, which we denote $\bar{f}_t^0(S_t)$. (We choose to set $\bar{f}_t^0(S_t) = 0$ for all t, S_t.) We wish to estimate $f_0(S_0)$, the cost for the entire horizon starting from the initial state S_0, which is assumed to be given. As in the classical DP algorithm, we evaluate the cost in each period recursively, as though we already know the cost function for future periods; the difference is that in ADP, the future costs are known only approximately. Given the current estimates $\bar{f}_1^0(S_1)$, we solve the minimization in (2.1) with f replaced by $\bar{f}$ to determine O_0, as described in the next paragraph, and we update $\bar{f}_0^0(S_0)$ by replacing it with the cost computed for O_0. The algorithm proceeds in this way, in each period t optimizing (2.1) with f replaced by $\bar{f}$ to determine O_t and replacing $\bar{f}_t^0(S_t)$ with the (approximate) cost of O_t.

In each period t, we solve the minimization problem in (2.1) by enumerating all possible combinations of appliances $O_t \subseteq R_t$ to turn on and choosing the set that results in the smallest cost. Obviously, this approach is only practical for moderately small instances. A more efficient algorithm for this combinatorial problem would improve the performance of the ADP algorithm as a whole, a topic that represents a promising future research direction. To evaluate the cost of a candidate solution O_t, we calculate the cost inside the braces in (2.1), approximating the expectation $\mathbb{E}[f_{t+1}(S_{t+1}|O_t)]$ in two ways. First, rather than enumerating all possible future states resulting from S_t and O_t, we approximate the future using sampling, generating several possible states S_{t+1} (we sampled 40 states in our implementation) using the state resulting from S_t and O_t, as well as the transition probabilities, and calculating their mean. Second, since we do not know the costs of these future states exactly, we instead use the current approximation $\bar{f}_{t+1}^{m-1}$.

After performing the optimization and update steps in period t, we simulate a random transition from the ending state in period t (defined by S_t and O_t) to a new state S_{t+1} in period $t+1$, calculate $\bar{f}_t^0(S_t)$, and so on. The ADP algorithm proceeds in this manner, simulating a sample path through the time horizon and updating the cost estimates for a given state whenever we visit that state.

This gives us a single estimate for $f_0(S_0)$. Typically, this estimate is not sufficiently accurate, so we repeat the procedure for a total of M iterations, each consisting of a new

sample path and an updated estimate for $f_0(S_0)$. States that are more likely to occur in the true system are also more likely to be visited in the sample paths (as a result of both the random process and the actions chosen).

The ADP algorithm for our demand response problem can be summarized as follows (adapted from Powell [2001]). In the algorithm description, S_t^m refers to the system state in period t during iteration (i.e., sample path) m.

1. Given an initial state S_0 and a number of iterations M. Initialize $\bar{f}_t^0(S_t)$ for all periods t and all states S_t. Set $m = 1$.

2. Set $t = 0$ and $S_0^m = S_0$.

3. Solve

$$\hat{f}_t = \min_{O_t \subseteq R_t} \left\{ \sum_{n \in O_t \cup A_t} C_t P_n + \sum_{n \in R_t \setminus (\bar{R}_t \cup O_t)} d_{nt} + \sum_{n \in \bar{R}_t \setminus O_t} \bar{d}_{nt} + \sum_{s' \in \varsigma} P(s' | S_t^m, O_t) \bar{f}_{t+1}^{m-1}(s') \right\} \quad (2.2)$$

   where:

   $\hat{f}_t$ is the estimated cost at time t

   s' is the state the system transitions to in period $t+1$, given the current state S_t^n and the decision O_t

   ς is the set of possible states that the system can transition to, given the current state S_t^n and O_t

   $P(s' | S_t^m, O_t)$ is the probability of transitioning to state $s' \in \varsigma$ in period $t+1$, given that the current state is S_t^m and that we choose to operate the appliances in O_t

   Let O_t^m be the optimal solution to the minimization problem.

4. Update $\bar{f}_t^{m-1}(S_t)$ by replacing the current value of $\bar{f}_t^{m-1}(S_t^m)$ with $\hat{f}$ and leaving $\bar{f}_t^{m-1}(S_t)$ intact for all $S_t \neq S_t^m$.

5. Simulate a random transition to the next state by setting $S_{t+1}^m = S^M(S_t^m, O_t^m)$, where $S^M(\cdot)$ is a function that randomly maps the current state and decision to the next state. Set $t = t + 1$. If $t \leq T$, go to step 3.

6. Set $m = m + 1$. If $m \leq M$, go to step 2.

This ADP algorithm is more efficient than the traditional DP algorithm. However, even the ADP is too slow for large instances. (See Section 2.5.2 for more details.) Therefore, in the next section, we introduce fast EMC policies.

2.4 Fast EMC Policies

Since EMCs need to make decisions on short time scales, we now introduce several policies that execute extremely quickly and produce solutions that are close to optimal. These policies may be thought of as extensions of the admission control methods (ACMs) of Kishore and Snyder [2010].

2.4.1 General Scheme

The general scheme of our EMC policies in a given period t is as follows:

1. Identify the requested appliances, which form the requested set R_t.

2. For each requested appliance, if the appliance can be turned on without violating the power limit constraint, then add it to the *admissible set* D_t. That is,

$$D_t = \left\{ n \in R_t \,\middle|\, P_n + \sum_{m \in A_t} P_m \leq L_t \right\}.$$

3. Sort the appliances in D_t according to some rule (as specified by the individual policies; see Section 2.4.2). Set $O_t = \emptyset$.

4. Let n be the first appliance in D_t. Remove n from D_t.

5. If $P_n + \sum_{m \in A_t \cup O_t} P_m \leq L_t$, add n to O_t.

6. If $D_t \neq \emptyset$, go to step 4.

7. Turn on the appliances in the admission set O_t, and delay the appliances in $R_t \setminus O_t$.

8. Random transition occurs, and the system moves to period $t + 1$.

The sort order specified by each of the policies is given in Section 2.4.2. Some of the policies are myopic, considering only the state and parameters in the current period, while others use information from future periods, such as electricity prices, to determine the admission set. Since the most computationally intensive operation in this algorithm is sorting, the policies execute extremely quickly, much more quickly than the ADP.

2.4.2 Policies

Below, we give details about how D_t, the set of appliances under consideration to be turned on, is sorted and otherwise modified. Throughout, we assume that ties are broken arbitrarily.

Policy 1 (Random): In this policy, requested appliances are sorted in random order. This policy is straightforward but naive since it overlooks other factors in the system that one may take advantage of to reduce the objective function value.

Policy 2 (Delay-Based): In this policy, admissible appliances are first sorted using one of the following three sorting rules:

(a) Sort the appliances in ascending order of their current states S_{nt}, i.e., the amount of time that they have been delayed. The motivation is that if an appliance has been delayed for a long time, then it should be given priority to turn on as soon as possible.

(b) Sort the appliances in descending order of their unit penalty cost d_{nt}. This sorting method favors turning on appliances whose penalty costs are large.

(c) Sort the appliances according to U_{nt}, where $U_{nt} = S_{nt}d_{nt}$, the product of the two factors.

Each of these three sorting rules measures the degree of urgency of the appliances. Rule (a) treats appliances as urgent if they have been delayed for a long time; (b) treats appliances as urgent if they have large delay penalties; and (c) combines the two approaches. We refer to these as Policies 2(a), 2(b), and 2(c), respectively.

Policy 3 (Price-Based): This policy considers fluctuations in the electricity prices. When the price will be lower in period $t+1$ than it is in period t, it may be advantageous to delay some of the requested appliances. In this policy, we first examine the appliances in D_t. If the savings in electricity cost from delaying an appliance exceeds its delay penalty, this appliance will be delayed (i.e., removed from D_t). The remaining appliances are sorted according to their delay penalties. In particular:

- If $C_t > C_{t+1}$, then for each $n \in D_t$, update D_t as follows: remove n from D_t if $(C_t - C_{t+1})P_n > d_{nt}$ (appliance n will be delayed). Sort the remaining appliances in D_t in descending order of $d_{nt}/(C_t - C_{t+1})P_n$, i.e., of d_{nt}/P_n.
- If $C_t \leq C_{t+1}$, leave D_t intact. Sort the appliances in D_t in one of the three sorting methods discussed for Policy 2.

We refer to these as policies 3(a), 3(b), and 3(c).

2.5 Computational Study

2.5.1 Experimental Design

We tested the ADP algorithm and the policies for a large number of instances to compare their performance. The instances are randomly generated but are meant to model realistic settings. Except as noted, each instance consists of $T = 24$ periods. We generated the parameters for the instances as follows:

1. Electricity usage, P_n: We divide the appliances into two groups, with power consumption levels motived by real appliances [U. S. Department of Energy, 2013]. The first group represents appliances with smaller power consumption such as lighting,

televisions, and computers; for this group, P_n (in kWh) is generated from a $U[0.05, 1]$ distribution. The second group represents appliances with larger power consumption such as water heaters, ovens, and air conditioners; for this group, P_n is generated from $U[1, 5]$. Each appliance is randomly assigned to group 1 with probability 0.8 or to group 2 with probability 0.2.

2. Electricity prices, C_t: Electricity wholesale prices are based on data from NYISO [NYISO, 2012]. For each instance, we set the prices in our 24-hour horizon equal to the hourly wholesale prices from a randomly selected day in 2011. The resulting prices in this sample range from \$0.01 to \$0.05 per kWh. We then added $N(0, \sigma^2)$ random noise to each price, where σ is generated from a $U[0.002, 0.02]$ distribution, i.e., roughly 5% to 50% of the average price.

3. Delay parameters, d_{nt}, $\bar{d}_{nt}$, ℓ_n: For each appliance n, we generated the maximum preferred delay ℓ_n from the discrete distribution $U[2, 8]$. We generated the delay cost d_{nt} (for periods before the maximum preferred delay) from $U[0.02, 0.08]$ and the penalty cost $\bar{d}_{nt}$ (for periods after the maximum preferred delay) from $U[1, 2]$. For a given appliance n, these costs are the same for all t, i.e., we generated a single delay cost and a single penalty cost for each appliance and used them throughout the horizon. Note that, since the electricity usage P_n ranges from 0.05 to 5 kWh and the electricity cost ranges from \$0.01 to \$0.05 per kWh, the cost per period to operate the appliances ranges from \$0.0005 to \$0.25. The range of d_{nt} falls within this range, so that some appliances are worth delaying while others are not. The penalty cost $\bar{d}_{nt}$ is much larger in order to discourage appliances from being delayed beyond their maximum preferred delay.

4. Transition probabilities, a_{nt}, b_{nt}: For each appliance n, we generated $a_n \sim U[0, 0.1]$ and $b_n \sim U[0.6, 0.8]$ and set $a_{nt} = a_n$ and $b_{nt} = b_n$. In order to account for the fact that most appliances have certain times of day during which they are more likely to be used, we added a $U[0, 0.5]$ variate to a randomly chosen 20% of the a_{nt} values. We did not similarly perturb the b_{nt} values since operation time distributions

for most appliances are constant throughout the day. We chose to draw the power limits L_t from a Poisson distribution in order to ensure that some limits were binding while others were very large and therefore non-binding. In particular, we let $\lambda = \sum_{n \in N} P_n$ and then used three distributions for the power limits: $L_t \sim \text{Pois}(\lambda)$, $L_t \sim \text{Pois}(\frac{2}{3}\lambda)$, and $L_t \sim \text{Pois}(\frac{1}{3}\lambda)$. We generated 20 instances (consisting of electricity usage, electricity costs, and delay parameters) and then replicated these instances three times, using each of the power limit distributions.

All computational tests were performed in MATLAB on an Intel Xeon 2.4GHz(x2) with 250 GB of RAM.

2.5.2 Performance of ADP Algorithm

We first tested the performance of the ADP algorithm on a very small instance with $N = 22$ appliances and $T = 3$ periods. The appliances have power consumptions of 1 and 3 kWh. We ran the ADP algorithm using $M = 10$ and found the average cost $f_0(S_0)$ to be 13.23, whereas the true optimal objective value (found by solving the DP exactly) is 13.20, an error of 0.2%. The CPU time to run the ADP algorithm with $M = 10$ was 0.40 seconds.

We then executed the ADP algorithm on a somewhat larger instance with $N = 5$ appliances and $T = 10$ periods. For this larger instance, the parameter M plays an important role in the CPU time and must be tuned well. Figure 2.2 plots the objective function value returned by the ADP algorithm for this instance for various values of M. Note that the objective value increases with M but stabilizes for M greater than 2600. This can be seen as evidence of convergence. (The hope is that the plot converges to the true optimal objective function value, though the true optimal value cannot be computed exactly due to the large instance size.) Another piece of evidence supporting our claim that the algorithm converges as M increases is that the states that have been visited along the sample path in each time period remain largely the same as M increases; for example, roughly 97.8% of the states visisted by the algorithm with $M = 3100$ are also visited when $M = 3200$.

Of course, as M increases, the CPU time increases, as well. Thus there is a tradeoff

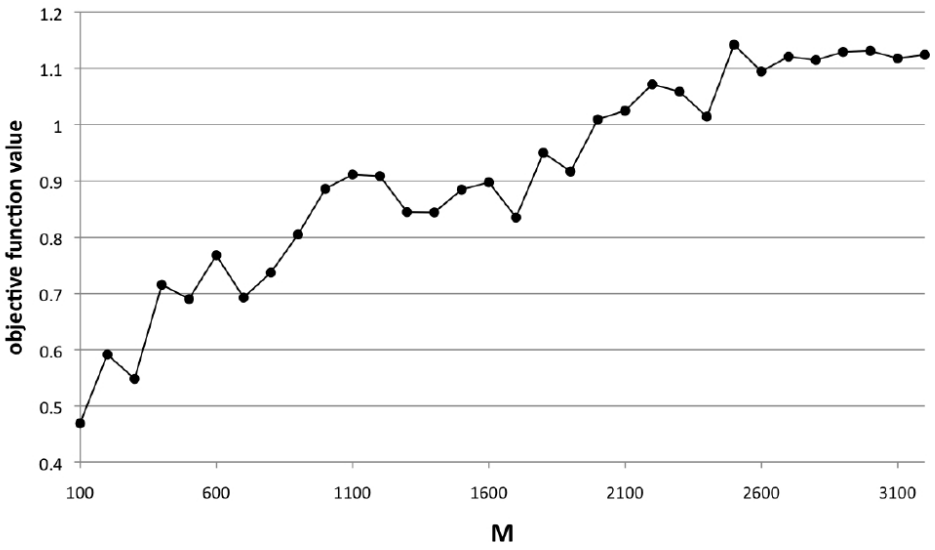


Figure 2.2: Objective function value vs. M

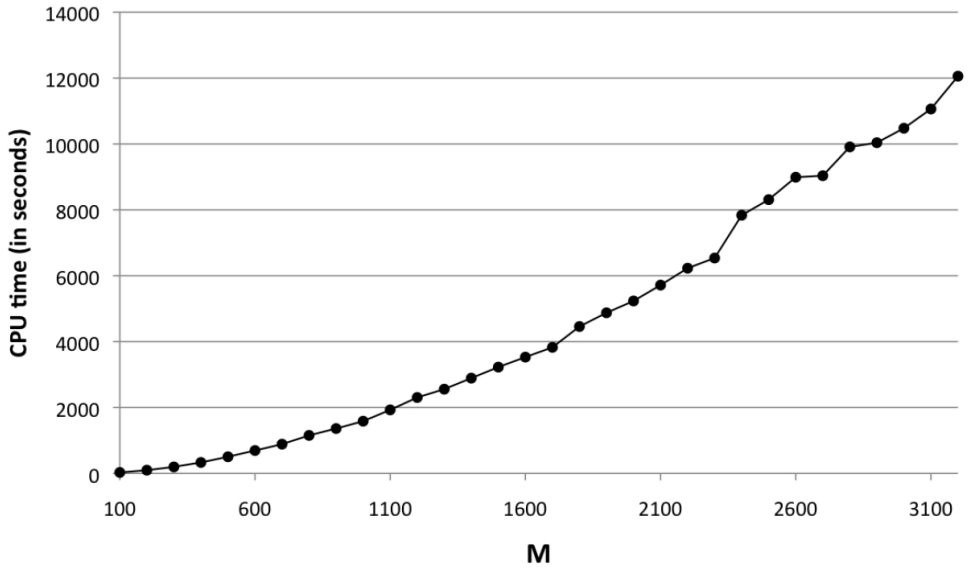


Figure 2.3: CPU time vs. M

between solution quality and CPU time. Unfortunately, CPU times are relatively long if near-optimal results are required. Figure 2.3 plots the CPU time as a function of M. It takes more than 3 hours to solve the instance when $M = 3200$.

Since this instance has only 5 appliances, we would expect the ADP to be much slower for realistically sized instances. Nevertheless, this is still an improvement over the exact DP algorithm of Kishore and Snyder [2010], which cannot solve an instance with

only 3 appliances within 3 hours.

2.5.3 Performance of Policies

For instances with more than a few appliances and a planning horizon of 24 periods or more, the ADP algorithm does not show any evidence of convergence within 30 minutes. This is not acceptable for the time scale on which these problems must be solved in practice. Next, we demonstrate that the policies proposed in Section 2.4 produce good solutions in a fraction of the time.

We first tested Policies 1, 2(a)–(c), and 3(a)–(c) on the previous small instance with $T = 3$; they provided near-optimal solutions (within 0.3% error) with CPU times measured in milliseconds. We then simulated 160 instances of various sizes, with N ranging from 30 to 150. For each instance tested, we simulated the system 10,000 times to estimate the performance of the policies. Each simulation consists of 5 days (120 periods). For each instance, we generated electricity prices for 24 hours and then replicated these over the 5 days. We evaluated the average objective function value for the middle three days, omitting the first and last day to avoid start-up and shut-down effects.

All policies executed very quickly: For each instance, the total time to simulate the 5-day horizon and to execute the policy in each period was less than 0.5 seconds. Figure 2.4 gives a performance profile comparing the various policies. The percentage error is the difference between the objective function value for a given policy and the best objective function value found (which is not necessarily optimal).

To interpret the performance profiles, let i represent the instance solved and p represent the policy used, and let the cost value obtained by solving instance i with policy p be denoted $J(i, p)$. Then the performance profile is generated by the following procedure:

1. For each instance i solved, find the minimum cost among all policies, $J_{\min}(i) = \min_{p \in \mathcal{P}} J(i, p)$. Here, $\mathcal{P}$ represents the set of policies; for our study,

$$\mathcal{P} = \{1, 2(a), 2(b), 2(c), 3(a), 3(b), 3(c)\}.$$

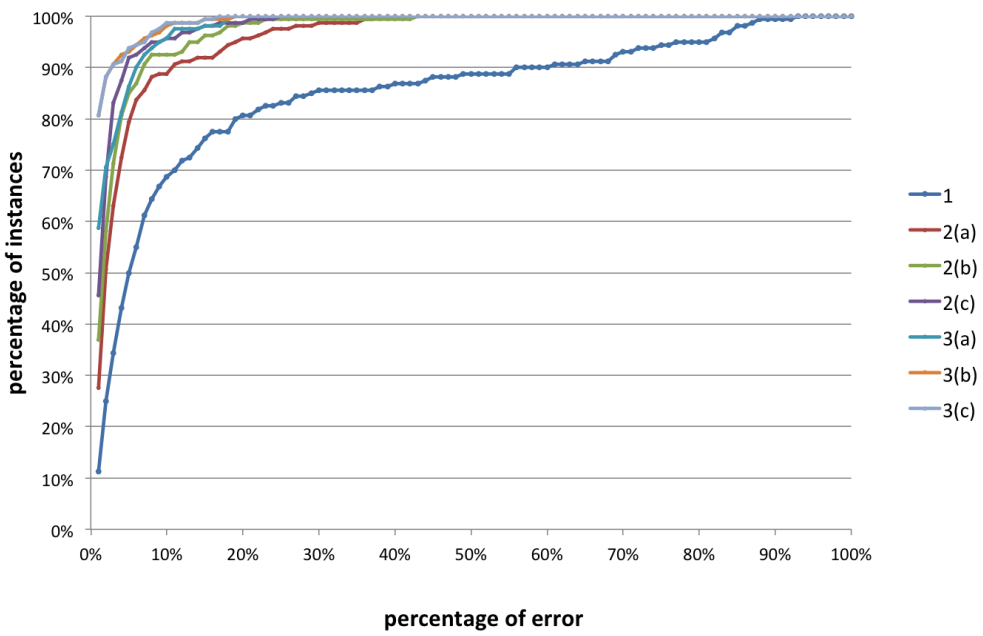


Figure 2.4: Comparison of policies over 160 instances

2. Compute the relative cost difference between each policy and the minimum-cost policy:

$$D(i,p) = \frac{J(i,p) - J_{\min}(i)}{J_{\min}(i)}$$

3. For each policy p, calculate $G(p)$, the percentage of instances for which $D(i,p) \leq \alpha$, for various $\alpha \in [0\%, 100\%]$.

4. Plot $G(p)$ as a function of α.

The plot indicates the percentage of instances that each policy solves to within a given error. For example, Policy 1 solves 50% of instances to within $\sim$ 7% and 90% of instances to within $\sim$ 65%. A policy is better if its curve converges to 100% quickly, i.e., if it is further toward the northwest corner of the plot.

The performance profile indicates that Policies 2 and 3, with any sorting method, significantly outperform Policy 1. This is not surprising since Policy 1 sorts the appliances randomly, while the other policies use smarter sorting methods. Policies 3(b) and 3(c) appear to be the best of the policies considered. We conjecture that this is because Policy

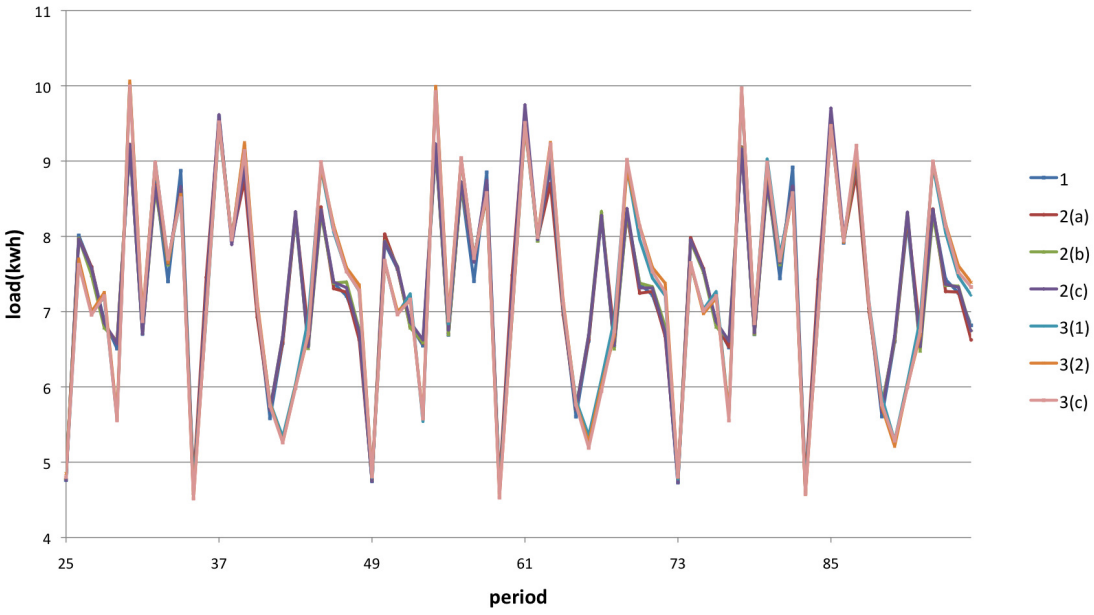


Figure 2.5: Loads generated during middle 3 days of horizon

3 accounts for fluctuations in electricity prices, while Policy 2 does not. To confirm this conjecture, we first examine the loads generated by the policies. We chose one of the 160 instances and simulated it 10,000 times, as described above. Figure 2.5 plots the loads generated by each policy during the middle 3 days of the horizon. Note that, since the electricity prices repeat each day, the loads also display periodicity, with a period of 24 hours. In Figure 2.6, we plot the average hourly load for these 3 days; the electricity price is plotted using the secondary vertical axis. We observe that the policies generate fairly similar average loads for most periods, except during the minimum and maximum prices (periods 6 and 18, respectively). During period 6, when the cost drops to its minimum, Policies 2 and 3 both increase the loads, but Policy 3 does moreso. During period 18, when the price spikes sharply, Policy 3 postpones some consumption to period 19 but Policy 2 does not. This provides an example of Policy 3's ability to react to electricity prices and suggests why it outperforms Policy 2.

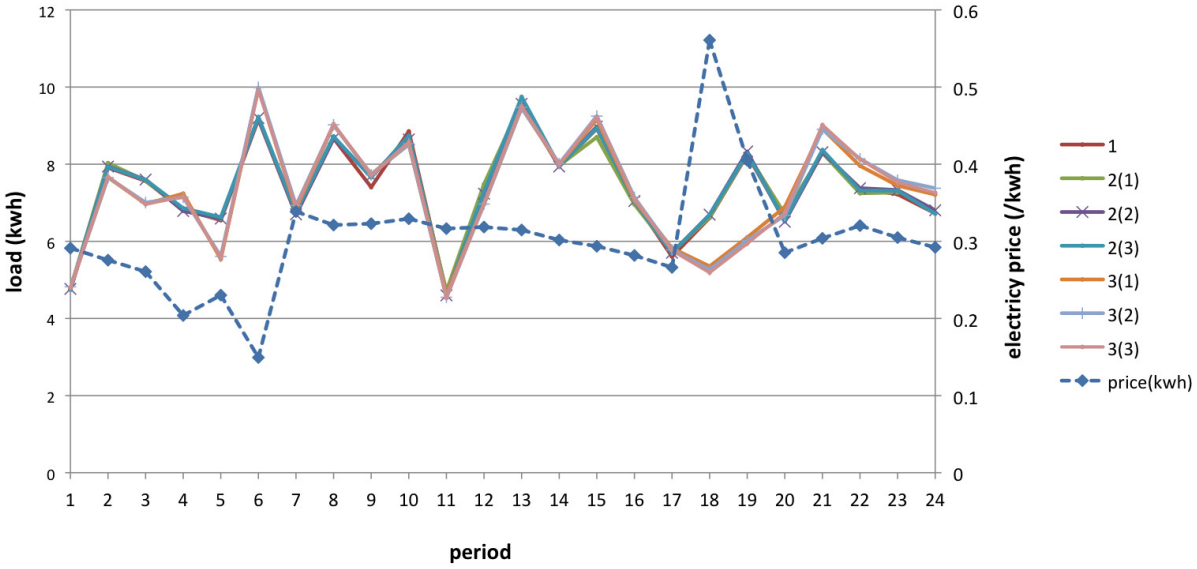


Figure 2.6: Average hourly load for middle 3 days of horizon

2.5.4 Comparison to No-EMC Case

In this section, we demonstrate the benefits of our scheduling policies in terms of both the consumer's electricity cost and the system's peak load. As a benchmark, we use Policy 1 to represent the no-EMC case, since Policy 1 simply turns appliances on as soon as they are requested (choosing them randomly if there is not enough available power).

For the cost savings of the other policies compared to the no-EMC policy (Policy 1), we refer again to the performance profile in Figure 2.4. From the figure it is clear that the cost under Policy 1 tends to be higher than that under the other policies, as evidenced by the fact that the curve for Policy 1 converges to 100% much slower than the other curves. For example, Policies 2(a) and 3(c) result in an average savings of 12.4% and 17.0%, respectively, across all instances tested.

In order to evaluate the reduction in peak load that results from demand response, we modify Policy 1 so that it ignores the power constraint (since otherwise the no-EMC case results in an artificially low peak). We evaluate the reduction in peak load resulting from the EMC policies by executing the policies and the no-EMC algorithm on an illustrative instance. We simulate each policy and the no-EMC policy for 5 days and compute the

Average Peak Load (kW)	
Policy 1	14.10
Policy 2(a)	14.08
Policy 2(b)	14.13
Policy 2(c)	14.09
Policy 3(a)	13.94
Policy 3(b)	13.93
Policy 3(c)	13.93
No-EMC Policy	17.10

Table 2.1: Average peak loads

average peak load over each sample path over the middle 3 days. To be more specific, let l_{ti} be the load in period t for sample path i; then average peak load is

$$\frac{1}{K}\sum_{i=1}^{K}\max_{t\in\{24,\ldots,72\}}\{l_{ti}\},$$

where K is the number of sample paths (we used 10,000).

The results are summarized in Table 2.1, which gives the average peak load for each policy. As expected, the average peak for the no-scheduling algorithm is larger than that of any of the policies. The policies reduce the peak by a minimum of 17.4% (for Policy 2(b)) and a maximum of 18.5% (for Policies 3(b) and (c)). In fact, for any instance in which the power requirement is high, the average peak load of the no-EMC policy will be large since this policy ignores the power limit.

2.5.5 Relationship between Cost and Power Limit

One would expect that the total cost is inversely correlated with the power consumption limit L_t, i.e., for the same instance, as the power limit becomes tighter, the cost tends to increase. This is because more appliances will be forced to postpone execution, resulting in increased delay costs (especially if the maximum allowable delay is exceeded) and possible higher electricity costs if appliances are forced to be delayed until high-cost periods.

To illustrate this, we choose one of the 160 instances and plot the average total cost, over 10,000 simulation trials, for a range of values of L_t. In particular, for each period t

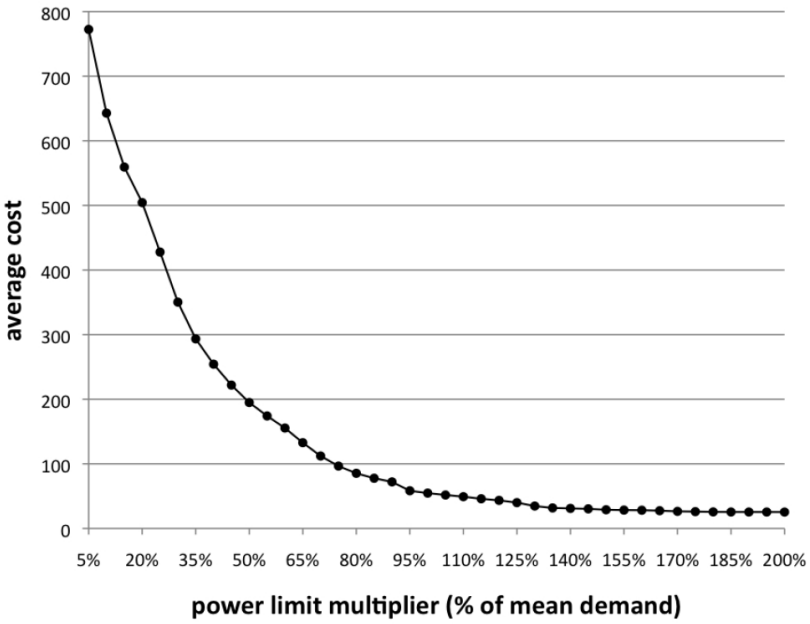


Figure 2.7: Average total cost vs. power limit L_t

we set L_t equal to a certain percentage of the mean demand that would occur in period t if there were no power limit constraint (estimated by simulating the system with $L_t = \infty$). The percentage multiplier ranges from 2% to 200%. We tested each of the policies for each simulation trial and record the best cost found for each trial in an attempt to estimate the optimal cost. This cost is plotted versus L_t in Figure 2.7. The curve decreases sharply at first, indicating that substantial reductions in total cost can be achieved with small increases in L_t, and then decreases much more slowly. Once L_t is greater than 120% of the mean demand, the curve is quite flat, indicating that the power limit constraint is generally non-binding, and further increases in the limit have little effect on cost.

The solutions represented in Figure 2.7 each have a different total cost, but also a different breakdown between the two cost components: electricity consumption cost and delay penalty. In Figure 2.8, we plot the tradeoff between these two cost components. From the figure, one can see that when the power limit is small, then many appliances will be delayed and the delay cost will be large.

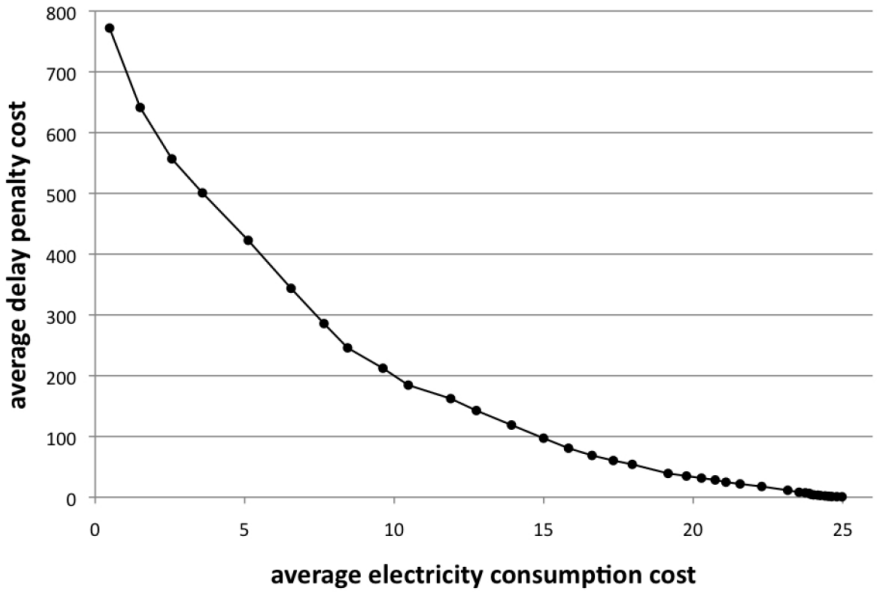


Figure 2.8: Penalty cost vs electricity cost

2.6 Conclusions

In this chapter, we consider approaches for the scheduling of appliances in a demand response program under stochastic appliance timing and a power usage constraint. We first developed an approximate dynamic programming (ADP) algorithm, which works well for small instances but cannot solve instances of realistic size sufficiently quickly. We then proposed several scheduling policies for very fast solutions of large scale problems. Simulation results indicate that the policies execute extremely quickly, and that sorting the requested appliances according to their operating urgency improves the cost. The policy that performs the best, Policy 3, accounts for electricity price information.

2.7 Future Work

Future research on this topic may include improving the minimization problem in (2.1) for given t and S_t, and finding tight lower bounds on the optimal objective function value in order to evaluate the performance of the policies accurately. We are also interested in the design of novel, more accurate policies. Moreover, in order to reduce the data burden on the user, it would be worthwhile to explore policies that require less (or no) information

about delay penalties.

Chapter 3

Power Consumption Optimal Control with a Battery Energy Storage System (BESS) under Demand Charge Tariff

3.1 Introduction

In an electricity network, the end users are charged for the energy they consume (measured in kilowatt hours, abbreviated kWh). Most customers pay for the energy they use, while some larger users are also charged for their peak demand (measured in kilowatts, abbreviated kW). The utility company measures the cost for a power consumer using information provided by an energy meter. Within each 15 minute interval, for example, the instantaneous electricity power usage is being averaged and recorded by the meter for billing calculation purposes. At the end of each month, in addition to the energy charge, the electricity users are also charged for their largest 15-minutes-average demand that happened within the month. Some utilities also charge for demand on a time of use (TOU) basis, in which customers pay for demand that is associated with certain time periods of a billing cycle. For example, a cost is incurred for the peak demand that happens between 12:00pm

and 6:00pm throughout the month.

From a system point of view, the transmission and distribution utilities must prepare sufficient power to be available for their customers. High demand from the customer can incur a heavy burden for the electricity provider, since it costs more in generation and distribution. A demand charge is a billing method to compensate the effort made by the electricity provider to ensure power availability for the end user.

Moreover, a demand charge offers incentives for end users to reduce their peak demand. Given that a certain amount of cost will be incurred for their peak demand, the end users will tend to reduce their peak demand to avoid this demand charge. Some power consumers change their power consumption pattern by shifting part of their tasks from peak hours to off peak hours, resulting in a lower peak demand. Some customers introduce a power buffer that provides power when demand is high. This process is called *peak shaving*. Traditional peak shaving is performed by devices such as diesel generators or gas turbines which can generate instant power quickly. But these generators consume fossil fuels, which can be expensive and polluting. An alternative is to deploy a battery energy storage system (BESS) to perform peak shaving for power consumers. The battery system can offer an energy buffer for the consumption system. It can be charged during off peak hours and discharged to supply power during peak hours, reducing peak demand and saving overall payment. It is important to utilize the battery system properly in order to manage the peak demand.

In reality, the customer's power consumption cannot normally be completely scheduled ahead of time. Moreover, some large customers may deploy renewable generation systems to help reduce energy payments. With the renewable installations, the demand for power from the grid is measured by the difference between the load and the renewables, which is called the net load (we call it *load* for simplicity in the chapter). Renewables are known for their randomness. This challenges the planner to come up with efficient schemes to plan under uncertainty.

The problem discussed in this chapter considers medium to large power consumers (such as industrial facilities, schools or governments) which incur a charge on their peak

demand from the utility company in addition to the energy cost. Their power demands are non-deferrable and stochastic but can be forecasted by a predictor.

In Section 3.4, we first propose a minimax dynamic programming algorithm that makes decisions for the battery system about how much to charge/discharge assuming the load is deterministic. This algorithm is straightforward, but it does not apply to problems in which the load is stochastic. We then introduce a novel dynamic programming algorithm in Section 3.6 to optimally solve the problem when the load is stochastic. Later in Section 3.7 we consider the practical case, in which the planner needs to make decisions in real time, and moreover, the system should be capable of adapting to new inputs over time. Therefore, we introduce a sample average approximation (SAA) algorithm that solves the problem in an efficient manner, and provides near-optimal solutions. After that, in Section 3.8 we develop three naive algorithms, which provide good solutions quickly, and we compare the results from those algorithms with the results from the DP and SAA algorithms. Then we simulate the real time battery power dispatch scenario: the mean and standard deviation of the load constantly gets updated by the predictor over time, and the system needs to dispatch battery power with limited knowledge about the future. We develop a real time SAA A lgorithm and study its performance on the practical problem.

3.2 Literature Review

From an economic standpoint, the customer needs to plan her power usage carefully according to the utility company's electricity tariff in order to reduce the electricity cost. In this section we discuss the literature related to methods for optimizing power consumption.

Many power consumers take actions to optimize their electricity costs. There are several ways to hedge against high costs; one effective way is to shift some of the deferrable load from peak hours to off-peak hours, so that the load profile will be distributed more evenly, and the total energy payment will be reduced. This is called demand side management (or demand response). There exists a significant amount of work on demand side management, see Su and Kirschen [2009], Conejo et al. [2010], Shao et al. [2010], Palensky and Dietrich [2011], Li et al. [2011b], Subramanian et al. [2012] and Gan et al. [2013].

However, sometimes the load is not flexible. For example, a farm needs power to water the crops during certain times of the day, otherwise they will dry out quickly. A laboratory needs a constant supply of power to keep the equipment functioning well for experiments. These loads cannot be shifted. In cases like this, a storage system can be deployed to perform two tasks in order to reduce payments for the customers: energy optimization and peak shaving. Energy optimization uses an energy storage system as a buffer, charging the system when the energy cost is low, and discharging the system to supply the load when the energy cost is high. There is an extensive literature about energy management for non-deferable load profiles, of which Kraning et al. [2011], de Ven et al. [2011], Codemo et al. [2013], and Huang et al. [2012] are most closely related to our work. Kraning et al. [2011] first develop a real time battery charge/discharge operating management system in response to fluctuations in the prices with a given demand portfolio. Then the authors configure a portfolio for the energy devices. Both of the problems can be approximately solved using convex optimization. de Ven et al. [2011] develop a storage management policy for customers in a retail energy market where the energy price fluctuates. The objective of the problem is minimizing the cost of energy storage. The author formulates this problem with dynamic programming and shows the optimal policy has a threshold structure. Codemo et al. [2013] study the impact of battery capacity, charge/discharge rate, and cost function on the cost saving that can be achieved by some selected energy storage algorithms. Then the author shows that the optimal storage strategy has a threshold structure under some assumptions about the cost function. Huang et al. [2012] develop a power control policy for power dispatching in order to achieve cost efficiency with storage. They propose a low-complexity demand response algorithm for power dispatch and battery sizing. These papers adopt a battery storage system to reduce the power purchased from the grid when the energy price is high. They assume the customer participates in the real time market where the energy price fluctuates over time.

While all of this research is complementary to our work, there is one major difference from our problem setting: These works focus on energy management under price fluctuations. However, for our problem, the energy charge specified by the utility company for

the end users is mostly flat. Even if there is a time-of-use (TOU) energy charge, the energy rates between peak hours and off-peak hours are relatively close. For example, according to the tariff specified by PG&E (PGE [2013]), during off peak hours, the price is 7 \$/kWh, whereas during peak hours the energy price is 10 \$/kWh. If a 100 kWh load is shifted from a peak hour to an off peak hour, the savings is \$3. On the other hand, the demand charge typically is of a different order of magnitude, mostly ranging from \$8/kW to \$16/kW. Suppose the demand charge is \$10/kW; if the battery reduces the peak grid purchase level by 100 kW over 1 hour, which is equivalent to shifting 100 kWh of energy, then the cost savings will be \$1000. It's clear that the savings that can be achieved in the energy cost is limited compared to potential demand charge savings, which motivates the focus on demand charges in our models.

Another way of reducing energy payments is through peak shaving. Here, the battery is deployed to reduce the peak grid purchase over the planning horizon. There has been much work on developing efficient control mechanisms for peak shaving tasks using battery systems as a buffer. Even et al. [1993] discusses the feasibility and economic aspects of using batteries for peak shaving tasks. Oudalov et al. [2006] perform a value analysis for peak shaving using a battery energy storage system, and analyze when peak shaving is profitable/un-profitable taking into consideration the battery costs. Our work mainly focuses on the optimization algorithm for the offline/online problems, in which offline means planning the battery power dispatch ahead of time, and online refers to planning in real time. The economic aspects of the battery such as system cash flow, pay back time, internal rate of return (IRR), special policies, etc., differ on a case by case basis and therefore need to be post-processed.

Maly and Kwan [1995], Oudalov et al. [2007] Koutsopoulos et al. [2011] and Dong et al. [2012] formulate dynamic programming models to perform peak shaving tasks for customers. These four papers assume the load profile is deterministic. However, during real time implementation, the system only sees a forecast of the load and the forecasting error needs to be properly addressed. The objective of these four papers is peak shaving although they have some slight variation. The major difference versus our work is that

none of them considers a stochastic load profile.

In this chapter, our primary focus is to reduce the customer's demand charge (under the rate specified by the utility company) when the load is stochastic. The reasons are as follows: first, the difference in energy cost between peak and off peak periods (1-3 $/kWh) is much smaller compared to the demand charge (8-20 $/kW), and therefore the benefit of energy optimization is of a different magnitude compared to peak shaving for most of load profiles. Second, during real time implementation, if an unexpectedly high load occurs when the battery is empty, then the system will fail the peak shaving task. The battery should really reserve its power for peak shaving tasks instead of performing energy optimization.

In this regard, our work is similar to that of Johnson et al. [2011], who introduce a peak shaving problem in which the customer is charged for its peak demand. The authors consider two scenarios, offline (deterministic) and online (random). For the online case, the authors propose an algorithm which assumes knowledge of the peak demand. But in reality the magnitude and timing of the peak demand is often not known beforehand. Also, the authors assume no charge and discharge limits for the battery, which are essential features of a battery. Our model accounts for these important factors.

Our objective function minimizes the maximum grid purchase over a specified horizon. In the deterministic case, this problem resembles the minimax path problem. We apply the minimax path dynamic programming algorithm to this problem in addition to linear programming algorithms. The study of the dynamic programming formulation provides valuable insights for the stochastic load case. In the stochastic setting, this problem is a multistage stochastic optimization problem. The cost function needs to be written in an additive form in order to be solved by dynamic programming (Bertsekas [1995]). But the demand charge problem is not additive and therefore cannot be solved by DP directly. We therefore introduce a novel DP algorithm that reformulates the cost function in additive form, and solves the stochastic demand charge problem to optimality.

The DP algorithm is proven to solve problem to optimality; however, it may be computationally time consuming. Especially during real time implementation, where the

load forecast constantly changes over time, planners need to make decisions quickly. We therefore introduce a sample average approximation algorithm that solves the problem in an iterative fashion yet provides near-optimal solutions. See Linderoth et al. [2006] and Shapiro [2001] for results concerning the theoretical upper bound, lower bound, and convergence complexity of the SAA Algorithm.

3.3 Notation

Throughout this chapter, we adopt the following notations

1. $t \in 1, \cdots, T$: the time index of the problem; T is the planning horizon; the time interval between t and $t+1$ is Δt.
2. C : battery capacity (in kWh).
3. s_t : the battery state of energy (SOE), which captures the amount of energy in the battery at time t, ranging from $0\%C$(empty) to $100\%C$(full).
4. $P\max$: battery's discharge power limit (in kW). ($-P\max$: battery charge power limit).
5. u_t : the battery charge/discharge power (in kW); $u_t < 0$ implies energy is being charged into the battery; $u_t > 0$ implies the battery discharges to supply the load; else the battery is not being used.
6. $\mathcal{U} = (u_1, \cdots, u_T)^T$: the vectors of charge/discharge decisions from time 1 to time T.
7. l_t : the load observation at time t.
8. L_t : the random variable representing the load at time t.
9. ξ_t : a realization of the random load L_t.
10. $g_t = l_t - u_t$: the grid purchase level at time t.

Note that we differentiate between a load *observation* l_t, which represents an event that actually occurs and that we can observe, and a load *realization* ξ_t, which is a sample

drawn from a scenario space but does not necessarily occur. We refer to observations when discussing the DP approach (since it responds to actual events) and to realizations when discussing the SAA approach (since it optimizes based on samples, not observations).

3.4 Deterministic Load Profile

Suppose the load l_t for $t = 1, \cdots, T$ is deterministic. The goal is to minimize the overall demand charge. Assume a demand charge d (\$/kW) is incurred on the peak grid purchase over the time horizon, and the grid purchase is the difference between the load and the battery action, namely, $g_t = l_t - u_t$. Note here we do not restrict the grid purchase level to only take non-negative values. Instead, we assume the customer can sell back to the grid when there is excess power. The value of g_t is decided by the optimization algorithm. Let J be the optimal objective value of the problem. The optimization problem is given below:

$$J = \min_{\mathcal{U}} \left\{ d \cdot \max_{t=1,\cdots,T} (l_t - u_t) \mid \text{some battery constraints} \right\} \tag{3.1}$$

3.4.1 Battery Constraints

The system needs to satisfy a set of battery constraints. Depending on the battery type, the battery performance can be different. For example, some batteries can be charged and discharged faster than others, producing sufficient instantaneous power for the system. Some batteries have high capacity that can constantly produce power over a long time. Another important characteristic of the battery is the charge and discharge efficiency. The battery efficiency captures the loss during charge and discharge processes. It is the ratio θ between the terminal power and the internal power, $0 < \theta \leq 1$. The internal power (IP) measures the amount of power that the battery obtains/releases during a charge/discharge processes, and the terminal power (TP) is the actual amount of power that is available to be consumed at the terminal. To be more specific, $\theta = IP/TP$ during charging and $\theta = TP/IP$ during discharging. This value is determined not only by the battery type but also by the battery state of energy (SOE). The algorithms and computational study we developed in this chapter are based on the assumption that $\theta = 1$ for simplicity. This

assumption can be relaxed easily in the analysis below, with only minor modifications required to the models and algorithms.

In general, a battery is characterized by its capacity and charge/discharge rate limit. For example, a 100 kW 2 hour battery means that the battery can provide maximum of 100 kW power over a 2 hour period before it is depleted, and the capacity of the battery is 200 kWh. Suppose $\theta = 1$, and the battery starts with a SOE of s_1; then the capacity and charge/discharge rate constraints will be:

$$0 \leq s_1 - \sum_{i=1}^{t} u_i \Delta t \leq C, \ \forall t \in 1, \cdots, T \tag{3.2}$$

$$-P\max \leq u_t \leq P\max, \ \forall t \in 1, \cdots, T \tag{3.3}$$

Here C is the capacity of the battery (in this example $C = 200$ kWh), and $P\max$, $-P\max$ are the discharge and discharge rate of the battery, respectively ($P\max = 100$ kW).

3.4.2 Minimax Route Dynamic Programming Algorithm

The system needs to solve the following optimization problem:

$$\begin{aligned} & J = \min_{\mathcal{U}} \left\{ d \cdot \max_{t=1,\cdots T} (l_t - u_t)) \right\} \\ s.t. \qquad & 0 \leq s_1 - \sum_{i=1}^{t} u_i \Delta t \leq C, \qquad \forall t \in 1, \cdots, T \\ & -P\max \leq u_t \leq P\max, \qquad \forall t \in 1, \cdots, T \end{aligned} \tag{3.4}$$

When the load profile is deterministic, this can be solved efficiently by several optimization algorithms, e.g., interior point method. The objective is nonlinear due to the $\max(\cdot)$ function. However, this can be linearized and then treated as a linear programming problem. However, these optimization algorithms do not apply when the load profile is stochastic. Therefore, we are interested in finding an algorithm that solves the problem to optimality, yet provides some insight for the stochastic case. In particular dynamic programming algorithm can handle both deterministic and stochastic problems. Let U_t be the set of feasible

charge/discharge actions at time t:.

$$U_t = \{u_t : 0 \leq s_t - u_t \Delta t \leq C,\ -P\max \leq u_t \leq P\max\},$$

equivalently,

$$U_t = [-\min\{P\max, (C - s_t)/\Delta t\}, \min\{P\max, s_t/\Delta t\}].$$

Then the recursive equation for the deterministic demand charge problem is:

$$J_t(s_t) = \min_{u_t \in U_t} \{d \cdot \max(l_t - u_t, J_{t+1}(s_{t+1} \mid u_t))\} \tag{3.5}$$

where $J_t(s_t)$ is the cost of being at state s_t, given the system acts optimally thereafter. The state update function is $s_{t+1} \mid u_t = s_t - u_t \Delta t$, and $s_t = s_1 - \sum_{i=1}^{t-1} u_i \Delta t$.

This problem resembles the minimax route problem (Berman and Handler [1987]). The minimax route problem considers a directed network with one source and one sink; and the maximum length of any edge in the route is minimized. In this case the grid purchase level $l_t - u_t$ represents the length of each edge in the network.

Figure 3.1 gives a demonstration of how to construct a minimax route network for the demand charge problem. Suppose the starting state of the battery is s_1, the planning horizon is $T = 2$, and there are only two possible charge/discharge actions at each state. The network diagram shows that at beginning of time 1, the battery state is s_1, and there are two possible charge/discharge actions; after each action, the battery SOE gets updated and the system moves to the next state s_{21} (corresponding to action u_1) and s_{22} (corresponding to action u_2). At time $t = 2$, the same procedure repeats. s_{3i} represents the battery termination state. D is a dummy sink node, and we assign 0 length to each edge that connects a termination state with the dummy sink node.

This problem can be solved using the minimax route dynamic programming (DP) algorithm (Berman and Handler [1987]). For optimization purposes, the system needs to keep track of the state of the system s_t which summarizes the past information. The

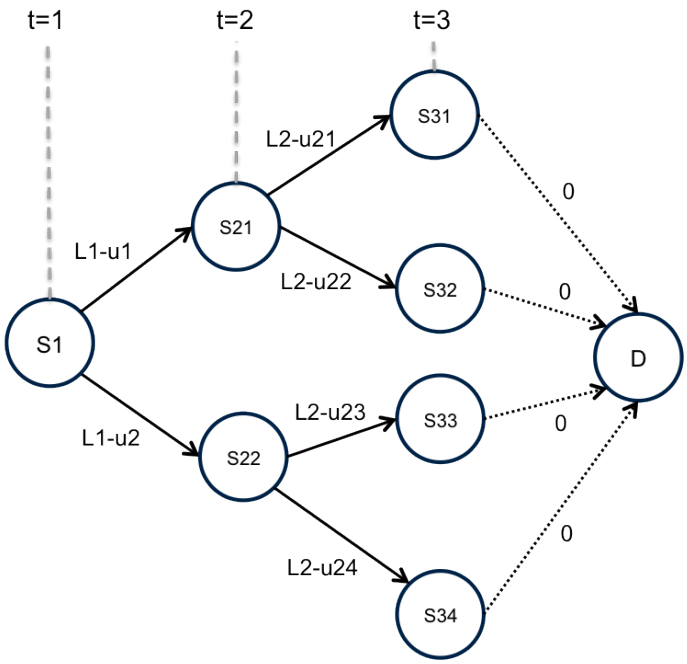


Figure 3.1: A demonstration of deterministic case

dynamic programming algorithm uses backward induction. During implementation, it maintains two separate lists. The rows of the first list account for all possible states s_t of the system, and the columns of the list accounts for all time periods from $t = 1$ to $t = T + 1$, where the last period is a dummy period. Each cell of the list stores the cost of being in the current state $J_t(s_t)$, that is, the total cost incurred from time t to $T + 1$, assuming the system acts optimally. After this, we create a second list, which has the same row and column structure but each cell stores the optimal decision u_t^* that corresponds to the cost $J_t(s_t)$. Based on to the minimax route DP algorithm, the deterministic demand charge algorithm is given in algorithm 1:

Let U_t be the feasible region at time t in state s_t, and let u_t be an action from the feasible region; then

$$u_t \in U_t = \left[-\min\left\{P\max, (C - s_t)/\Delta t\right\}, \min\left\{P\max, s_t/\Delta t\right\}\right].$$

Algorithm 1 DP algorithm (Deterministic)

1: Let $t = T + 1$, $J_{T+1}(s_{T+1}) = 0 \ \forall s_{t+1}$.

2: According to the Bellman equation, at any given battery SOE state s_t, for any action u_t within the feasible region, evaluate:

$$\bar{J}_t(s_t, u_t) = \max\{l_t - u_t, J_{t+1}(s_t - u_t \Delta t)\}$$

3: Find u_t^* that minimizes the cost, i.e.,

$$u_t^* = \arg\min_{u_t \in U_t} \bar{J}_t(s_t, u_t), \quad \text{and} \quad J_t(s_t) = \min_{u_t \in U_t} \{\bar{J}_t(s_t, u_t)\}$$

Record the optimal action u_t^* in the action list and the optimal cost $J_t(s_t)$ in the cost list corresponding to each state.

4: If $t = 1$, STOP. Otherwise, $t = t - 1$; go to 2.

3.4.3 Deterministic Case Study

Here we study a sample problem. All of the computational tests were done in Matlab 10.8 with an Intel(R) Xeon(R) CPU, 3.20GHz processor (RAM 12GB). We create a test problem with a planning horizon of one day, 24 hours with 1 hour resolution. The parameters are summarized as follows:

1. planning horizon: $T = 24$, time resolution $\Delta t = 1$ hour;
2. demand charge: $d = 1$, demand charge is incurred on the peak purchase over 24 hours;
3. battery size: 50 kW, 2 hour battery: $P\max = 50$ kW, $C = 100$ kWh;
4. battery efficiency: $\theta = 1$;
5. Net load as shown in Figure 3.2.

Discretization of the Problem

The dynamic programming algorithm is based on a set of discretizations of the state space and the action space. The discretization has a great impact on the performance of the algorithm. Throughout this thesis, we use the notation $\underline{z} : \Delta : \bar{z}$ to represent a discretization of the range $[\underline{z}, \bar{z}]$ into equal periods of size Δ. For this instance, we choose the following

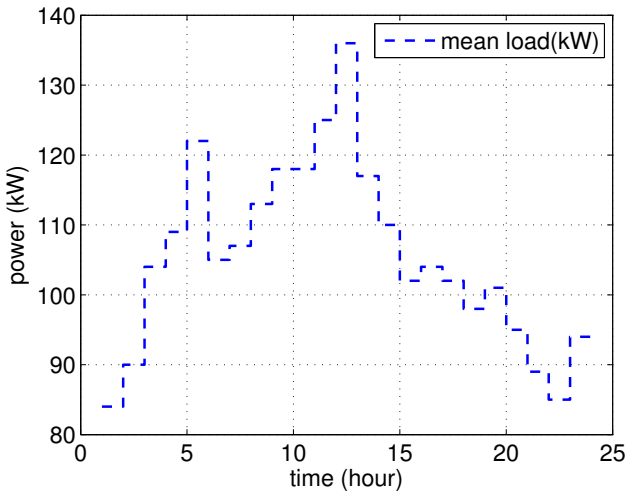


Figure 3.2: Net load values

discretizations: The state space assumes values from $\underline{z} = 0$ to $\bar{z} = 100$, discretized into 201 different values, i.e., $\Delta = 0.5$.

At any given state s_t, based on the battery charge/discharge rate and capacity constraint, we discretize the action space into 40 different values:

$$\Delta u = [\min\{P\max, s_t/\Delta t\} + \min\{P\max, s_t/\Delta t\}]/40;$$

therefore;

$$u_t \in [-\min\{P\max, (C - s_t)/\Delta t\} : \Delta u : \min\{P\max, (C - s_t)/\Delta t\}].$$

3.4.4 Computational Performance of Deterministic Case Study

The CPU time for the DP algorithm under the specified discretization is 50.8 seconds. Figure 3.3 shows the load and the grid purchase level after applying the DP algorithm. The maximum grid purchase over 24 hours is 111.47 kW. For comparison, we also solved the problem using an interior point algorithm in Matlab 10.8 using *fmincon* with an optimality tolerance of 5×10^{-3}; the cost of the resulting solution is 111.38, and the CPU time is 6.8 seconds. The difference is less than 0.1%. The maximum grid purchase from the DP algorithm is slightly higher due to the discretization. The maximum grid purchase over

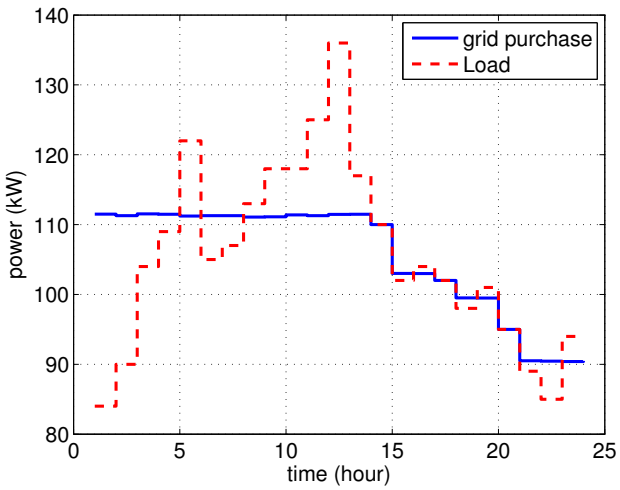


Figure 3.3: Result of case study

24 hours is 111.38 with the DP algorithm under a discretization of $0 : 0.2 : 100$. However the CPU time is 380.8 seconds.

Although the DP algorithm cannot compete with other algorithms in terms of solution time, it still provides insight about how to solve the problem optimally in an incremental fashion, which is essential as a starting point for our algorithm for problems in which the load is stochastic.

3.5 Stochastic Load Profile

Now assume that the loads are stochastic. To be more specific, assume the random loads L_t for $t = 1, \cdots, T$ are independent and normally distributed with known mean and variance, $L_t \sim N(\mu_t, \sigma_t^2)$. In practice, we normally observe that the demand in the current time period and the next are correlated. However, it's reasonable to assume that the load is independent over time for the following reason: In practice, the load will be predicted by a forecasting method. Ideally, such a method will make very accurate predictions, and any temporal correlation among load values will be captured by the forecasted mean values. The remaining forecasting errors are typically independent across time. Therefore, the forecasting process mitigates any temporal correlation, allowing us to treat the loads as independent.

In each time period t, the events occur in the following sequence:

1. the system observes the demand l_t and the battery SOE s_t;
2. the planner makes battery charge/discharge decision u_t based on some optimization algorithm;
3. the system executes the battery action u_t, and updates the battery SOE state as $s_{t+1} = s_t - u_t \Delta t$.

This sequence of events assumes that the system observes a demand request first and then chooses an action based on the observation and the optimization result; this is known as *a posteriori*.

Along the same lines as the deterministic formulation (3.4), in the stochastic case, the goal is to find battery charge/discharge values such that the expected maximum grid purchase is minimized.

$$J = \min_{\mathcal{U}} \left\{ E\left(d \cdot \max_{t=1,\cdots,T} (L_t - u_t) \right) \mid \text{constraints } (3.2), (3.3) \right\} \tag{3.6}$$

3.6 Dynamic Programming Algorithm

3.6.1 Introduction to Dynamic Programming Algorithm

From the previous section it is clear that the biggest challenge for this problem is that the cost is not additive, which is a prerequisite for stochastic dynamic programming models. The question is how to transform this problem into one with an additive cost.

Consider the following structure: Suppose the demand charge is incurred at the end of the time horizon, $t = T$, but still based on the maximum grid purchase over the whole horizon, and there is no charge during the rest of the time periods. This structure allows an additive cost form: The incremental cost is 0 in periods $t = 1, \cdots, T$, while the cost is $d \cdot \max_{t=1\cdots T} (l_t - u_t)$ at the end of the horizon. In order to accommodate this cost structure, the system not only needs to keep track of the SOE of the battery, but also has to keep track of the maximum grid power purchase in $t = 1, \cdots, T$.

Let p_t denote the maximum grid purchase from period 1 through $t-1$, i.e,

$$p_t = \max\{g_1, \cdots, g_{t-1}\},$$

where $g_t = l_t - u_t$ represents the grid purchase level, the difference between the load and the battery charge/discharge. In this case, the state is a two dimensional vector: $x_t = (s_t, p_t)$. Under this definition, the state update functions are as the follows:

$$\begin{aligned} s_{t+1} &= s_t - u_t \\ p_{t+1} &= \max(p_t, l_t - u_t) \\ x_{t+1} &= (s_{t+1}, p_{t+1}) \end{aligned} \tag{3.7}$$

After the planner has made the decision of how much to charge/discharge the battery, the next SOE is determined uniquely, since the SOE does not depend on the random load. The peak grid purchase value will be updated according to the purchasing history and the current grid purchase level. According to the state update function, if the current grid purchase exceeds the peak purchase in the past, then the current state purchase determines the demand charge, otherwise the peak to date stays the same. Given that L_t is random, p_{t+1} is defined not only by the decision itself, but also by the realization l_t of the random load L_t at time t.

The demand charge is incurred at the end of the planning horizon after the peak grid purchase is realized. We define a dummy period $T+1$ for the demand charge calculation, and define $J_{T+1}(x_{T+1}) = d \cdot p_{T+1}$.

The Bellman equation for battery optimization for the a priori case is as follows:

$$\begin{aligned} J_{T+1}(x_{T+1}) &= d \cdot p_{T+1} \\ J_t(x_t) &= \min_{u_t} \{0 + E_{L_t}(J_{t+1}(x_{t+1}|u_t)) \mid \text{constraints (3.2)(3.3)}\} \quad \forall t = 1, \cdots, T \end{aligned} \tag{3.8}$$

$E(J_{t+1}(x_{t+1}|u_t))$ represents the expected future cost given the current action u_t, and

$$x_{t+1}|u_t = (s_t - u_t, \max(p_t, l_t - u_t)).$$

The dynamic programming algorithm uses backward induction. The goal is to find a battery power dispatch policy, or more specifically, for each possible state that occurs, the battery charge/discharge value that minimizes the expected cost. To achieve that, we construct two tables, one that stores the optimal actions, and another that stores the cost associated with each action. The procedure is given in Algorithm 2.

Algorithm 2 DP algorithm (A priori)

1: Let $t = T + 1$, $J_{T+1}(x_{T+1}) = dp_{T+1}$, and $u^*_{T+1} = \emptyset$.
2: For every state $x_t = (s_t, p_t)$, determine the feasible charge discharge range:

$$U_t : [-\min\{P\max, (C - s_t)/\Delta t\}, \min\{P\max, s_t/\Delta t\}].$$

3: For any $u_t \in U_t$, compute the cost incurred by charging/discharging u_t amount of power:

$$\bar{J}_t(x_t, u_t) = E_{L_t}(J_{t+1}(x_{t+1}|u_t))$$

4: Find u^*_t that minimizes the cost:

$$u^*_t = \arg\min_{u_t \in U_t}\{\bar{J}_t(x_t, u_t)\}, \text{and } J_t(x_t) = \min_{u_t \in U_t}\{\bar{J}_t(x_t, u_t)\}$$

5: Fill in the cell $(t, (s_t, p_t))$ of the action list with the best action u^*_t, fill in the cell $(t, (s_t, p_t))$ of the cost list with the optimal cost $J_t(x_t)$.
6: If $t = 1$, STOP; else let $t = t - 1$ and go to 2. Update the battery SOE state: $s_{t+1} = s_t - u_t\Delta t$.

The algorithm walks backward in time. At the end of the planning horizon, the maximum grid purchase is realized and the demand charge is incurred. From time $t = T$ down to $t = 1$, we compute the cost of being in any state using the Bellman equation to find the best action u^* among all possible actions. First, according to the state and the battery constraints, a range of possible actions is computed. This range makes sure the battery charge/discharge level is feasible according to the battery capacity charge/discharge rate constraints. For any possible u_t, we compute the cost incurred for making that action. According to the state update relationship in (3.7), the future cost can be computed as

follows:

$$
\begin{aligned}
&E(J_{t+1}(x_{t+1}|u_t)) \\
&= E(J_{t+1}(s_t - u_t, \max\{p_t, L_t - u_t\})) \\
&= \textstyle\int_{u_t+p_t}^{+\infty} J_{t+1}(s_t - u_t, l - u_t) f(l)\,\mathrm{d}l \qquad + \int_{-\infty}^{u_t+p_t} J_{t+1}(s_t - u_t, p_t) f(l)\,\mathrm{d}l \\
&= \textstyle\int_{u_t+p_t}^{+\infty} J_{t+1}(s_t - u_t, l - u_t) f(l)\,\mathrm{d}l \qquad + J_{t+1}(s_t - u_t, p_t) F(u_t + p_t)
\end{aligned}
\tag{3.9}
$$

Here $f(\cdot)$ and $F(\cdot)$ are the probability density function and the cumulative density function of the random load L_t. The future cost can be computed exactly using (3.9).

This algorithm is similar to the DP for the deterministic load setting. The major difference between this algorithm and the previous minimax DP algorithm is that the demand charge is incurred at the end of the time horizon, after the all the demands and actions are realized. Therefore the cost becomes additive. The tradeoff is that we introduce an additional dimension to the state space, p_t, to keep track of the grid purchase level from the past.

There are some considerations during the implementation of the stochastic DP algorithm. First, discretization of the action space is needed in this case. How to discretize the action space depends on the problem itself; we give an example in section 3.4.2. We define the discretization of this DP problem to be $(\Delta_s, \Delta_p, \Delta_u)$, where $\Delta_s, \Delta_p, \Delta_u$ represent the number of possible battery SOE states, number of possible peak purchase levels and number of possible actions, respectively. The SOE and battery action states are bounded; here we assume that the peak grid purchase is bounded above by $P\max + \max_{t=1\cdots T}\{\xi_t + 3\sigma_t\}$ and bounded below by 0. Second, according to the definition, the state space consists of all combinations of all possible battery SOE values and all possible peak values, so the number of possible states at time t is $\Delta_s \times \Delta_p$. The SOE is bounded by the battery capacity (between 0 and C). But the peak grid purchase value is uncertain. In this case, we have to limit the peak purchase level to only take certain values.

3.6.2 Stochastic Load Case Study

This case study inherits the same problem setting as in Section 3.4.3 expect the load is stochastic. Assume the mean load is as shown in Figure 3.2, and the standard deviation of the load in any time period is 15 kW.

This case study is designed to have two phases. Phase 1 is the optimization phase. During this phase, the system solves the optimization problem using the DP algorithm with the inputs and assumptions provided in Section 3.6. After execution, the algorithm will provide two lists recording the optimal cost and the optimal action corresponding to each possible state. Phase 2 is called the execution phase. In order to study the performance of the dynamic programming algorithm, we create 1000 random test instances (load trajectories) from the same distribution ($\xi^i = (\xi^i_1, \cdots, \xi^i_{24})^T$, $i = 1, \cdots, 1000$ and $L^i_t \sim N(\mu_t, \sigma^2_t)$). For each realization, we execute according to the DP solution. Then we study the mean and standard deviation of the demand charge over these 1000 instances. The mean objective reflects on average how much demand charge will be incurred if we solve the problem using the DP algorithm. The standard deviation represents the volatility of the cost. It is desirable to have stable costs.

It is our expectation that the a posteriori case will be associated with better system performance, for the system has one additional load piece of information on hand before the action is taken, and therefore the decision can be better. Unfortunately, however, to solve the problem in the a posteriori case exactly with DP, we need to introduce one additional state variable, l_t, representing the realized load in period t, and the state vector becomes (s_t, p_t, l_t).

In this case the state update function is:

$$s_{t+1} = s_t - u_t \Delta t,$$

$$p_{t+1} = \max\{p_t, l_t - u_t\}$$

$$L_{t+1} \sim N(\mu_{t+1}, \sigma^2_{t+1}).$$

The Bellman equation is:

$$J_T(s_T, p_T, l_T) = \min_{u \in U_T} \max\{p_T, l_T - u_T\}$$

$$J_t(s_t, p_t, l_t) = \min_{u \in U_t} \left\{0 + E_{L_{t+1}}(J_{t+1}(s_t - u_t \Delta t, \max\{p_t, l_t - u\}, L_{t+1}))\right\}$$

The downside of this problem is that the state vector has 3 dimensions. The CPU time will increase sharply as the state dimension increases.

Instead, we propose an approximate method. The basic idea is to use the a priori DP algorithm in the optimization phase and then adjust the resulting solutions during the execution phase by solving an additional single-variable optimization problem for each (s_t, p_t). In particular, for given (s_t, p_t, l_t), we find the u_t that optimizes the a priori cost function in which the expectation over L_t degenerates to the single value using l_t. This approach is not exact since the calculation of the cost resulting from each possible action u_t assumes that in future periods we will choose actions before observing loads, when in fact this is not the case. However, the hope is that the resulting solution is not far from optimal. Algorithm 3 includes the execution phase of the a posteriori case.

Algorithm 3 Execution phase of approximate algorithm for a posteriori case

1: Start from time $t = T$ to time $t = 1$, observe load l_t. For any state $x_t = (s_t, p_t)$, evaluate the feasible charge discharge range U_t.
2: Solve the optimization problem to obtain the optimal action u_t^*:

$$J_t(s_t, p_t) = \min_{u_t \in U_t} \left\{\hat{J}_{t+1}(s_t - u_t \Delta t, \max\{L_t - u_t, p_t\})\right\}$$

where $\hat{J}_{t+1}(x_{t+1})$ can be obtained from the optimal cost table of the a priori case;
3: Take the action u_t^*, the system moves to post decision state: $x_{t+1} = (s_{t+1}, p_{t+1})$.

3.6.3 Computational Performance of Demand Charge Algorithm - A posteriori

We implement this case study using Matlab 10.8. It is important to select a proper discretization $\delta = (\delta_s, \delta_p, \delta_u)$. But this is not a trivial task. For this problem, we do the following: choose the discretization to be $\delta = (\delta_s, \delta_p, \delta_u) = (a, a, \frac{1}{2}a)$, where a takes values

from 10 to 100. The action values were discounted by $\frac{1}{2}$ since the battery system is a 2 hour battery. In this case the SOE takes values from 0 to 100, but the charge/discharge actions only take values from 0 to 50.

To test the performance, we randomly generated 1000 instances (trajectories) from the scenario space, e.g., for $i = 1 \cdots N$, we generated $\{\xi_1^i, \cdots, \xi_T^i\}$, where ξ_t is a realization of the random load $L_t \sim N(\mu_t, \sigma_t)$. In this case, for any given discretization δ, and any instance i, we run the approximate DP-Aposteriori algorithm for the execution phase, and the cost we obtain for this instance is denoted as $v_{\delta,i}$. The mean cost and the standard deviation of the cost for a given discretization is $\mu_\delta = \mu(v_{\delta,i})$, $\sigma_\delta = \sigma(v_{\delta,i})$.

The relative error of the mean cost for a given discretization δ' is defined as

$$\frac{\mu_{\delta'}}{\min_\delta \mu_\delta},$$

the ratio between the mean cost from 1000 test instances and the minimum mean cost among all discretizations. Similarly, the relative error of the standard deviation is defined as

$$\frac{\sigma_{\delta'}}{\min_\delta \sigma_\delta}.$$

Figure 3.4 plots the CPU time as a function of the relative error of the standard deviation. The plot indicates the tradeoff between solution volatility and time efficiency.

The plot in Figure 3.4 shows that the CPU time increases as the relative error decreases. As the relative error decreases to < 2.5, the solution time increases significantly. For example, the difference in relative error between $a = 90$ and $a = 100$ is 1.2%, but the difference in solution time is 3700 sec.

Now we examine the tradeoff between mean cost and time efficiency. Figure 3.5 plots the CPU time as a function of the relative error of the mean cost. The general trend of this plot indicates that it takes more CPU time to obtain a solution with smaller relative error. But the trend is not monotone. One explanation is that the DP algorithm for the a posterior case is not exact.

After examining the mean cost and standard deviation we set the discretization to be

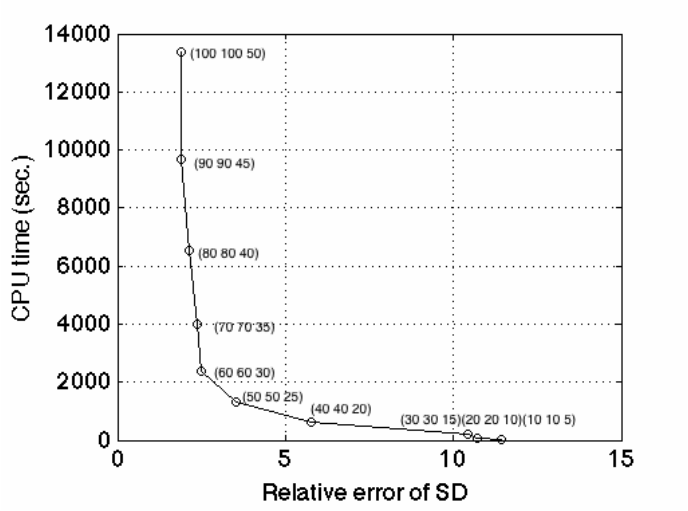


Figure 3.4: CPU time vs. relative error of standard deviation

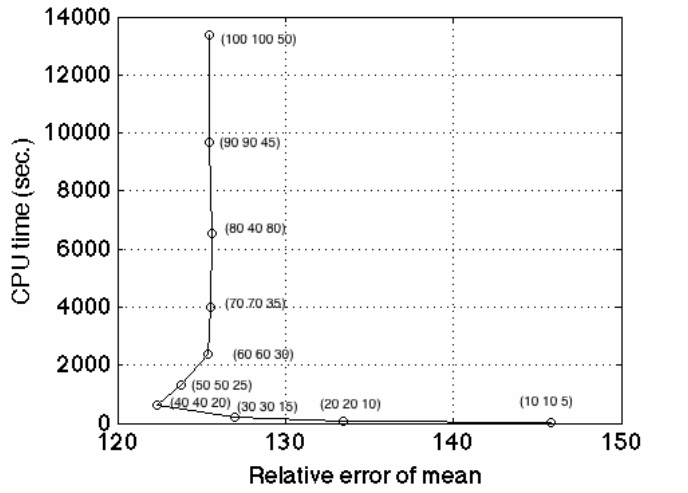


Figure 3.5: CPU time vs. relative error of mean

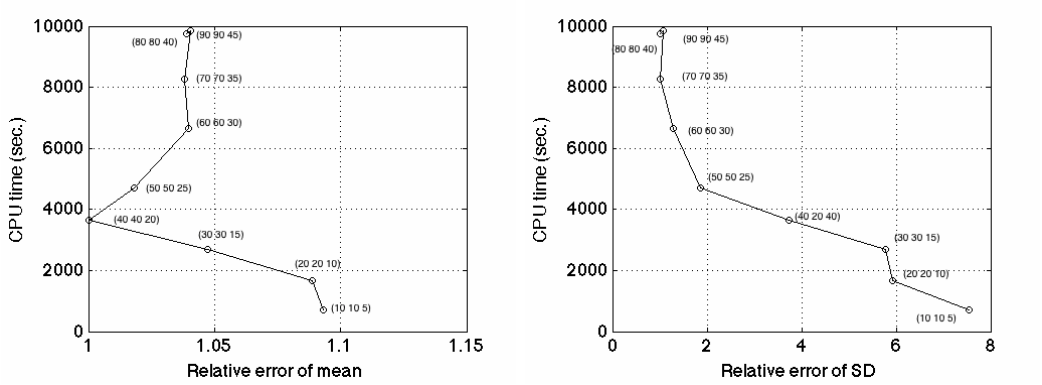


Figure 3.6: CPU time vs. relative error - fixed δ_s, δ_u

$a = 90$, where the mean cost is 125.48 and standard deviation of the cost is 1.88, and the solution time is 9669.64 seconds. Up to now we have allowed the discretization of battery SOE, action, and peak grid purchase to change simultaneously. Next we fix two of the discretization levels, and allow one of them to change at a time.

First we set $\delta = (\delta_s, \delta_p, \delta_u) = (90, a, 45)$, letting a take values from 10 to 90; the discretization of the battery SOE level and the action space are fixed. Figure 3.6 shows the relationship between the discretization of the peak purchase level and the solution quality. In Figure 3.6, the standard deviation of the cost decreases while the CPU time increases.

Similarly, we set $\delta = (\delta_s, \delta_p, \delta_u) = (a, 90, 45)$, letting a take values from 10 to 90. Figure 3.7 shows the relationship between the discretization of the battery SOE level and the solution quality. Compared to Figure 3.6, the relative errors of standard deviation are smaller. This indicates that the system does not need a very fine discretization for the battery SOE state in order to achieve a good solution.

According to these results, for this instance, considering the CPU time and the solution quality, we choose $\delta = (\delta_s, \delta_p, \delta_u) = (20, 90, 45)$ to be the best discretization level, where the mean cost is 125.11 and the standard deviation of the cost is 2.29. The CPU time for this discretization is 1673 seconds. Note the discretization is designed only for this instance; the discretization can be different case by case. Figure 3.8 shows the maximum grid purchase level for the first 100 instances out of the 1000 instances using the no-battery case for comparison. Note that the demand charge is \$1/kW; therefore the maximum grid purchase represents the demand charge. The red curve represents the no battery case,

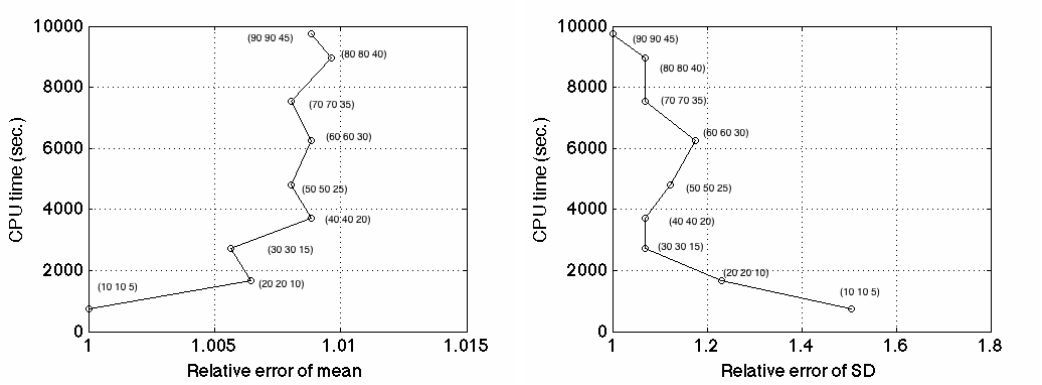


Figure 3.7: CPU time vs. relative error - fixed δ_p, δ_u

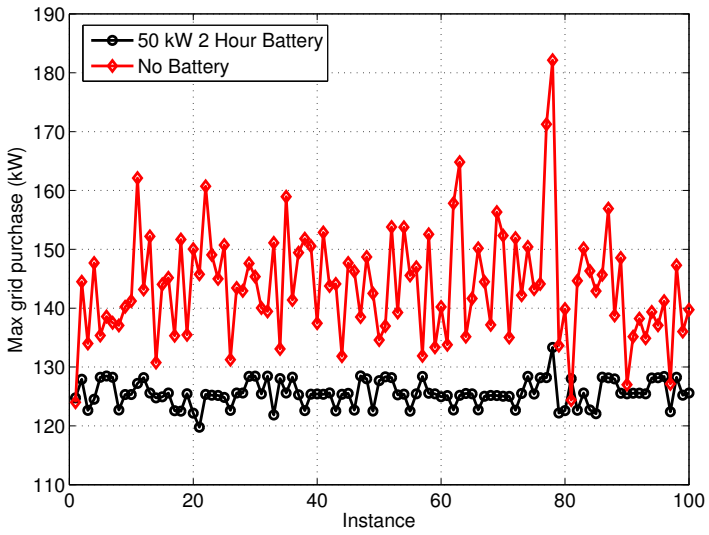


Figure 3.8: Maximum grid purchase for fist 100 instances, DP algorithm.

where the grid purchase value equals the load itself. The black curve represents the demand charge using a battery as a buffer. It is clear that with the battery, the system can reduce the maximum grid purchase. In 98% in the 100 instances, the system achieves a lower demand charge with the battery, with a 13.8% reduction of the peak demand on average.

It is also obvious from the figure that the variance of the maximum grid purchase is smaller with a battery compared to the no-battery case. The red curve oscillates, reflecting differences in demand charges from one case to the other, whereas the black curve appears to be more stable. This is evidence that the battery properly shaves off the peak demand.

Let us take a closer look at two individual instances. Figure 3.9 shows the battery performance for instance number 11. The dashed stair plot represents the mean load, and the red stair plot represents the realization of the load. It can be seen that for some periods, the load realization is quite different from its mean, e.g., in period 8, the difference between the power realization and mean is greater than $2\sigma = 30$ kW. But the dynamic programming algorithm accounts for the distribution of the load and uses the battery properly to shave off the peak. In Figure 3.10, we plot the battery SOE level throughout the planning horizon. The plot shows that, starting with an empty battery, the battery is charged when the load is low and discharged when the load is high.

There are also cases, although rare, where the DP algorithm is very close to or even worse than the no-battery case. In this 1000-instance testing set, instances 1 and 81 are the only two such examples. The plot of the grid purchase for instance 8 in Figure 3.11 reveals that the actual maximum grid purchase during the 24 hour period is greater than the maximum load. The reason for this phenomenon is as follows. At the beginning of the time horizon, the system sees the realized load for period 1 only, along with the information about the future load distribution, and the system decides to charge the battery to prepare for the upcoming peak load around period 12, in which the mean load is the highest. However, the peak load didn't occur around period 12. Instead, the realized load for period 12 is much lower than its mean, and the battery did nothing. As a result, the demand charge was set up by the first few periods — period 4, more specifically. Clearly, there is a tradeoff between acting greedily and conservatively. Since the future can only be predicted but never foreseen, the system will prepare to hedge against cases which are more likely to happen. The same explanation can be applied instance 1.

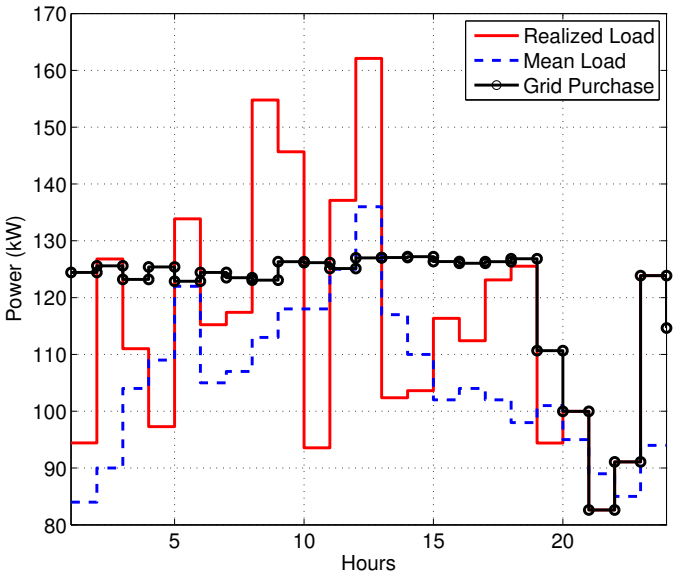


Figure 3.9: Grid purchase for instance 11, DP algorithm

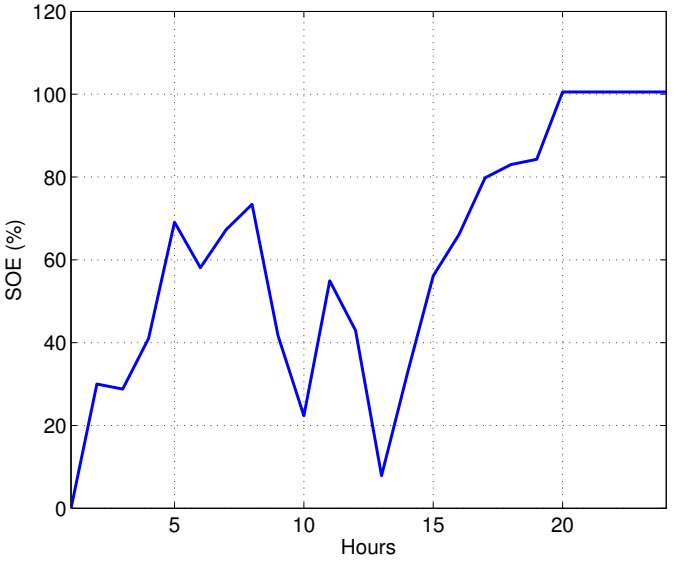


Figure 3.10: Battery SOE for instance 11, DP algorithm

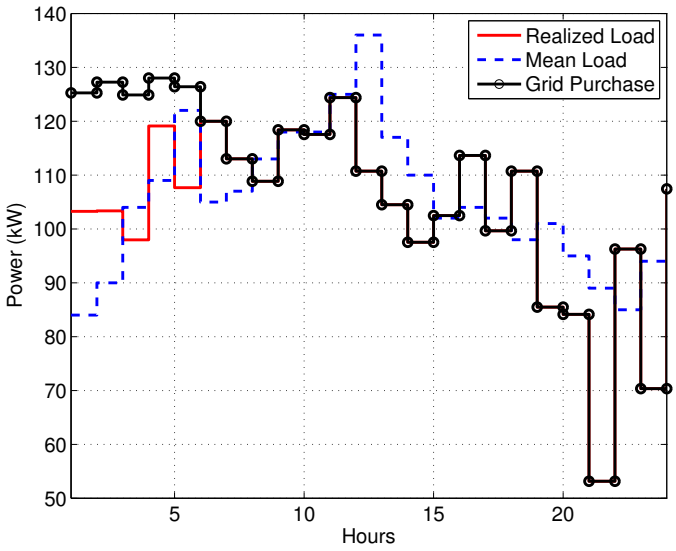


Figure 3.11: Grid purchase for instance 81, DP algorithm

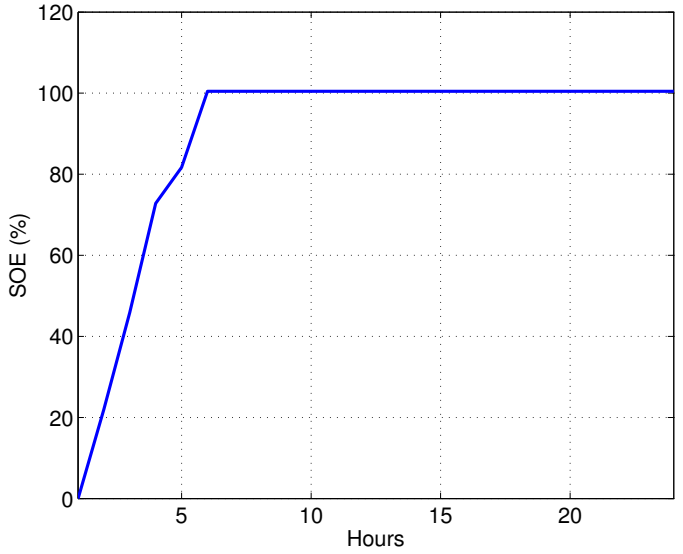


Figure 3.12: Battery SOE for instance 81, DP algorithm

3.7 Sample Average Approximation Algorithm for Stochastic Loads

3.7.1 Introduction to SAA Algorithm

The dynamic programming algorithm solves the problem to optimality. However, the traditional dynamic programming algorithm computes the cost of being in all possible states, which takes a long time to solve if the state space is large. In real world planning, this is not acceptable. For one reason, during real time implementation, the forecast of the mean and the variance of the load constantly changes over time, and therefore the planning tool has to re-solve the DP problem every time with updated inputs. In our preliminary testing, we observed that the DP implementation will require up to 2000 seconds to achieve a desired result for a 24 hour problem with 1 hour resolution. Moreover, according to the tariff defined by many utility companies, the resolution time should be 15 minutes during real world planning. In this case, it will take approximately $4 \times 2000 = 8000$ seconds, which is not acceptable.

Sample Average Approximation (SAA) is a method based on drawing a sample $\{\xi^1, \xi^2, \cdots, \xi^N\}$ from the scenario space and using these samples to approximate the stochastic objective in the optimization problem. The objective is achieved by solving a deterministic equivalent optimization problem based on the sample. This method can be applied to the demand charge problem. The sample average approximation algorithm for the demand charge problem is:

1. Generate N random trajectories $\{\xi^1, \xi^2, \cdots, \xi^N\}$; each trajectory is generated according to the mean and the variance of the load: $\xi^i = \{\xi_1^i, \xi_2^i, \cdots, \xi_T^i\}$, where each ξ_t^i is a realization of the random variable $L_t^i \sim N(\mu_t, \sigma_t^2)$.

2. Solve the optimization problem:

$$J = \min_{u \in U} \left\{ \hat{f}_N(u) := N^{-1} \sum_{i=1}^{N} \max_{t \in \{1, \cdots, T\}} (\xi_t^i - u_t) \right\}, \tag{3.10}$$

where $\hat{f}_N(u)$ is a sample average approximation (SAA) of the objective function $f(u)$.

U is the feasible region defined by the battery charge/discharge rate constraints and capacity constraints (3.2) and (3.3), and $u = (u_1, \cdots, u_T)^T$.

We wish to study the performance of SAA for the demand charge problem. An important issue here is to understand how close the objective J that we get from solving the SAA problem (from equation (3.10)) with a set of randomly generated sample paths is to the true objective value namely, w^*. The objective of this problem w^* cannot be derived analytically, but a lower and upper bound on the gap between J and w^* can be found in [Linderoth et al., 2006].

3.7.2 Computational Performance of SAA Algorithm

It is important to see how the demand charge SAA algorithm performs compared to the dynamic programming algorithm. In order to do this, we test the performance of the SAA algorithm on the same case study defined in Section 3.6.2.

Unlike the dynamic programming implementation procedure, the sample average approximation algorithm only has one phase, which includes both optimization and execution. At the beginning of each time period, the system observes the demand first, then plans based on the SAA algorithm and then executes the actions suggested by the optimization result. This procedure is repeated for every time period. A complete scheme of the implementation is given in Algorithm 4.

The algorithm uses the following notation for period t:

1. ξ_t^i is a realization of the random variable $L_t \sim N(\mu_t, \sigma_t^2)$.

2. $\bar{u}_j$ is the action at time $j < t$; the bar above indicates that the action has already been executed.

Algorithm 4 shows that at every time instant, the SAA algorithm will solve an optimization problem based on the observations of the load in the past and a set of samples generated with given mean and variance of the future. Also note the number of decision variables decreases as t increases, whereas the actual planning horizon T stays fixed as the time period t increases.

Algorithm 4 Real time SAA algorithm

1: At the beginning of time t, observe battery SOE level s_t and load l_t.
2: Solve the optimization problem to obtain the optimal action u_t:

$$f(n) = \left\{ \begin{array}{c} J_t = \min\limits_{U_t} \left\{ \hat{f}_N(u) \right\} \\ s.t.\ 0 \leq s_t - \sum\limits_{i=t}^{j} u_i \Delta t \leq C,\ \forall j \in 1, \cdots, T \\ -P\max \leq u_j \leq P\max,\ \forall j \in 1, \cdots, T \end{array} \right\} \tag{3.11}$$

where $\hat{f}_N(u) := \frac{1}{N} \sum_{i=1}^{N} \max(l_1 - \bar{u}_1, \cdots, l_{t-1} - \bar{u}_{t-1}, l_t - u_t, \xi_{t+1}^i - u_{t+1}, \xi_T^i - u_T)$.
3: Execute the action u_t; update battery SOE $s_{t+1} = s_t - u_t \Delta t$.
4: If $t < T$, set $t = t + 1$ and return to 1, otherwise stop.

Figure 3.13 plots a summary of the first 100 instances out of the 1000 instances. The overall performance is very similar to the DP algorithm. The overall mean cost over these 100 instances of the SAA algorithm is 124.4989 with standard deviation is 1.59, compared to the DP algorithm, which has mean 125.4782 and standard deviation 1.8797. The average CPU time of the SAA algorithm is 2.37 seconds. The SAA algorithm and DP algorithm produce very similar results. The slight differences are due to:

1. the effect of discretization in the DP algorithm;

2. the effect of random sampling in the SAA algorithm.

Figures 3.14 and 3.15 show the peak shaving and battery SOE of instance 11 using the SAA algorithm. The battery is charged at the beginning of the horizon and discharged during peak hours to reduce the maximum grid purchase. The maximum grid purchase is 127.0508 compared to 127.2160, which is the grid purchase from the DP algorithm. Figures 3.16 and 3.17 show the performance of instance 81. Similar to the DP algorithm from this instance, the maximum grid purchase occurs at the beginning of the horizon, and the realized load during peak hours is actually much lower than predicted.

3.8 Naive Algorithms

It is important to study the performance of the DP algorithm and the SAA algorithm in comparison with simple approaches to justify how effective both algorithms are. These

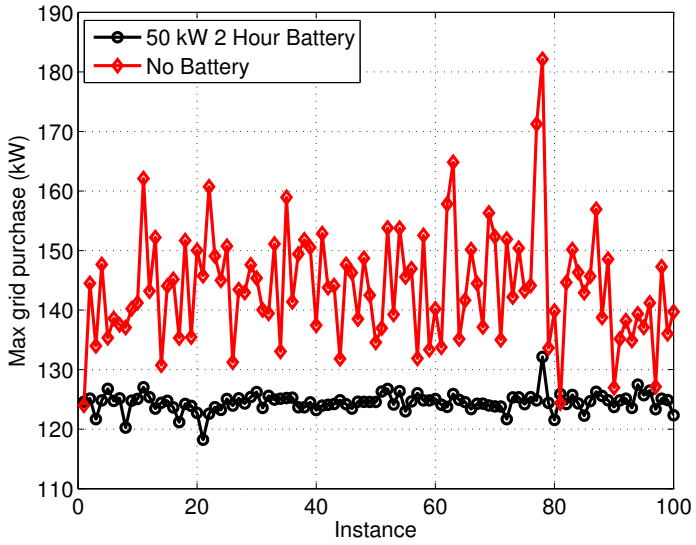


Figure 3.13: Maximum grid purchase for first 100 instances, SAA algorithm

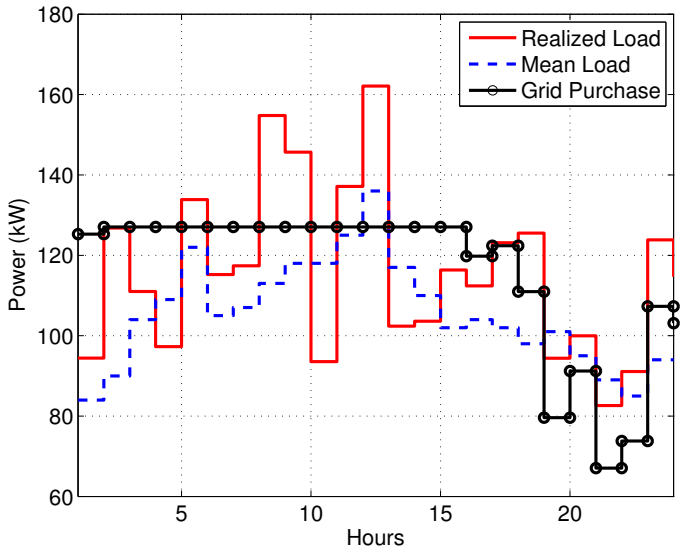


Figure 3.14: Grid purchase for instance 11, SAA algorithm

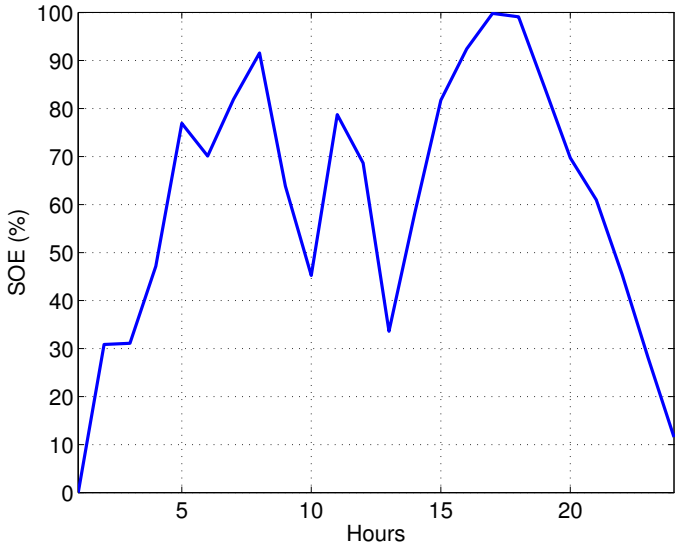


Figure 3.15: Battery SOE for instance 11, SAA algorithm

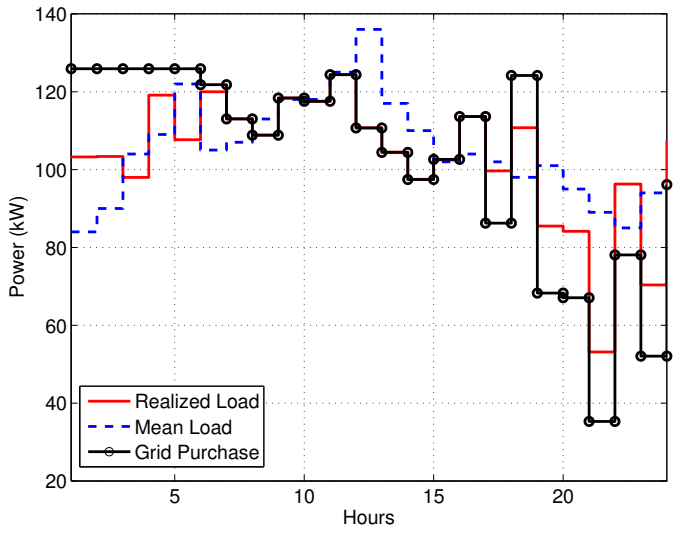


Figure 3.16: Grid purchase for instance 81, SAA algorithm

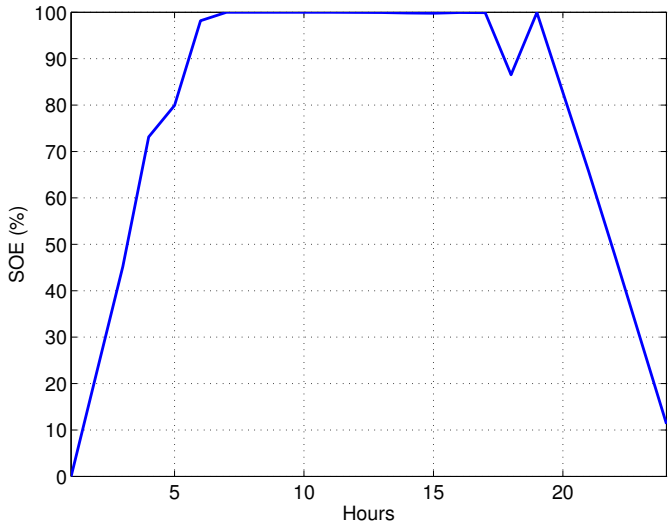


Figure 3.17: Battery SOE for instance 81, SAA algorithm

approaches, which we call "naive algorithms," are meant to approximate the decision-making process that a real planner might use, in the absence of the optimization algorithms proposed here. The naive algorithms should be straightforward and fast, and we need to compare the performance of these algorithms with the performance of the DP and SAA algorithms. In the sections that follow, we introduce three naive algorithms.

3.8.1 Naive Algorithm 1

Description

Imagine that the planner chooses a threshold η at which the peak can be reduced and executes according to this reference. During implementation, if the realized load is greater than the threshold η, then we discharge the battery to bring down the grid purchase as close to η as possible subject to the battery constraints. If the realized load is lower than the threshold, then we charge the battery to bring up the grid purchase as close to η as possible while respecting the battery constraints. This method is straightforward, and takes little time to execute at every time instant. There can be many ways to obtain a proper threshold. One way is straightforward: suppose the planner solves an optimization problem with the mean load value only, then selects the threshold value to be the maximum

purchase level based on the mean load. Below is the complete structure of Naive Algorithm 1:

Naive Algorithm 1 aims to search for a threshold defined by

$$\eta = \max_{t\in\{1,\cdots,T\}} \{E(L_t) - u_t^*\}, \tag{3.12}$$

where u_t^* is the solution to the optimization problem:

$$\min \left\{ \max_{t\in\{1,\cdots,T\}} \{E(L_t) - u_t\} \,|\text{constraints (3.2), (3.3)} \right\}. \tag{3.13}$$

Algorithm 5 describes the implementation of Naive Algorithm 1.

Algorithm 5 Naive Algorithm 1

1: At time 0, observe the battery SOE level. Solve the optimization problem (3.13) to obtain the threshold value η in equation (3.12).
2: At the beginning of time t, observe the battery SOE level s_t and load l_t:
If $L_t < \eta$, charge the battery up to η:

$$u_t = -\min\{\eta - l_t, P\max, (C - s_t)/\Delta t\};$$

else if $L_t > \eta$, discharge the battery down to η:

$$u_t = \min\{l_t - \eta, P\max, s_t/\Delta t\};$$

else do nothing.
3: Execute the action u_t; update the battery SOE $s_{t+1} = s_t - u_t\Delta t$.
4: If $t < T$, set $t = t + 1$ and return to 1, otherwise stop.

Computational Performance of Naive Algorithm 1

We test the performance of the Naive Algorithm 1 on the same case study defined in Section 3.6.2. The results of this algorithm are summarized in Figure 3.18. The overall mean and standard deviation of the objective function are 125.13 and 14.71, respectively. The mean value is comparable with the mean from the DP algorithm and the SAA algorithm. But the standard deviation is much higher compared to 1.59 from the SAA algorithm and 2.23 from the DP algorithm. This can be seen from Figure 3.18, in which for some instances the objective function is remarkably lower than the other two algorithms; for example, instance

22. Figures 3.23 and 3.24 plot the grid purchase and battery SOE for this instance. For this instance the peak grid purchase from using Naive Algorithm 1 is 111.41 kW, whereas the peak grid purchase from using DP and SAA algorithm are 125.34 and 122.61, respectively. On the other hand, many instances have higher objective values; for example, instance 11. Figures 3.19 and 3.20 plot the grid purchase and battery SOE for this instance. The peak purchase level from Naive Algorithm 1 is 162.10, and 127.22 compare to 127.05 for the DP and SAA algorithms.

Now we explain why Naive Algorithm 1 does worse than the DP and SAA algorithms in instance 11. The DP algorithm models the trade off between shaving the current grid purchase level lower with the battery and saving battery power to perform peak shaving in the future. Similarly, the SAA algorithm approximates possible scenarios and then makes decisions based on those scenarios. They both consider the tradeoff given that the load is random. On the other hand, Naive Algorithm 1 solves an optimization problem based on the mean load only and then executes strictly according to the threshold. The randomness of the load is not considered in the optimization problem. For instance 11, the threshold according to the mean is 111.38; if the realized load is exactly the same as the mean load, then the threshold value is optimal. However, it can be observed from Figure 3.19 that the realized loads from time 1 to 13 are in general higher than the mean load, which means the battery will try to reduce the grid purchase to the threshold, with the consequence that the battery hardly charges. In period 13, the peak happens, but at that moment the battery is empty. Therefore in this instance the battery does not make any contribution. For instance 22, it can be observed from Figure 3.23 that from time 1 to 13 (where the peak happens), there are a few periods in which the realized load is significantly lower than the mean load, which enables the battery to be almost charged before time 13. In hour 13, the peak load is much higher than the mean load, so the battery discharges the maximum power, 50 kW, which is the upper bound on the discharge amount. In contrast, if we refer to Figures 3.9 and 3.14, the battery is acting more conservatively for the DP and SAA algorithms since randomness is considered.

The benefit of Naive Algorithm 1 is obvious: once the threshold value is calculated

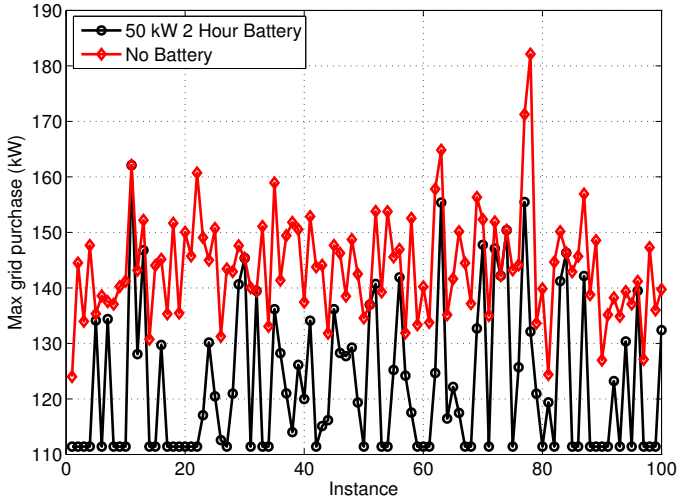


Figure 3.18: Maximum grid purchase for first 100 instances, Naive Algorithm 1

at the beginning of the implementation, the system does not need to re-optimize at each time period, and therefore it can execute extremely fast. Moreover, the average objective value (over 1000 instances) is very close to the values from the DP and SAA algorithms. The downside of this algorithm is also clear: the standard deviation is high, which means the results are not consistent. This is a serious drawback to the system. Imagine that for one month, the algorithm does well in terms of peak shaving and, as a result, the payment for demand charge is low. However, the next month, the system performs very badly and the demand charge is much higher. This is not acceptable from the customer's point of view, as their payments vary dramatically from month to month. For better performance the system needs to provide a more consistent result.

3.8.2 Naive Algorithm 2

Description

One possible drawback of Naive Algorithm 1 is that the load observations have no impact on the threshold value. Now suppose the planner makes use of the observations when determining the threshold values. Assume he develops an algorithm in which the threshold

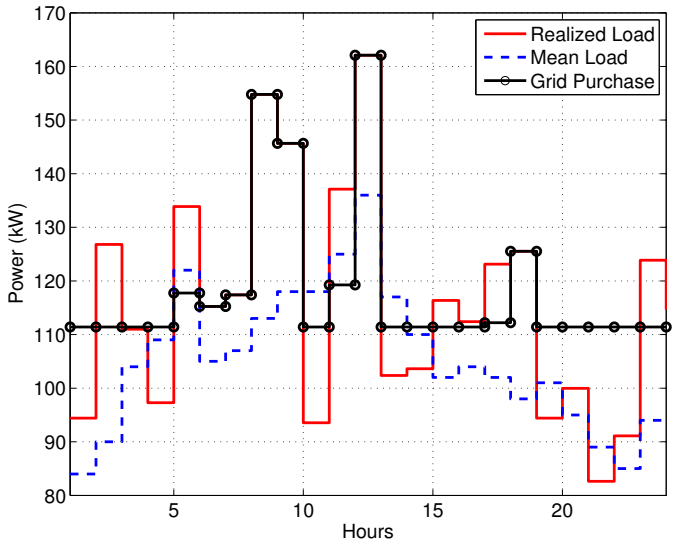


Figure 3.19: Grid purchase for instance 11, Naive Algorithm 1

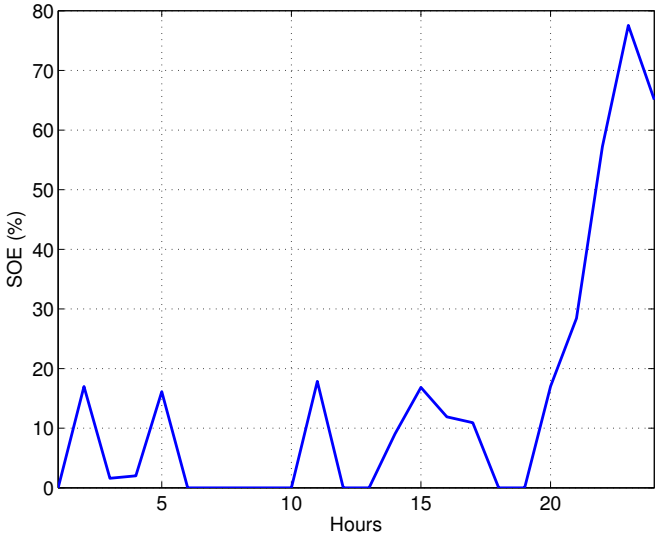


Figure 3.20: Battery SOE for instance 11, Naive Algorithm 1

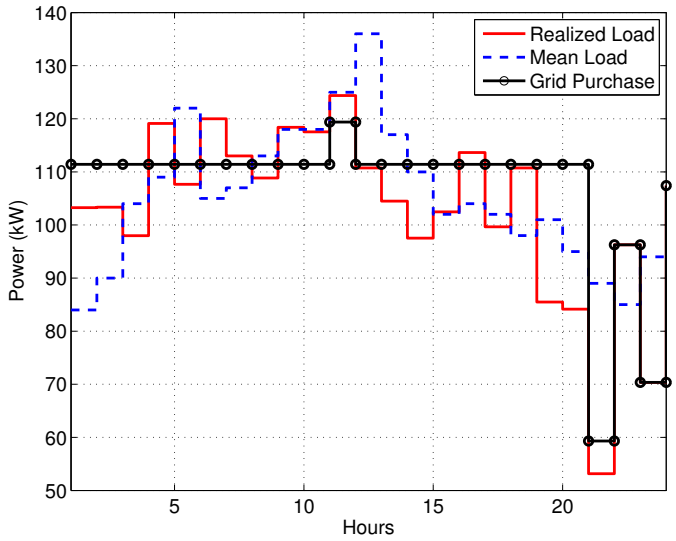


Figure 3.21: Grid purchase for instance 81, Naive Algorithm 1

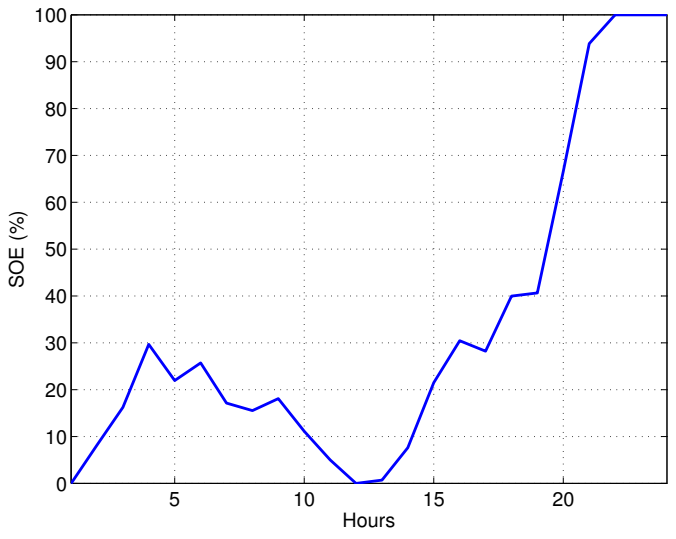


Figure 3.22: Battery SOE for instance 81, Naive Algorithm 1

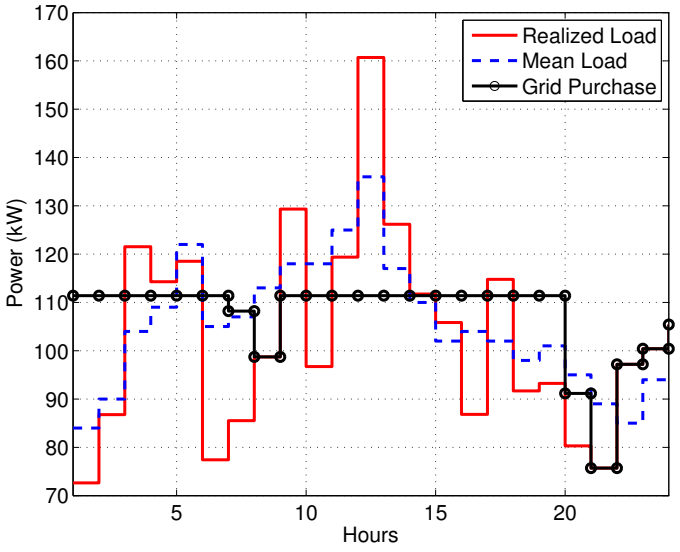


Figure 3.23: Grid purchase for instance 22, Naive Algorithm 1

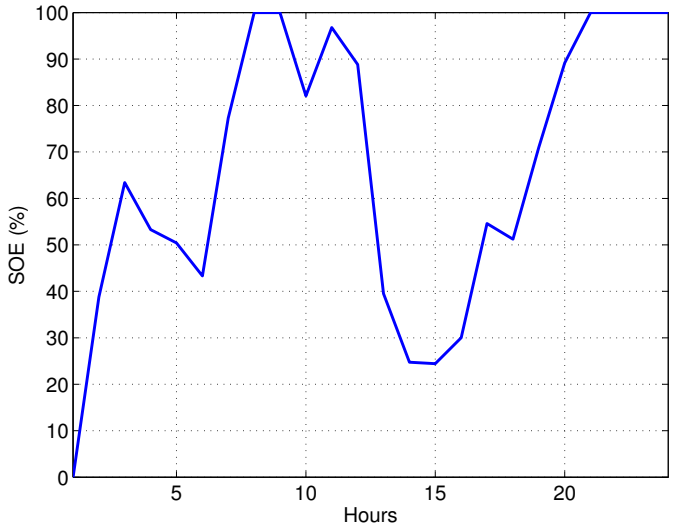


Figure 3.24: Battery SOE for instance 22, Naive Algorithm 1

is updated at every time period:

$$\eta_t = \max\left\{l_1 - \bar{u}_1, \cdots, l_{t-1} - \bar{u}_{t-1}, l_t - u_t^*, E(L_{t+1}) - u_{t+1}^*, E(L_T) - u_T^*\right\} \tag{3.14}$$

where $\bar{u}_i$ means the action has been taken at time $i < t$. For $j \geq t$, u_j^* is the solution to the optimization problem:

$$\min\left\{\max\left\{\begin{array}{c}\{l_1 - \bar{u}_1, \cdots, l_{t-1} - \bar{u}_{t-1}, l_t - u_t, E(L_{t+1}) - u_{t+1}, E(L_T) - u_T\} \\ s.t.\ 0 \leq s_t - \sum_{i=t}^{j} u_i \Delta t \leq C,\ \forall j \in 1, \cdots, T \\ -P\max \leq u_j \leq P\max,\ \forall j \in 1, \cdots, T\end{array}\right\}\right\} \tag{3.15}$$

Algorithm 6 describes the implementation of Naive Algorithm 2.

Algorithm 6 Naive Algorithm 2

1: At beginning of time t, observe battery SOE level and load L_t:
Solve optimization problem in defined in (3.15) to obtain η_t.
If $L_t < \eta_t$, charge the battery up to η_t:

$$u_t = -\min\{\eta_t - L_t, P\max, (C - s_t)/\Delta t\};$$

elseif $L_t > \eta_t$, discharge the battery down to η:

$$u_t = \min\{L_t - \eta_t, P\max, s_t/\Delta t\};$$

else do nothing.
2: Execute the action u_t; update the battery SOE $s_{t+1} = s_t - u_t \Delta t$.
3: If $t < T$, set $t = t + 1$ and return to 1, otherwise stop.

Computational Performance of Naive Algorithm 2

We implement Naive Algorithm 2 on the same case study defined in Section 3.6.2. The performance is summarized in Figure 3.25. The overall mean and standard deviation are 124.71 and 12.85, respectively. Compared to Naive Algorithm 1, the standard deviation value is improved by 12.6%. This is an indication that by incorporating the load observation in the optimization procedure, the cost variability can be improved. However, this is still not ideal compared to either the DP algorithm or the Sample Average Approximation algorithm.

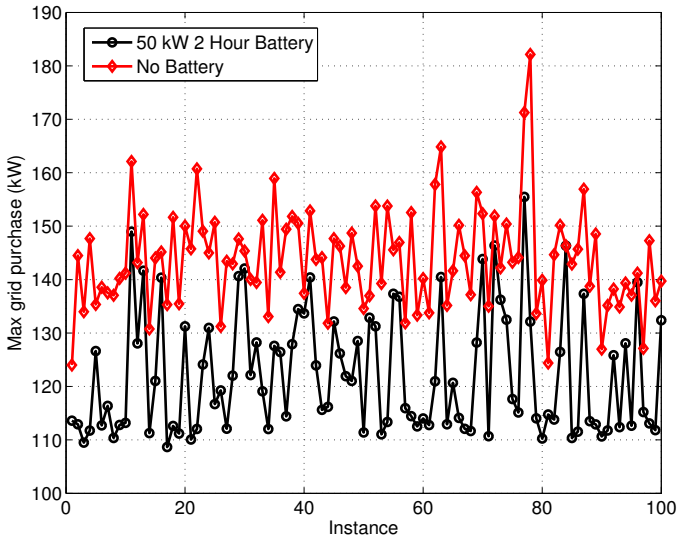


Figure 3.25: Maximum grid purchase for first 100 instances, Naive Algorithm 2

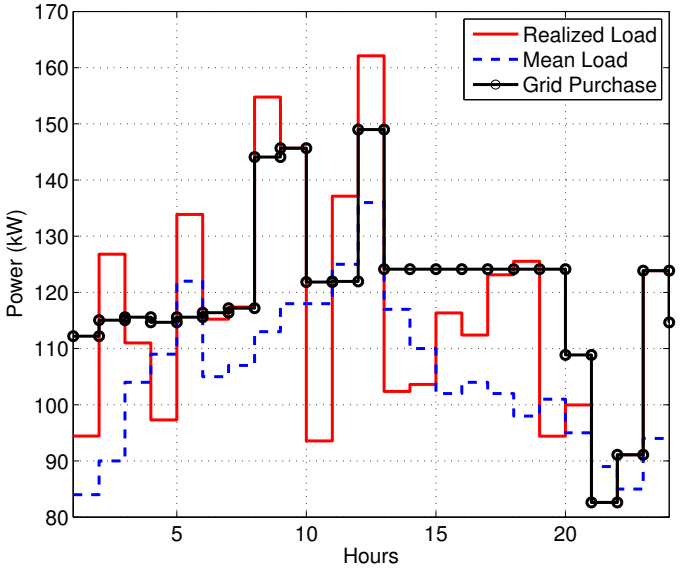


Figure 3.26: Grid purchase for instance 11, Naive Algorithm 2

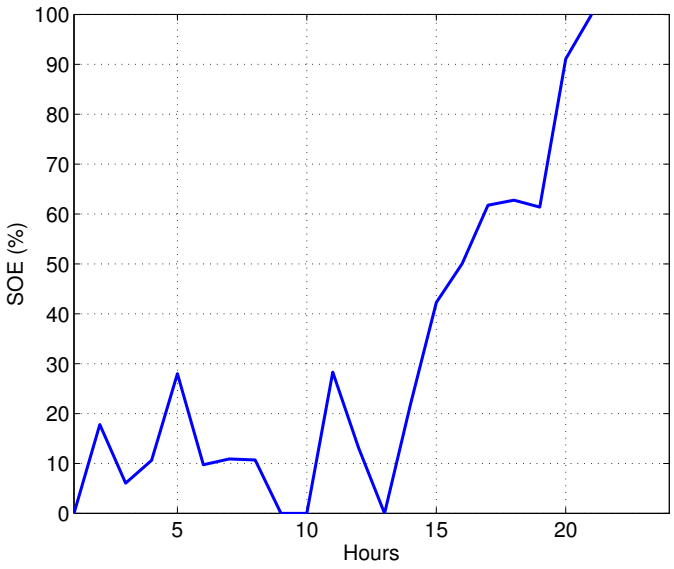


Figure 3.27: Battery SOE for instance 11, Naive Algorithm 2

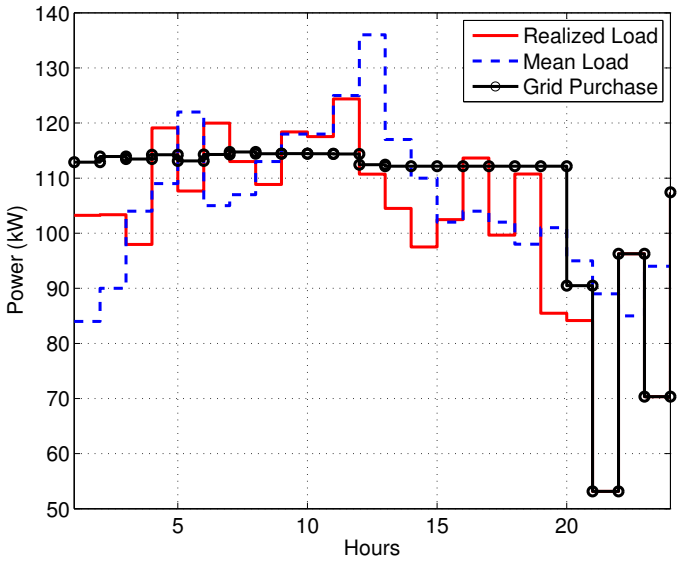


Figure 3.28: Grid purchase for instance 81, Naive Algorithm 2

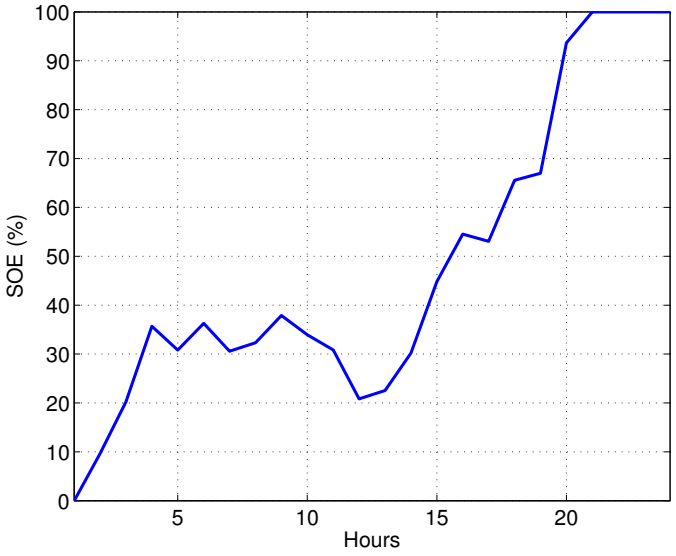


Figure 3.29: Battery SOE for instance 81, Naive Algorithm 2

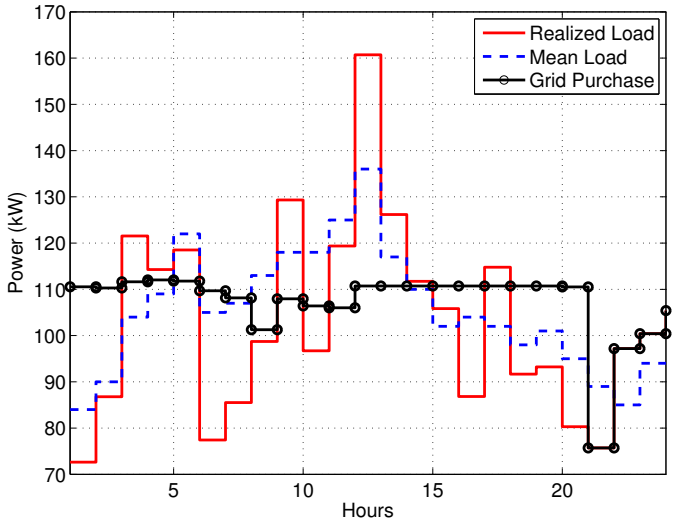


Figure 3.30: Grid purchase for instance 22, Naive Algorithm 2

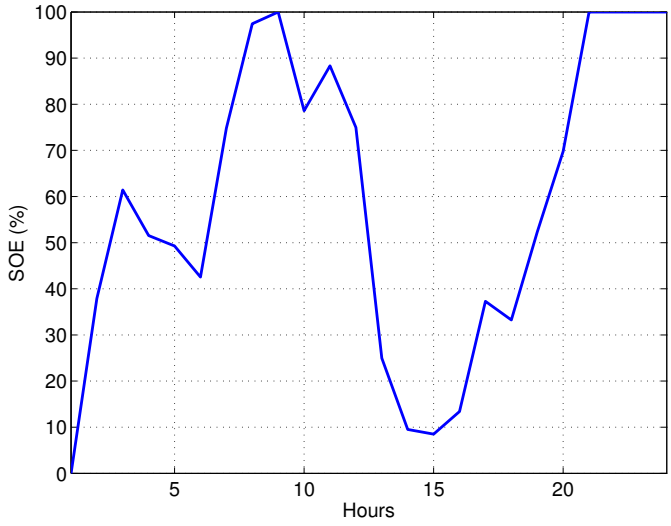


Figure 3.31: Battery SOE for instance 22, Naive Algorithm 2

3.8.3 Naive Algorithm 3

Description

Naive Algorithm 2 is similar to Naive Algorithm 1, in that it is trying to search for a proper threshold level, the level the grid purchase will be reduced to. The performance of Naive Algorithm 2 is not satisfactory, and one explanation for that is that the battery system does not plan for unexpected demand. If the load profile is deterministic, then the η value from the mean is the optimal threshold level. If the load is random, then with the same battery system, the optimal solution may not be able to achieve the same threshold level, since the battery system not only needs to reduce the peak, but also needs to buffer against the randomness of the demand. In this case, we let the threshold level vary from the mean η by a factor r, and then we test if there is any effective factor that can improve the performance of the test instances.

Naive Algorithm 3 has the same structure as Naive Algorithm 1; see Algorithm 5. The only difference is that we use $r\eta$ as a threshold instead of η, where $r > 0$. Now the question is how to find a proper r. We propose the following:

1. Generate a set of M instances (trajectories) ξ^i from the distribution. (ξ_t^i is a realization of the random demand with distribution $N(\mu_t, \sigma_t)$.)

2. Given any $r \in \{r_{\min} : r_{\max}\}$, for each randomly generated trajectory ξ^i, execute Naive Algorithm 3. Obtain the mean cost value μ_r and the standard deviation of the cost σ_r.

3. Generate a tradeoff curve by plotting μ_r vs. σ_r. The desirable factors correspond to points that lie in the bottom left corner of the plot.

4. Select a subset of factors for which the mean cost value is satisfactory according to the decision makers' preference. Among these, choose the factor whose standard deviation value is the smallest.

Note that in this selection method the value M needs to be large enough so that the mean μ_r and the standard deviation of the cost σ_r, are accurate.

Computational Performance of Naive Algorithm 3

We test the performance of Naive Algorithm 3 on the same case study defined in Section 3.6.2. To find the proper factor, we let $r \in \{0.5 : 0.01 : 2\}$ and choose $M = 2000$. For each factor r, there is a mean cost value μ_r, and standard deviation of the cost σ_r. To see the tradeoff, we plot μ_r vs. σ_r in Figure 3.32. Each point in the curve represents a factor value. We want to choose a factor for which both the mean and the standard deviation values are low. From the plot, we can see that the desirable solutions are in the bottom left corner, which have been marked red. Clearly there is a "cost-variance tradeoff". If the planner is very risk-averse and wants to make sure the result is as stable as possible, he/she will choose the factor with the smallest standard deviation (right end point in the red band). On the other hand, if the planner prefers lower average cost, then he/she will choose the factor with the smallest average cost (left end point in the red band). Any point in between these two points can be a proper factor.

In this example, based on the solutions from the DP and SAA algorithms. We choose factors with $120 \leq \mu_r \leq 135$, among which the factor corresponding to the smallest standard deviation is $r = 1.18$. Therefore $r = 1.18$ is the choice for the factor. Then the threshold value is $111.38 \times 1.12 = 131.43$. We set $\eta = 131.43$ and execute Algorithm 5 on

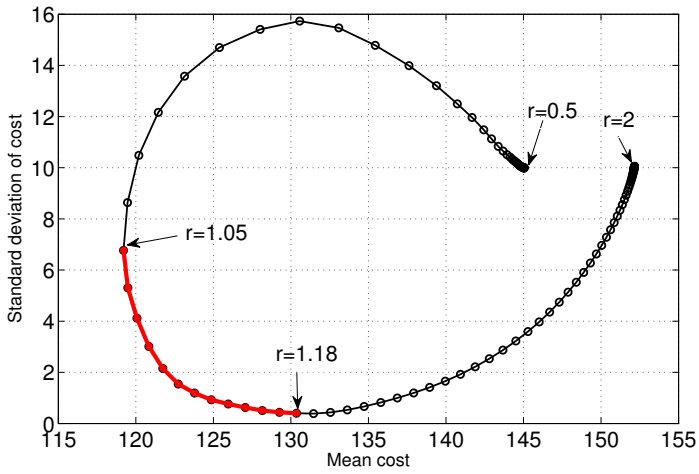


Figure 3.32: μ_r vs. σ_r for $r \in \{0.5 : 0.01 : 2\}$

the same set of instances we use to test the DP and SAA algorithms. The result shows the mean cost is 131.43 and the standard deviation of the cost is 0.94, which is smaller even than the standard deviation of the cost for the DP and the SAA algorithms. Though the mean cost is higher. The average CPU time for this algorithm is 10.10 seconds. If we were to choose the result with the smallest mean cost, the threshold value would be $r = 1.05$. The resulting cost is 117.02 and the standard deviation is 6.95.

The results are comparable to the results we get from the DP and SAA algorithms. Naive Algorithm 3 is straightforward, searching for a proper fixed threshold level for the system and executing according to this threshold. It offers flexibility for the planner to choose according to her preference. The downside of Naive Algorithm 3 is that choosing a proper threshold value is not trivial, since when the variance of the load increases, the proper threshold value also increases. We haven't managed to find a generic threshold-selecting policy that finds the proper factor.

3.9 Summary of Results

We summarize the performance of the proposed algorithms on the case study defined in Section 3.6.2 in table 3.1. The result shows that Naive Algorithm 3 (minimum mean cost) provides the minimum mean cost, but the variance of the cost is the largest. Naive

Algorithm	Mean of the cost	SD. of the cost	CPU time (sec.)
DP	125.48	1.88	1673.00
SAA	124.50	1.59	7.35
Naive Algorithm 1	125.13	14.71	0.30
Naive Algorithm 2	124.71	12.85	5.00
Naive Algorithm 3 (risk averse)	124.87	0.94	10.10
Naive Algorithm 3 (min cost)	117.02	6.95	10.10

Table 3.1: Result summary of different algorithms

Algorithm 1 and 2 also produce large variances. As we already stated, large variance is not desirable. DP, SAA and Naive Algorithm 3 (risk averse) provide more stable results. The CPU time of the DP algorithm is much longer than the other two algorithms. Both SAA and Naive Algorithm 3 (risk averse) perform well on this particular instance. We next test the algorithms on a set of additional cases to compare their performance.

3.10 Performance Analysis

In this section, we discuss the performance of all algorithms on a number of instances consisting of different sizes of battery systems and randomly generated load profiles. We randomly generated 20 instances. The mean load profile μ is a composed of two *sin* curves ($sin(x_1)$, $sin(x_2)$, only the choose single wave crest for each *sin* function) plus some random noise. $sin(x_1)$ centered at period 12 to capture the peak during the noon, it has period of 8π, and the amplitude of the curve is randomly generate from a uniform distribution, $\theta_1 \sim U(200, 500)$. $sin(x_2)$ centered at period 17 to capture the peak during the afternoon (5:00 p.m.), it has period of 7π, and the amplitude curve is randomly generate from a uniform distribution, $\theta_1 \sim U(300, 1000)$. The standard deviation is $\sigma_t = 0.2\mu_t + \epsilon_t$ for t, where $\epsilon_t \sim U(0, 200)$.

For the DP approach, the discretization level is fixed, $\delta = (\delta_s, \delta_p, \delta_u) = (80, 80, 40)$. For the Naive Algorithm 3, we provide the results which yield the minimum cost (called min cost), and the minimum standard deviation (called risk adverse) on a fixed range of weighted threshold values where $r \in [0.5, 2.5]$. The results of these 20 instances are listed in Table 3.2, and 3.3.

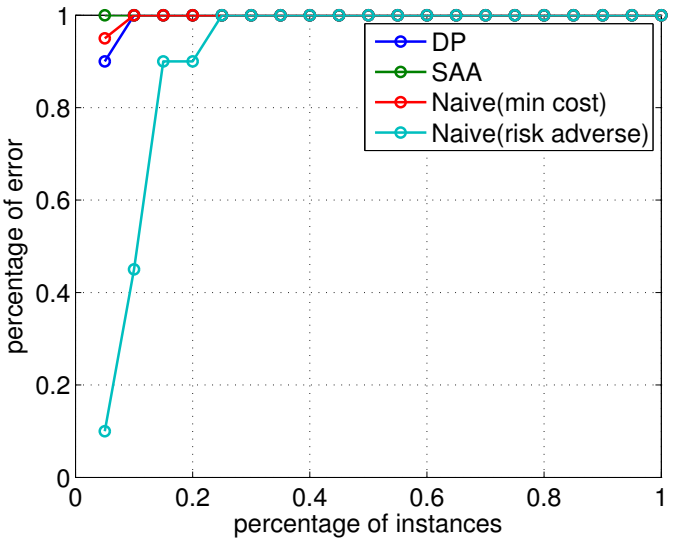


Figure 3.33: Comparison of policies over 20 instances

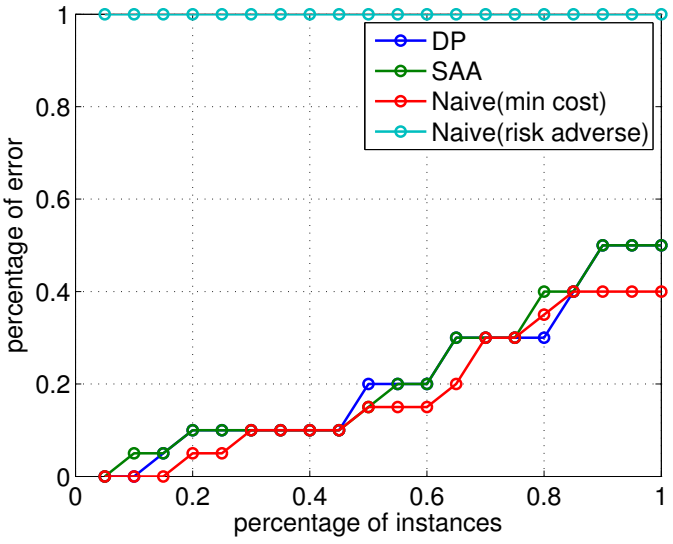


Figure 3.34: Comparison of policies over 20 instances

Instance	DP Algorithm			SAA Algorithm		
	Mean	SD	CPU time	Mean	SD	CPU time
1	125.5	1.88	6037.1	124.56	1.59	1.63
2	171.77	18.45	5537.3	169.63	18.82	2.12
3	1159.83	50.41	5433.2	1144.05	49.16	2.31
4	296.4	14.97	6011.2	295.24	14.4	1.6
5	562.21	21	5866.3	559.43	19.56	2.72
6	612.25	21.8	5319.3	610.8	20.16	2.16
7	610.9	52.4	5519.8	608.29	51.99	1.67
8	590.8	95.1	5668.8	586.96	95.02	2.42
9	495.2	45	6988	492.51	44.44	2.16
10	548.7	65.8	6996.4	545	65.95	2.06
11	894	102.9	6586.3	887.84	103.06	1.43
12	600.4	69.2	5434.7	597.59	66.21	2.2
13	1290.2	184.44	11530	1275.5	186.6	8.92
14	1069	169	12756	1036.4	159.7	10.03
15	810	99	12687	789	93.5	13.28
16	763	108	13210	727	90.66	13.47
17	1492	286.5	6976	1480	285.1	2.7
18	666.8	95.8	6120	661.97	95.6	2.95
19	1056.2	123.5	5756	1049	123.3	1.7
20	1053	133	23228	965.23	125.97	72.56
Total Average	743.41	87.91	8183.07	730.30	85.54	7.50

Table 3.2: Result summary of DP and SAA algorithms on different instances

Figures 3.33, 3.34 and 3.35 show the performance profiles for the mean cost, standard deviation of the cost and the CPU time of the cost. The percentage error is the difference between the value for a given algorithm and the best value found.

To interpret the performance profiles, let i represent the instance solved and p represent the policy used, and let the cost value obtained by solving instance i with policy p be denoted $J(i,p)$. Then the performance profile is generated by the following procedure:

1. For each instance i solved, find the minimum cost among all algorithms, $J_{\min}(i) = \min_{a\in\mathcal{A}} J(i,a)$. Here, $\mathcal{A}$ represents the set of algorithms; for our study, $\mathcal{A}$ = {DP, SAA, Naive Algorithm (min cost), Naive Algorithm (risk adverse)}.

2. Compute the relative cost difference between each algorithm and the minimum-value algorithm:
$$D(i,a) = \frac{J(i,a) - J_{\min}(i)}{J_{\min}(i)}$$

3. For each algorithm a, calculate $G(a)$, the percentage of instances for which $D(i,a) \leq$

Instance	Naive Algorithm min cost			Naive Algorithm risk adverse		
	Mean	SD	CPU time	Mean	SD	CPU time
1	119.08	6.32	11.17	132.53	0.45	11.17
2	166.29	25.28	11.91	189.55	11.53	11.91
3	1154.62	49.84	12.71	1191.94	23.37	12.71
4	290.76	30.71	12.22	322.64	5.83	12.22
5	561.4	24.41	11.72	588.69	4.57	11.72
6	611.4	20.15	11.7	655.02	0.47	11.7
7	614.96	60.85	11.7	655.15	47.47	11.7
8	592.68	105.24	11.89	641.85	63.55	11.89
9	494.46	46.77	11.94	540.84	19.41	11.94
10	546.81	65.6	11.75	594.84	35.61	11.75
11	889.34	101.86	11.96	945.13	68.68	11.96
12	606.6	78.45	12.15	662.31	37.2	12.15
13	1294.86	179.63	23.77	1458.61	99.88	23.77
14	1063	158.91	23.23	1182.55	78.82	23.23
15	817.88	92.25	23.28	894.78	45.29	23.28
16	766.3	101.16	23.94	874.52	35.65	23.94
17	1494.07	281.11	12.17	1663.23	173.69	12.17
18	665.12	106.19	12.21	795.5	43.25	12.21
19	1049.41	123.13	12.79	1076.29	104.29	12.79
20	998.08	118	47.13	1083.04	70.08	47.13
Total Average	739.86	88.79	16.07	807.45	48.45	16.07

Table 3.3: Result summary of Naive Algorithms on different instances

α, for various $\alpha \in [0\%, 100\%]$.

4. Plot $G(a)$ as a function of α.

An algorithm is desirable if it converges to 1 faster. From Figure 3.33, SAA, Naive (min cost) and DP produce similar results for the mean cost. Among them SAA performs the best. For the standard deviation of the cost in Figure 3.34, Naive (risk adverse) outperforms all other algorithms. This is a result of the "cost-variance" tradeoff as seen in Section 3.8.3. The CPU time comparison show in Figure 3.35 indicates that SAA is the overall winner. Note that although the Naive Algorithm is straightforward, there is no generic way of finding the best threshold levels, and to check for a range of threshold level ($r \in [0.5 : 2.5]$) takes time. SAA, on the other hand, is very efficient in terms of the CPU time.

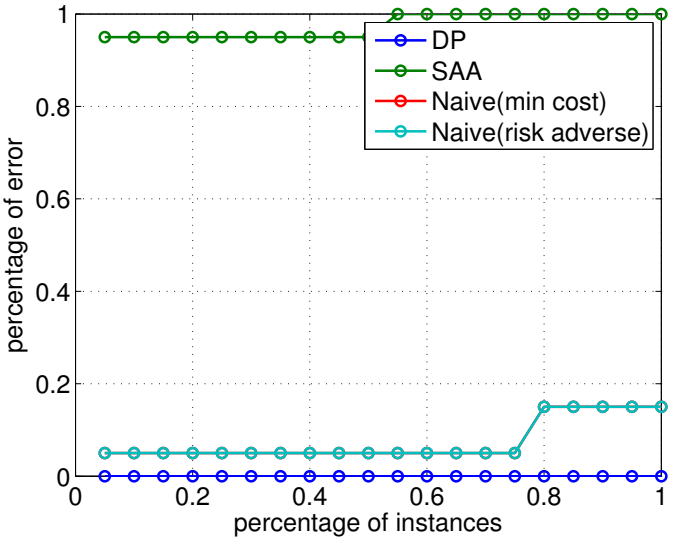


Figure 3.35: Comparison of policies over 20 instances

3.11 Real Time Planning

In practice, the input data constantly gets updated. For example, the forecast load value for 11 am made at 6 am may be very different from the forecast for the same load at 10:45 am. The system is required to study changes in the inputs and then respond to these inputs and make decisions for the battery in a fast manner.

Forecasting tools such as this one exist in practice. For example, our industrial partner has implemented a tool that works in this way. However, for reasons of scope and confidentiality, details of the forecasting method are omitted from this proposal. Instead, we assume that the forecasts $\mu_{k,t}^{(j)}$ are simply given as inputs to the battery planning algorithm.

3.11.1 Introduction to Real Time Planning Algorithm

The demand charge will be incurred based on the peak demand within each month. It is worth pointing out the difference in time scope, since the demand is forecasted for the next 96 time steps each time, but the demand charge is calculated based on the peak power usage within each month. Ideally, the optimization should have a planning horizon of one month, but this is not practical. One reason is straightforward: for each time period in

the optimization problem, there will be a decision variable; for a month with 15 minute resolution the number of variables is $4 \times 24 \times 30 = 2880$, which is too large to be solved quickly enough. Moreover, the predictor can only provide demand forecasts for the next one or two days with a reasonable forecasting error. The predicted demand one month ahead is not reliable since the power demand can be affected by many factors (temperature, random events, etc.). From this point forward, we assume the predictor forecasts the load one day ahead of time, and the planning horizon is no larger than the prediction horizon, i.e., we assume the planning horizon is also one day.

Therefore, the optimization algorithm will solve the problem one day at a time, and for each day, the battery will strive to minimize the grid purchase that happens within this day. Suppose there are altogether K days in a month. Once the system has progressed through all days of that month, the demand charge can be post-processed: It is measured by the peak grid purchase that happened within this month multiplied by the unit demand charge for that month: $\max_{k\in\{1\cdots K\},j\in\{1\cdots T\}}\{l_{kt}-u^*_{kt}\}d$, where u^*_{kt} is the action provided by the SAA Algorithm.

We assume that the battery SOE level at the end of each day can be carried over to the next day. In the deterministic case where the load is known ahead of time, the demand charge obtained by solving an optimization problem one day at a time is an upper bound on the cost when the optimization problem solves for the whole month at once. This is intuitively straightforward. Solving the planning problem one day at a time overlooks the tradeoff between cutting the current peak to a lower level and saving battery charge to prepare for the peak load in the future.

Since the demand charge is based on the peak grid purchase within one month, and the optimization horizon is only one day, the optimization problem needs to introduce a value that accounts for the "peak demand up to now" within each month. We denote this value as $gmax$.

Suppose the system starts from day 1, time period 1. Assume the planning horizon is one day, 24 hours, with 15 minute resolution, and the number of days in a month is K. For each month, the general real time planning scheme is given in Algorithm 7. The general

Algorithm 7 Real time SAA Algorithm

1: At day $k \leq K$ time period $t \leq T$, observe the battery SOE level and demand $l_{k,t}$. Obtain forecasting mean and forecasting error of the load for the next $96 - t$ periods. If $k = 1$ and $t = 1$, set $gmax = 0$.

2: Solve the optimization problem

$$J = \min_{u \in U_{k,t}} \{f(u) := E(\max(gmax, l_{kt} - u_{k,t}, L_{k,t+1} - u_{k,t+1}, L_{k,T} - u_{k,T}))\} \quad (3.16)$$

by SAA for the the optimal action u^*, which is:

$$\begin{gathered} \hat{J} = \min \left\{ \hat{f}(u) := N^{-1} \sum_{i=1}^{N} \max(gmax, l_{kt} - u_{kt}, \xi^i_{k,t+1} - u_{k,t+1}, \xi^i_{k,T} - u_{k,T}) \right\} \\ s.t.\ 0 \leq s_{k,t} - \sum_{z=t}^{m} u_{k,z} \leq C, \quad \forall m = t, \cdots, T, \\ -P \max \leq u_{k,m} \leq P \max \quad \forall m = t, \cdots, T. \end{gathered} \quad (3.17)$$

3: Execute the action u^*_t, update the maximum grid purchase and the battery SOE level:

$$gmax = \max\{gmax, l_{kt} - u^*_{kt}\},$$

$$s_{k,t+1} = s_{k,t} - u_{k,t}\Delta t.$$

4: If $t < T$, set $t = t + 1$ and return to 1, else if $k < K$, set $k = k + 1, t = 1$, and return to 1, else stop.

real time planning scheme states that at every time instant in the optimization algorithm, the system needs to solve optimization problem (3.4) using the SAA Algorithm.

When the real time planning algorithm is adopted, the system re-optimizes every 15 minutes after each new power usage demand is observed. The solution time is therefore very critical for real time implementation. Originally, for 24 hour-15 minutes resolution problem, the optimization time on average is around 300 sec., to solve a problem of one year, then approximate optimization time is $300 \times 365 = 109,500$ sec. which is equivalent to 30.4 hours. Clearly this is not practical. However, we manage to reduce the solution time considerably by specifying the gradient of the objective function explicitly using an approximation due to [Bertsekas, 1995].

3.11.2 Computational Performance of Real Time Planning Algorithm

We first study the performance of the real time planning algorithm on a medium sized facility: a facility with maximum monthly power demand more than 500kW but less than 1000 kW. We have one year of historical load data available for such a facility. This data

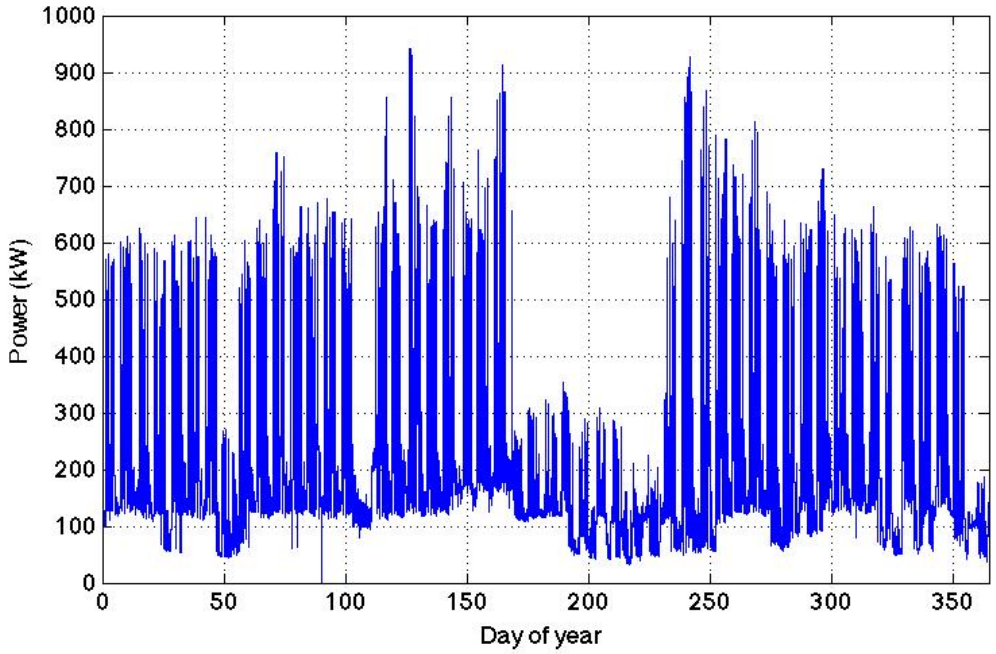


Figure 3.36: Facility historical load

set is of 1 hour resolution. Figure 3.36 shows the complete historical load profile. It can be seen from this plot that from May to August the power consumption is much lower than other months, which indicates that those are months with less power usage activity, for example, summer break months. It also can be seen from the load profile that there is a major difference between weekdays and weekends. Figure 3.37 shows a typical weekday (day 60) and weekend day (day 238). It is clear that during weekdays, energy consumption reaches its peak during the middle of the day. During the weekends, the consumption level is much lower. In practice, events such as summer/winter breaks, weekends, holidays, system maintenance, etc., are scheduled ahead of time therefore can be forecast well. The predictor has a prediction horizon of one day and is trained and tested based on this one year of data. The way that the predictor works is not our focus in this proposal; rather, it is used as an input for the algorithm. Then at day k of the month and time t, the system observes the load $l_{k,t}$, and the predictor provides load prediction for the next 23 hours with 1 hour resolution and a forecasting error around each load prediction point. Now we address the following question: if the next year's load profile is identical to the historical year, then

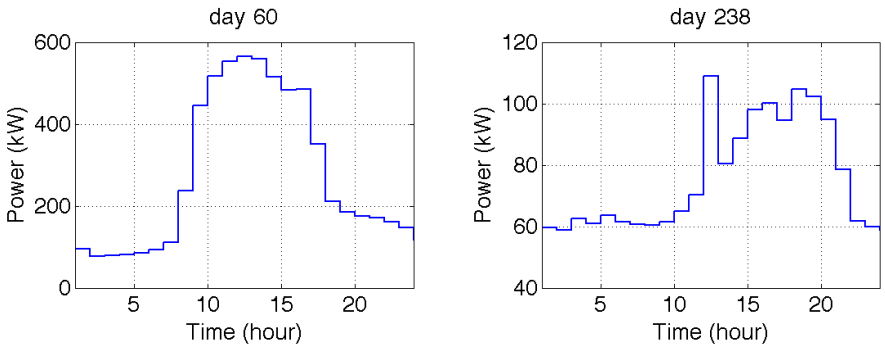


Figure 3.37: Weekday and weekend load

what is the performance of the real time planning algorithm? To answer this question, we run the real time planning algorithm on this case study. The predictor provides the forecasting mean and the forecasting error. Here we assume the forecasting error for the load made j periods ahead of time is the same, e.g., given j, $E[\xi_{k,t}^{(j)}] = \mu^{(j)},\ \forall\ k,\ t$.

Computational Performance of Real Time Planning Algorithm on a Single Day

We first study the performance of a particular day. Note the real time planning algorithm has a planning scope of one month. To convert it to account for a single day, let $K = 1$, which is equivalent to saying there is only one day in a month. The demand charge is measured by the peak grid purchase of that day.

In order to evaluate the efficiency of the real-time planning algorithm, we introduce two baselines: no-battery case and perfect-knowledge case. If there is no-battery in the system, then at every instant of time, the grid purchase equals the demand. Therefore, the demand charge for the no battery case (denoted v_o) is an upper bounded on the cost provided by the real time planning algorithm (denoted v_s). On the other hand, if the power usage demand can be foreseen ahead of time, more specifically, if the system sees the power usage demands $l_1, l_2, \ldots l_T$ before the day starts, then by solving the deterministic optimization problem

$$\min \left\{ \max_{t\in\{1,\cdots,T\}} (l_t - u_t) | \text{constraint (3.2) (3.3)} \right\} \tag{3.18}$$

the system will make a perfect schedule for battery charge and discharge. The cost obtained

from solving this problem (denoted v_p) is a lower bound on the cost obtained from the real time planning algorithm.

Therefore, we define a metric α to help evaluate the performance of the real time planning algorithm:

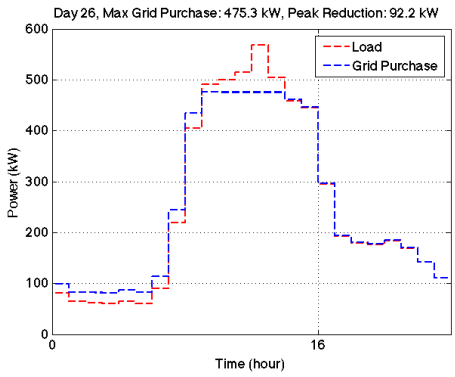
$$\alpha = \frac{v_s - v_p}{v_o - v_p}$$

where v_s is the objective value from the real time SAA algorithm. And v_p is the objective assuming load is known before hand. v_o is the objective value without the battery system. α represent how real time SAA algorithm is close to the perfect knowledge case, using the no-battery case as a baseline. If $\alpha = 1$, then $v_s = v_p$, and if $\alpha = 0$, then $v_s = v_o$. Note that sometimes α can be negative. For example, in the previous case study of Section 3.5, the result of Instance 81 by the SAA Algorithm (Figures 3.16 and 3.17) indicates the actual grid purchase is greater than the peak demand. In this case, $\alpha < 0$. The real time planning algorithm is efficient if α is close to 1.

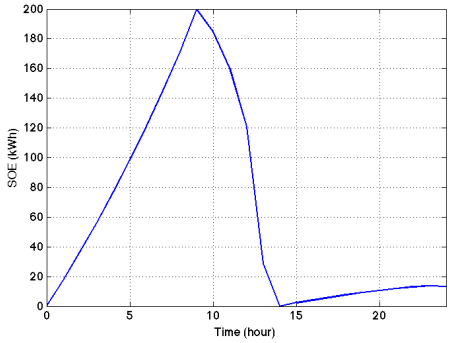
We first pick day 26 of the year for demonstration. Day 26 a weekday. We assume the battery is a 100 kW, 2 hour battery and that the system starts with an empty battery.

We first study the perfect knowledge case. Figure 3.38(a) shows the load and the grid purchase level of every time period. It can be seen that the peak load happens during the middle of the day, and the peak load is shaved off by the battery, resulting in a lower maximum grid purchase level. Figure 3.38(b) plots the battery SOE level in every time period. It can be seen from this subplot that the battery starts to charge gradually from the beginning of the horizon, and gets fully charged at time 9; from that time on, the battery starts to discharge in order to reduce the maximum grid purchase. At time 14, the battery is depleted. And after that, the battery again slowly charges itself. If the load is known ahead of time, the system can use the battery most efficiently. In this case $v_p = 475.3$ kW.

If there is no battery installed, the maximum grid purchase is the maximum load, which is $v_o = 567.5$ kW. Now we study the performance of the real time planning algorithm. We show the result by plotting the load observation/prediction and grid purchase in real time; see Figure 3.40. Each subplot corresponds to a different value t of the current time



(a) Grid purchase of day 26



(b) SOE of day 26

Figure 3.38: Perfect knowledge for day 26

period, and for each, we plot 3 values:

1. The red curve represents the load realization from time 1 up to time t.
2. The blue curves, with one curve for each period up through t, plot the mean demand forecast from time t up to the end of the time horizon, T,
3. The green curve records the maximum grid purchase up to time t.

At time $t = 1$, the system observes load l_1, and receives forecast information for the future. Then according to the real time planning algorithm, it solves optimization problem (3.17) to obtain the best action; from the figure we can see that the load realization is 80 kW, and the maximum grid purchase is 100 kW; therefore we charge the battery by 20 kW. This is intuitive, since the forecast of the future indicates there could be a much higher load in the future. Therefore, in order to reduce the overall demand charge, the battery needs to charge right now and prepare for the peak in the future. After the action has been taken, the system moves to time $t = 2$. The system first observes the demand at time 2. Then the predictor responds to the observations of time 1 and time 2 and generates a new forecast for the future. Again, a battery charge and discharge action is suggested by solving the optimization problem (3.17) with a new set of inputs. At time 2, the decision is to charge the battery. The system will keep repeating this process, following the action suggested by the algorithm. It can be seen from the figure that the battery starts to discharge at hour 10 up to hour 13, striving to reduce the peak demand to a lower level. After that,

the demand becomes lower, and the maximum grid purchase stops increasing. Figure 3.41 shows the evolution of the battery SOE throughout this process. Starting from zero, the battery charges to its full capacity, and then starts to discharge between time 10 and 13, and after that charges to the capacity level again.

The maximum grid purchase for this case is $v_s = 517.6$ kW. Therefore, the peak reduction and the α value for this case are:

$$\text{Peak Reduction} : v_o - v_s = 567.5 - 475.3 = 49.9 \text{ kW}$$

$$\alpha(\text{day26}) = \frac{v_o - v_s}{v_o - v_p} = \frac{567.5 - 517.6}{567.5 - 475.3} = 54.1\%$$

Given a 100 kW, 2 hour battery, the real time planning algorithm achieves 54.1% of the peak reduction that would be possible in the perfect knowledge case.

The comparison between the perfect knowledge case and the stochastic case reveals that the real time planning algorithm in general behaves more conservatively.

Computational Performance of Real Time Planning Algorithm for One Year

We now implement the real time planning algorithm with one year of data. Figure 3.39 plots a summary of the results. This plot shows the monthly peak reduction in kW for the three algorithms. For any algorithm P, the peak reduction for month j is computed as follows: Assume there are K_j days in month j and T time periods within each day. Then

$$\text{Reduction}(j, P) = \max_{k \in \{1,\ldots,K_j, t \in \{1,\ldots T\}\}} l_{k,t} - \max_{k \in \{1,\ldots,K_j, t \in \{1,\ldots T\}\}} g^P_{k,t},$$

where $g^P_{k,t}$ is the grid purchase of day k, time t obtained using algorithm P. Let $P \in$\{stoch,perf\} where stoch is the stochastic case, and perf is the perfect knowledge case. The stochastic case provides peak reduction of 75% of the perfect knowledge overall. That is,

$$\frac{\sum_{j=1}^{12} \text{Reduction}(j, \text{stoch})}{\sum_{j=1}^{12} \text{Reduction}(j, \text{perf})} = 75\%$$

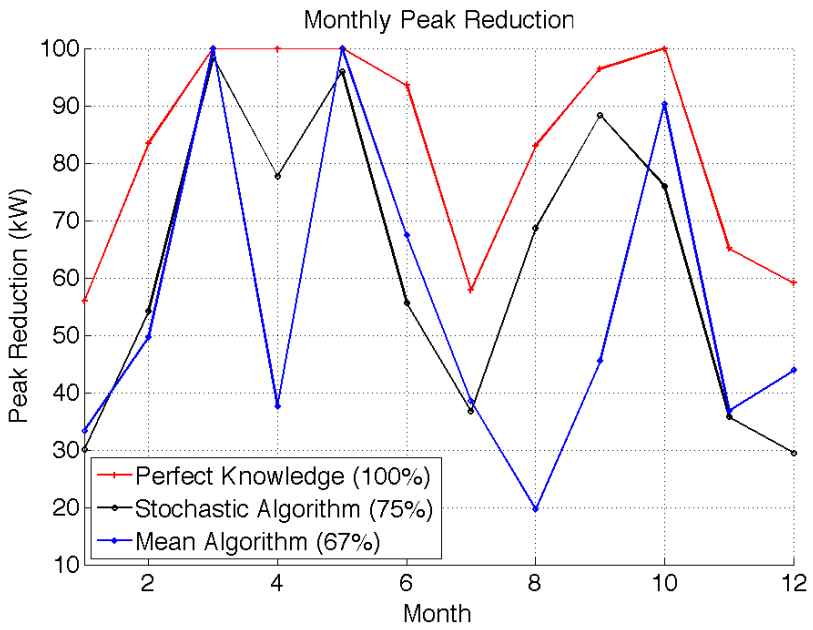


Figure 3.39: Performance summary for one year example

. In contrast, the mean algorithm achieves a reduction of 67% in total. We also notice that there are some months in which both the stochastic algorithm and the mean algorithm perform well in peak shaving. For example, during March and May, these two algorithms achieve the perfect knowledge case. There are also months in which the stochastic algorithm outperforms the mean algorithm. During April and August, the stochastic algorithm outperforms the mean algorithm by more than 40 kW.

3.12 Conclusions and Future Work

In this chapter, we study a power consumption optimization problem with a battery storage system under a demand charge tariff. We develop a minmax dynamic programming algorithm for the deterministic case. We then introduced a novel dynamic programming algorithm for the case where the load is assumed to be stochastic. Next we move to real time implementation; in this case, the planner expects to suggest commands for the battery system in a fast manner, and reacts to the change of the load input. We develop a Sample Average Approximation algorithm. To study the performance of the SAA Algorithm, later we develop three other naive algorithms, as comparison. The case study shows

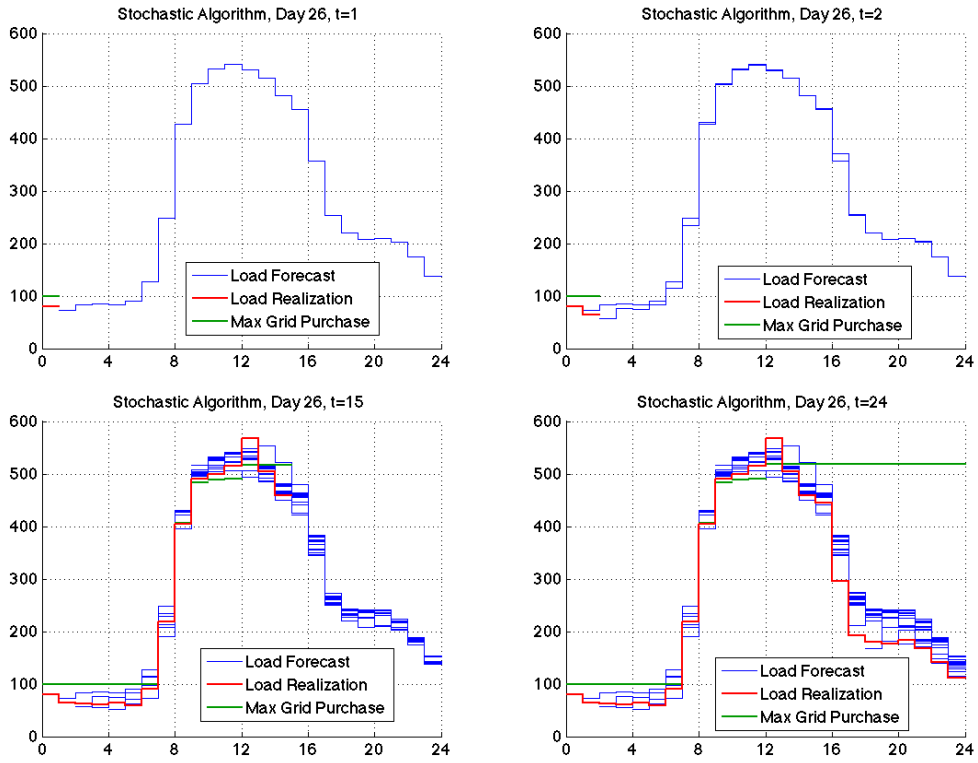


Figure 3.40: Real time planning for day 26

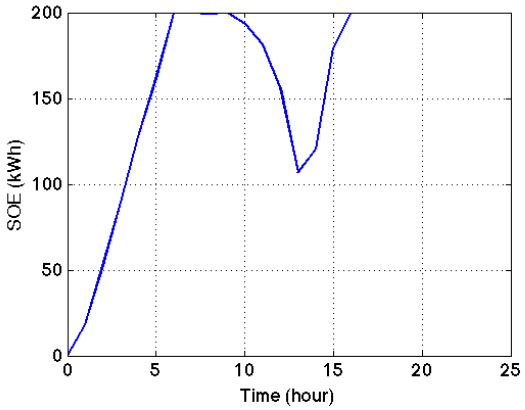


Figure 3.41: Evolving of battery SOE level

that the SAA Algorithms, DP algorithm and Naive Algorithm for minimum cost produce cost effective solutions. But the SAA Algorithm and Naive Algorithm with minimum cost are more time efficient. Next we develop a real time SAA Algorithm, assuming the load forecast gets updated every 15 minutes. The study shows that the planning tool achieves 75% of the savings compared to the case of perfect knowledge.

In the future, we interested to see if there exist efficient policies or heuristics which can improve the performance of the planning tool in general. In particular, it is worthwhile studying the existence of an optimal threshold policy for the stochastic problem. We can then also extend this topic to additional applications, for example, to develop the planning algorithm under a multiple-demand-charges (demand charge defined by TOU) tariff.

Chapter 4

A PV (Photovoltaic) Plant Power Feed-in Problem with Battery Storage System under Profile Constraints

4.1 Problem Introduction

The increased amount of photovoltaic penetration has presented opportunities to reduce CO_2 emissions. However, standard PV plants introduce variability. A battery storage system can be deployed to reduce power fluctuation. During the energy trading process, the battery storage system is deployed to buffer against the volatility of the PV generation and help reshape the PV power feed-in level according to constraints incurred by the grid operator.

In this chapter, we study the following problem: a PV farm equipped with a storage system plans to sell PV generation to the grid operator. To regulate the power received from the PV generator, the grid operator has imposed complex constraints on the feed-in profile. The feed-in profile includes three phases: ramp up phase, quasi stationary phase and ramp down phase. There are constraints associated with each phase. Moreover, the

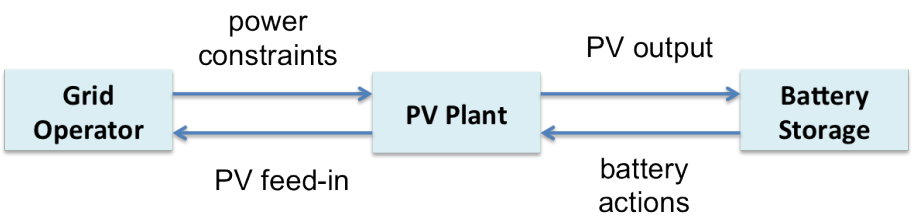


Figure 4.1: System layout

length of each phase is also to be decided by the PV system itself. A penalty will be incurred if for some reason one or more of the constraints are violated during real time implementation. Figure 4.1 shows the information exchange among entities within the system. During implementation, the grid operator presents the power feed-in constraints to the PV generator. The battery system observe the PV output level. The optimization scheme collects this information and makes plans regarding the amount of power to inject/withdraw from the battery, as well as the amount of power to be fed in to the grid. In this study, we approach this problem both from the deterministic and the stochastic point of view and make the following contributions: for the deterministic setting, we model this problem as a mixed integer programming (MIP) problem; our case study shows that it solves quickly enough for a 5 minutes resolution 24 hour problem. For the stochastic setting, we provide a dynamic programming algorithm, and show that if the phases constraints are relaxed, then there exists an optimal threshold inventory policy for this problem.

The remainder of this chapter is organized as follows. In Section 4.2 we review the literature and in Section 4.3 we provided the MIP formulation and show the result of this model on some typical days. In Section 4.3.5 is the case study in which we provide performance analysis on the model. In Section 4.4 we derive the dynamic programming algorithm for the stochastic problem. In Section 4.5 we show the structural inventory result after relaxing the grid constraints.

4.2 Literature Review

Standard PV plants exhibit power variations. To properly utilize this resource, rules and regulations require combining the PV generator with energy storage technology, among

which the battery storage systems have been considered in many case studies. There are many applications for PV combined with storage, for example, frequency regulation, spinning reserve, load-shifting, peak shaving, etc. In this study, we are interested in a PV feed-in problem with complex constraints imposed by the power purchaser. This problem has features of both the PV smoothing problem and the power commitment problem. The goal is to develop a power control and dispatch strategy for this problem.

The performance of the system is associated with the type of the battery storage system and the type of PV system used, and their sizing. Singh [2013] provides a review of the progress made in solar power generation by PV technology. There is an extensive literature on the sizing of PV farms, highlighted by Diaf et al. [2007] and Diaf et al. [2008]. For a battery storage system, Vazquez et al. [2010] presents a review of different battery technologies and their applications. Papers by Tan et al. [2010], Roy et al. [2010], Venu et al. [2009], and Borowy and Salameh. [1996] discuss how to size the battery system when renewables are present. In our study, we assume the size of the PV farm is given, and two different sizes of battery system are being studied.

Applications of control to the PV smoothing problem are widely studied. Among these, Li et al. [2013] study a control strategy for regulating power output levels and battery state of charge. Defourny et al. [2013] propose to use a point forecast of energy from renewable to handle high level of uncertainties. Senjyu et al. [2008] propose an energy storage system (ESS) based control method for PV smoothing. Daud et al. [2013] present an improved control strategy for a grid-connected hybrid PV/BES system for reducing PV farm output power fluctuations. Ellis et al. [2012] describe the real time control algorithm for PV smoothing. Fakham et al. [2011] designed a power control system of a battery charger in a hybrid PV generator for load-following applications. These studies related to our work, but we approach the problem from the optimization point of view, in which rigorous constraints are present.

From the optimization point of view, the power dispatch problem has been studied in recent decades for setting that include, but are not limited to, photovoltaic power. Among those studies, Teleke et al. [2010], Kim and Powell [2011], Grillo et al. [2012], Salas and

Powell [2013], Shu and Jirutitijaroen [2014], and Zhou et al. [2014] are closely related. Teleke et al. [2010] provide a control paradigm for a battery storage system in order to dispatch battery power to satisfy the requirement which respecting the battery limitations. The uncertainty of the wind energy is not accounted for. Kim and Powell [2011] consider a wind power commitment problem for a wind farm with storage. Their goal is to find the optimal commitment quantity for each period in order to optimize the overall power sale. A certain penalty is incurred if the commitment is not met. An analytical solution is derived in the end. Their problem is similar to yet different from our problem, in that in order to derive the analytical solution, their model requires a number of assumptions such as stationarity in the wind and price processes, and that the error in the wind is uniformly distributed. Grillo et al. [2012] study the use of energy storage to enhance renewable energy integration. In this paper, the management strategy is implemented by a forward dynamic programming algorithm. No analytical result is derived from this problem. Salas and Powell [2013] present an ADP algorithm that provides a near optimal solution for wind storage problems. Shu and Jirutitijaroen [2014] propose an adaptive optimal policy for operation of an energy storage system with stochastic wind generation. The objective is to optimize daily profit by selling wind power to the grid. The problem is modeled as a stochastic dynamic programming problem. No closed form solution was derived under this setting. Zhou et al. [2014] study the optimal wind power trading problem with battery storage under transmission capacity, assuming both trading price and wind generation are stochastic. The authors proof that if the electricity price is non-negative, there exists an optimal battery inventory structure. These works are related to our problem; however our problem is different in that the power purchaser incurs complex phase constraints and penalties.

4.3 Deterministic PV Feed-in Problem

In this section, we consider the PV feed-in problem assuming the PV generation is deterministic. As stated previously, the goal of the PV farm is to sell power to the grid operator (power purchaser) to obtain maximum revenue. To make sure that the power they receive

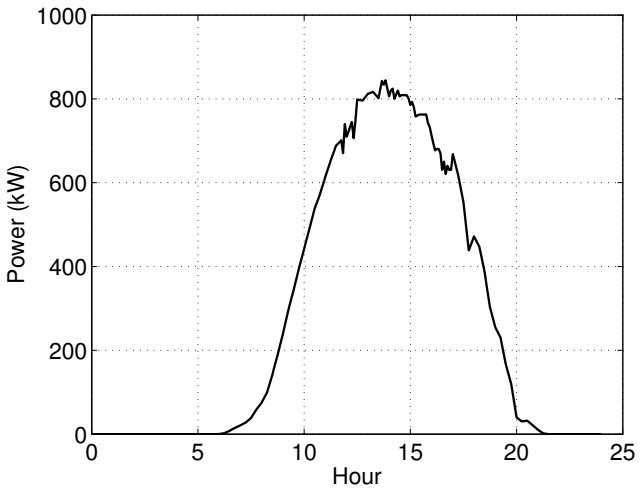


Figure 4.2: PV power generation, sunny day

is manageable, the grid imposes incur certain constraints on the power profile received. A 24 hour, 5 minute resolution PV generation profile is shown in Figure 4.2 to provide a better understanding of these constraints.

Figure 4.2 demonstrates the PV generation of a 1 MW PV plant for a normal sunny day. In this day, the PV generation increases after sunrise until it peaks in the afternoon, then it start to ramp down and diminishe after sunset. According to the nature of PV generation, the grid operator has defined 3 phases for the PV power feed-in over the 24 hours horizon:

1. ramp up phase: the phase where the PV feed-in level gradually increases or stays the same.

2. qusai-stationary phase: the phase where the PV feed-in power varies within a certain band, and the band is defined as a reference level $\pm$ a fixed oscillation value.

3. ramp down phase: the phase where the PV feed-in level gradually decreases or stays at the same value.

During the ramp up phase, the power feed-in increase rate is constrained to be no larger than 0.6% of $Pmax$ per minute, where $Pmax$ is the maximum PV output available, e.g., the capacity of the PV plant. During the qusai-stationary phase, the power feed-in

should be within the range $Pref \pm 2.5\% Pmax$, where $Pref$ is a reference value announced by the PV farm at the end of the ramp up phase. And according to the grid operator, $Pref$ should not be greater than 40% of $Pmax$. For example, if the PV farm is rated of 1 MW, the reference power value should not exceed 400 kW. Similarly to the ramp up phase, during the ramp down phase, the power feed-in level should decreasing, and the rate of decrease should be no more than $0.6\% Pmax$ per minute. If any of the phase constraints are violated, the next hour's power feed-in revenue will be forfeited.

The phase constraints have to be in order: ramp up - quasi stationary - ramp down. Once the system starts to ramp down, it will not go back to the other two phases. Also, to avoid degeneration (where the system goes from the ramp up phase right to the ramp down phase), the system imposes a constraint on the length of the quasi stationary phase.

What makes this problem even more challenging is the choice of the critical time points at which the feed-in power makes the transformation from the ramp up phase to the quasi-stationary phase, and from the quasi-stationary phase to the ramp down phase. Let us call these two time periods t_s and t_f. It is required by the grid operator that the choice of t_s and t_f can only be on the hour or half hour. During real time implementation, these two time points must be announced beforehand.

Figure 4.3 demonstrates the power feed-in profile according to the constraints for one day. The black curve captures the PV generation profile for a sunny day of a 1 MW PV plant. The blue curve demonstrates the PV feed-in power for the entire day with a 500 kW 1 hour battery. The choices of t_s and t_f are marked by red circles. In this example $t_s = 10$ (10:00 a.m.) and $t_f = 19.5$ (7:30 p.m.) Therefore, the ramp up phase is from sunrise to 10:00 a.m., the qusai-stationary phase is from 10 a.m. to 7:30 p.m., and the ramp down phase is from 7:30 p.m. until sunset. The reference power level of the quasi stationary phase is set to 400 kW, exactly 40% of the capacity of the PV plant. It can be observed from the output that the battery is charged during morning and afternoon, and after around 6:00 p.m. it starts to discharge to provide power when the PV generation decreases. There is no violation during the entire process; therefore the PV operator earns revenue for.

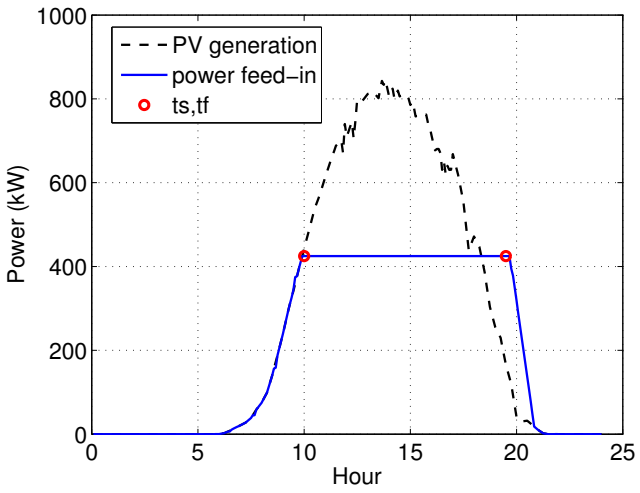


Figure 4.3: PV power generation, sunny day

Figure 4.3 is a demonstration of the power dispatch result for a typical day. In practice, we may observe intricate power profiles, especially for some cloudy days. And the power dispatch plan may not be as straightforward. Next, we study the mathematical formulation of the problem which takes a general power output profile as input. We formulate this problem as a mixed integer programming problem.

4.3.1 Problem Formulation for the Deterministic Case

We assume the battery has the following characteristics. The battery capacity, is presented by the high and low energy level, ($\overline{B}$, $\underline{B}$). The maximum battery charge and discharge rate are denoted by ($\underline{U}$, $\overline{U}$). We assume that the battery has a charge and discharge efficiency of η_c and η_d. (The round trip efficiency is $\eta_c\eta_d$.) The convention is that a negative value means power is flowing out of the battery (discharge) and a positive value means power is flowing out of the battery (charge).

We define the following parameters:

T: total number of optimization periods;

Δt: time interval (in minutes) between time $t-1$ and time t;

$\overline{PV}_t$: power output of the PV plant;

D: number of optimization periods within one hour;

L: the shortest allowable length (in periods) of the quasi stationary phase;

$M_1, M_2, \cdots, M_5$: big M value introduced to perform the "if" and "or" operations;

ϵ: error introduced for the flow balance constraint;

Decision Variables:

q_t: the power feed-in for which the PV operator gets paid;

P_t: the pesudo power feed-in assuming the grid constraints are satisfied. Note this is not the actual power feed-in;

a_t: an auxiliary variable for flow balance constraints. The actual PV feed-in level is $P_t - a_t$;

PV_t: the power from the PV plant that is utilized;

$Pref$: the reference power during the quasi stationary phase;

u_t: the internal power of the battery;

u_t^+, u_t^- positive and negative part of the battery power, $u_t^+ \geq 0$, $u_t^- \geq 0$;

E_t: the terminal power of the battery;

k_t: a binary variable, $k_t = 1$ if $-\epsilon \leq a_t \leq \epsilon$, $k_t = 0$ otherwise;

l_t: a binary variable, $l_t = 1$ if $a_t \geq \epsilon$, $l_t = 0$ if $a_t \leq -\epsilon$;

x_t: a binary variable, $x_t = 1$ if t is in the ramp up phase;

y_t: a binary variable, $y_t = 1$ if t is in the quasi stationary phase;

z_t: a binary variable, $z_t = 1$ if t is in the ramp down phase;

ts, tf: the periods in which the ramp up phase ends, and the the ramp down phase begins, respectively, in 30 minute unit.

The mixed integer formulation of the deterministic problem is as follows:

$$\max \sum_{t=1}^{T} q_t \Delta t/60 \tag{4.1}$$

$$\text{s.t.} \quad -M_1(1-x_{t-1}) \leq P_t - P_{t-1} \leq 0.006Pmax + M_1(1-x_{t-1}),\ \forall t = 2,\ldots,T. \tag{4.2}$$

$$-2.5\%Pmax - M_2(1-y_t) \leq P_t - Pref \leq 2.5\%Pmax + M_2(1-y_t), \forall t = 1,\cdots,T \tag{4.3}$$

$$-0.006Pmax - M_3(1-z_{t+1}) \leq P_t - P_{t-1} \leq M_3(1-z_{t+1}),\ \forall t = 2,\ldots,T. \tag{4.4}$$

$$P_t = E_t + PV_t + a_t, \qquad \forall t = 1,\ldots,T. \tag{4.5}$$

$$PV_t \leq \overline{PV_t}, \qquad \forall t = 1,\ldots,T. \tag{4.6}$$

$$P_t \leq Pref + 0.025Pmax, \qquad \forall t = 1,\ldots,T. \tag{4.7}$$

$$Pref \leq 0.4Pmax; \tag{4.8}$$

$$u_t = u_t^+ - u_t^-, \qquad \forall t = 1, \ldots, T. \tag{4.9}$$

$$E_t = u_t^+ \eta_d - u_t^- / \eta_c, \qquad \forall t = 1, \ldots, T. \tag{4.10}$$

$$\underline{B} \leq S_0 - \sum_{j=1}^{t} u_j \Delta t/60 \leq \overline{B}, \qquad \forall t = 1, \ldots, T. \tag{4.11}$$

$$\underline{U} \leq u_t \leq \overline{U}; \qquad \forall t = 1, \ldots, T. \tag{4.12}$$

$$x_1 = 1; \tag{4.13}$$

$$z_T = 1; \tag{4.14}$$

$$x_t \geq x_{t+1}, \qquad \forall t = 1, \ldots, T-1. \tag{4.15}$$

$$z_t \leq z_{t+1}, \qquad \forall t = 1, \ldots, T-1. \tag{4.16}$$

$$\sum_{t=1}^{T} y_t \geq L, \qquad \forall t = 1, \ldots, T. \tag{4.17}$$

$$x_t + y_t + z_t = 1, \qquad \forall t = 1, \ldots, T. \tag{4.18}$$

$$ts = \sum_{t=1}^{T} x_t \Delta t/30, \qquad \forall t = 1, \ldots, T. \tag{4.19}$$

$$tf = \sum_{t=1}^{T} (x_t + y_t) \Delta t/30, \qquad \forall t = 1, \ldots, T. \tag{4.20}$$

$$a_t \leq \epsilon + M_4(1 - k_t) \qquad \forall t = 1, \ldots, T. \tag{4.21}$$

$$-a_t \leq \epsilon + M_4(1 - k_t) \qquad \forall t = 1, \ldots, T. \tag{4.22}$$

$$a_t \leq \epsilon - M_4(1 - l_t + k_t) \qquad \forall t = 1, \ldots, T. \tag{4.23}$$

$$-a_t \leq \epsilon - M_4(l_t + k_t) \qquad \forall t = 1, \ldots, T. \tag{4.24}$$

$$l_t + k_t \leq 1 \qquad \forall t = 1, \ldots, T. \tag{4.25}$$

$$q_t \leq P_t - a_t \qquad \forall t = 1, \ldots, T. \tag{4.26}$$

$$\sum_{j=0}^{j=D-1} q_j \leq M_5 k_t \qquad \forall t = 1, \ldots, T. \tag{4.27}$$

$$q_t, p_t, PV_t, Pref, u_t^+, u_t^- > 0, k_t, l_t, x_t, y_t, z_t \qquad \text{binary}, t_s, t_f \text{ integer} \tag{4.28}$$

The objective in (4.1) represents the amount of PV energy for which the PV operators gets paid. Constraints (4.2)-(4.4) model the requirements for the ramp up, quasi stationary

and ramp down phase. Note that the indices of the three phase constraints are being adjusted to avoid disjunctions between ramp up phase and the qusai-stationary phase, and qusai-stationary phase and ramp down phase. Constraint (4.5) models the flow balance constraints. Constraint (4.6) makes sure the total PV power used is less than the PV power available. Constraint (4.7) states that the power feed-in at any time is less than the maximum power feed-in in the quasi-stationary phase $Pref + 0.025Pmax$. (4.8) makes sure the reference power in the quasi stationary phase, $Pref$, is no higher than the 40% of the $Pmax$. Constraint (4.9) separates the battery internal power into the positive part and the negative part. Constraint (4.10) states the relationship between internal power and terminal power. Constraints (4.11), (4.12) state the battery's capacity constraint and charge/discharge rate constraint, respectively. Constraints (4.13), (4.14) ensure that the system starts in the ramp up phase and ends in the ramp down phase. Constraint (4.15) requires that if the ramp up phase has started, it will either stay in ramp up, or go to the other two phases, but once the system enters either of the other two phases, it cannot go back to the ramp up phase again. Similar reasoning holds for constraint (4.16). Constraint (4.17) forces the quasi stationary phase to be no shorter than L periods. Constraint (4.18) makes sure that the system can only be in one phase for each time period. Constraints (4.19), (4.20) force the choices of ts and tf can only be at half hours or the hours. Constraint (4.21)-(4.25) require that if $-\epsilon \leq a_t \leq \epsilon$, then $k_t = 1$, else $k_t = 0$. Constraint (4.26) makes sure that the power for which the operator is paid is smaller or equal to the power feed-in to the grid. Constraint (4.27) forces no payment for the next one hour if $k_t = 1$ at the current time period.

4.3.2 Analysis of the Problem

It is said that to modeling is more of an art than a science. We need to understand the problem well in order to come up with an efficient formulation. There may be many ways of formulating the same problem. Here we briefly analyze the current formulation. We formulated this problem as a mixed integer programming problem with big Ms. The big Ms are chosen to express the "if $\cdots$ then" relationships without introducing any nonlinear

constraints. On the other hand, the big M method is known to produce a loose relaxation. In this case, the performance of this formulation needs to be studied in practice.

One can be creative with the formulation. For example, constraints (4.21)-(4.25) can be formulated without introducing the variable l_t:

$$\epsilon k_t - M_4(1 - k_t) \leq a_t \leq \epsilon k_t + M_4(1 - k_t) \tag{4.29}$$

In our numerical study, we will examine the performance of both of the formulations. For comparison, let us call the model with constraints (4.21)-(4.25) model A and the model with constraints (4.29) model B.

4.3.3 Illustrative Examples

We implemented the model with CPLEX 12.5.1.0. Below is a summary of the instance we consider:

1. The PV profile is available for 24 hours at 5 minute resolution. Capacity of the PV plant $Pmax$ is 1000kW.
2. The storage size is 500 kW, 500 kWh. The storage has a charging efficiency of 0.8 and a discharging efficiency of 1.
3. The grid feed-in price (USD/kWh) over time is a constant value.
4. The initial SOE of the storage is 0.

Below, we focus on a few days from this instance for illustrative purposes. We discuss the case study as a whole in Section 4.3.5.

Sunny Day

Here we show the model results for two typical power profiles, a sunny day and a cloudy day. We obtain the PV output profile from CAISO in CAI [2014]. For the sunny day, the result is shown in figure 4.4. In this example, $t_s = 9.5$ and $t_f = 19.5$. The blue curve plots the paid feed-in. It is clear from the output that at the beginning, the feed-in level starts to

ramp up, and at hour 9.5, the quasi stationary phase starts, and the reference value $Pref$ is set to 400 kW, which is the maximum value allowed by to the constraints. At hour 19.5, the system starts to ramp down according to the ramp down constraint. The overall paid feed-in energy is 4900.5 kWh, whereas the total PV output is 6340.1 kWh. The paid feed-in percentage is 72.7%. Figure 4.5 plots the evolution of the SOE level for the day. From the plot we see that we start with 0 battery SOE level, and the battery is almost empty before hour 10, since the system sells all the power generated in the PV plant to the grid. After that the PV output power reaches its highest output level. (From the problem, we know that 425kW is the maximum feed-in value.) Therefore, the excess power is charged into the battery. There is no penalty incurred during this process. All the PV feed-in is getting paid. One interesting phenomenon we observe here is that the battery is depleted near hour 13. At first, this is not intuitive, since there is plenty of power output between time 10 to 13, and the battery should keep charging. Taking a closer look at the problem gives us the answer to this problem. The battery is bounded by its rate and capacity. The best thing it can do is to be fully charged before the PV output level drops down and elongate the feed-in qusai-stationary phase to get more power feed-in value. Clearly there is still excess power even after the battery is fully charged. In this case, the excess power will be wasted and that is the reason that we observe the battery is emptied in the middle of the day.

Cloudy Day

In a cloudy day, the power output level is much lower than a sunny day and the level of output is not stable. Here the battery is used to buffer out the variations in the output and elongate the ramp down period if possible.

We implement the same model on a cloudy day; the results are shown in Figure 4.6, and Figure 4.7. As one can see from the PV output profile, around hour 12 and 16 there are heavy clouds. In this example, the $t_s = 12.5$ and $t_f = 17$. The paid feed-in value is plotted in orange. We observe from the output that at the beginning, the feed-in level start to ramp up, and at hour 12.5, the quasi stationary phase started, the reference value $Pref$

Figure 4.4: Power feed-in, sunny day

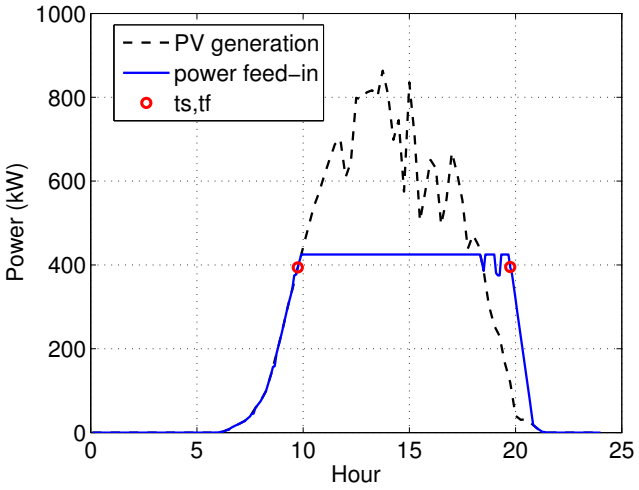
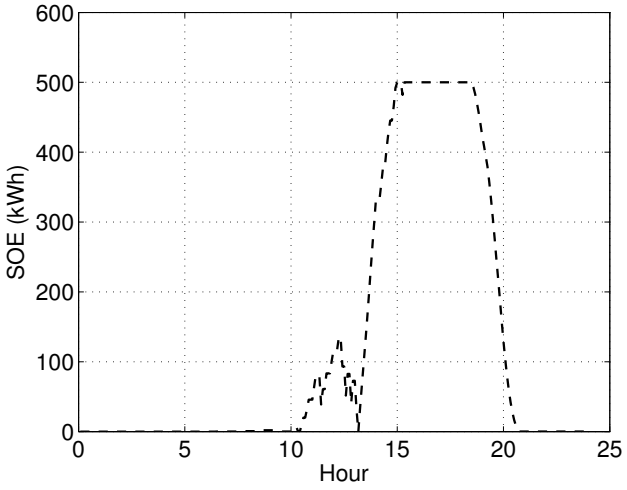


Figure 4.5: Battery SOE, sunny day



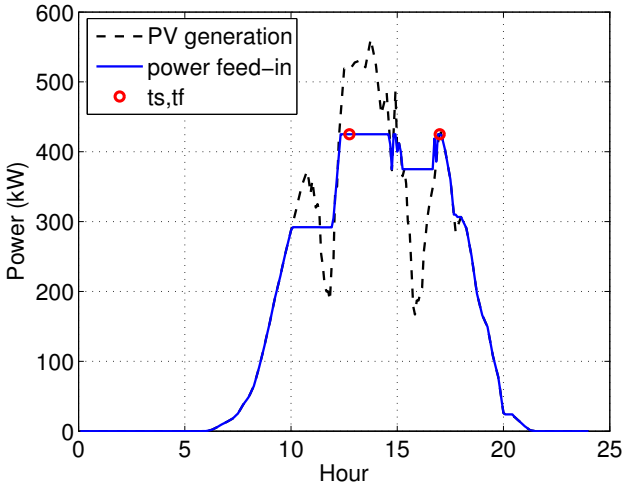


Figure 4.6: Power feed-in, cloudy day

is set to 400 kW, which is the at its maximum value according to the constraints. At hour 17, the system starts to ramp down according to the ramp down constraint. The overall paid feed-in power is 3681.2 kWh, whereas the total PV output value is 3725.2 kWh. The feed-in percentage is 98.5%.

Table 4.1 summarizes the results of these two cases. Both model A and B produce the same results. For the cloudy day, the PV output level is lower; therefore the total energy feed-in is lower. However, with the same size battery, for a cloudy day the system can achieve a higher feed-in percentage, which means less power is wasted. The CPU times of models A and B are different; in particular, model A is solved more quickly than model B. The reason for this difference could be that the LP relaxation of model A is stronger than that of model B.

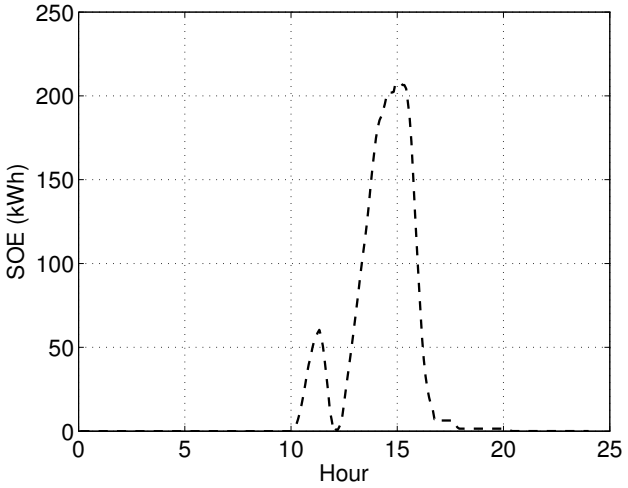


Figure 4.7: Battery SOE, cloudy day

Result:	Sunny Day	Cloudy Day
Feed-in energy	4900.5	3681.2
Feed-in percentage	72.7%	98.5%
Penalty	0	0
CPU time, model A	11.2	48.0
CPU time, model B	29.7	61.3

Table 4.1: Comparison between sunny and cloudy day

A Case Where the Penalty is Incurred

In the examples discussed above, the battery is assumed to be 500 kW, 1 hour, which is relatively large compare to the PV output value. Now we consider a small battery; in this case, the battery may not be large enough to supply enough power to satisfy the phase constraints if the reference value is set high. In this case, two things may happen: 1. the battery lowers the reference value, 2. violation occurs and some feed-in revenue will be forfeited. The decision will be made by the optimization model. Here we show a case where the penalty is incurred. In this case, we set the battery size to 50 kW, 2 hour.

Figure 4.8 shows the result for this instance. (The CPU time for this instance is 201

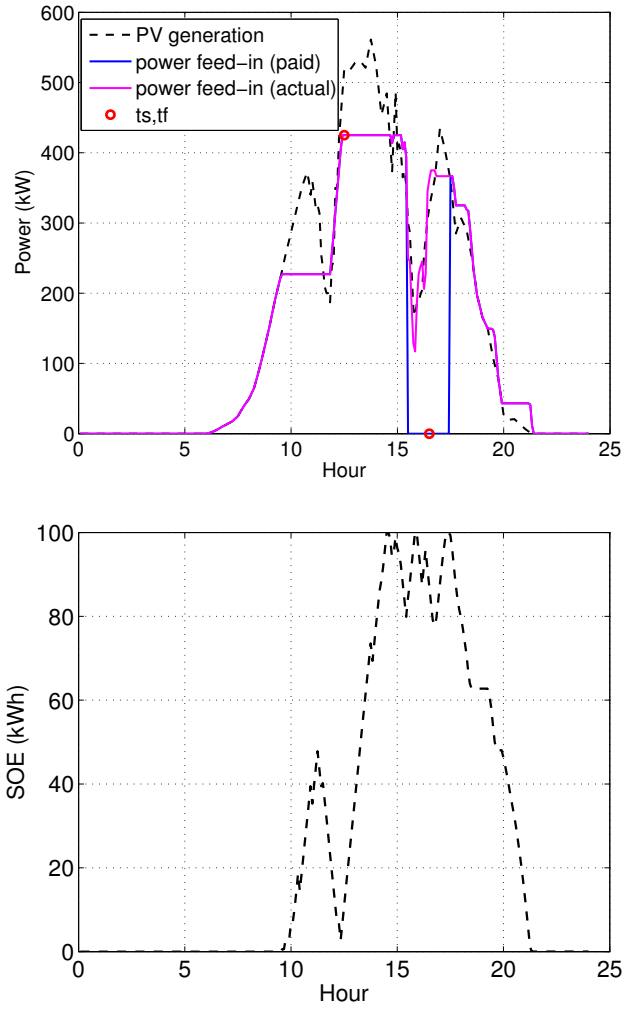


Figure 4.8: Power feed-in and battery actions, cloudy day

sec., objective value is 2852.8.) The penalty starts from hour 15.5 and ends at hour 17.5. For this instance, the penalty is incurred and lasts for 2 hours. The blue curve represents the paid PV feed-in value and the magenta curve represents the actual PV feed-in value. It's not difficult to understand that the system will still inject power to the grid even if it is not getting paid, since that the current action will affect the payment for the next period.

4.3.4 Starting with Nonempty Battery

One of the concern is whether it is more beneficial to start with a non-empty battery. The intuition is that it will not make a significant difference in the deterministic case, since we

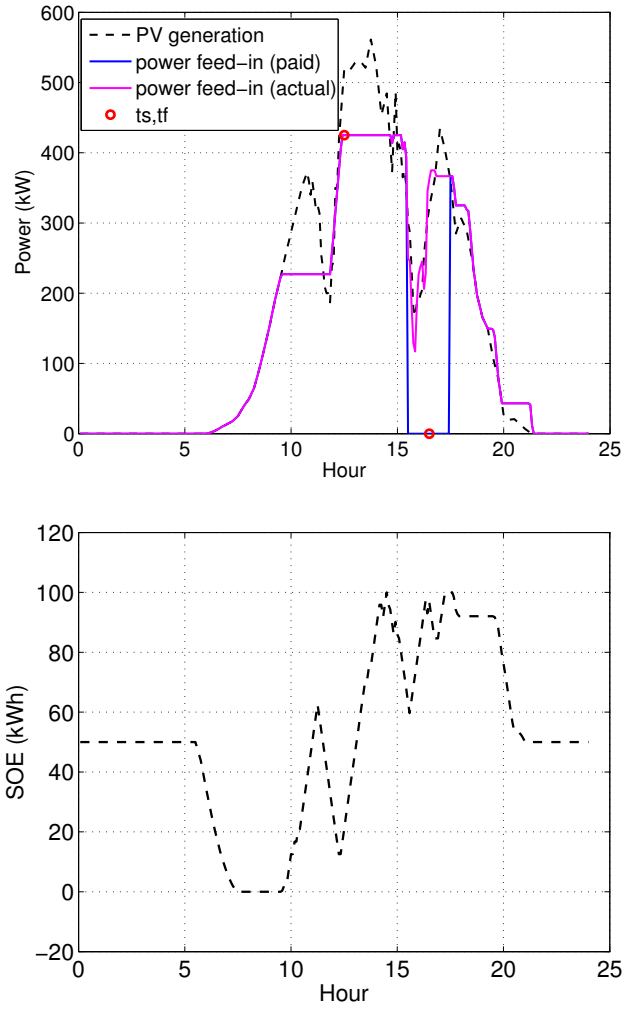


Figure 4.9: Power feed-in and battery actions, cloudy day

observe that the battery is utilized after sunrise and mostly makes a round trip dis/charge during the day according to the PV profile.

Here we show the result of the same cloudy day, where the battery starts halfway charged (50%), and is forced to end with the same SOE level. The total power feed-in level is 2852.8 kW, which is the same as the previous case shown in Figure 4.8.

4.3.5 Case Study

We have shown the optimization result for some typical days. To understand the performance of the MIP model, especially in terms of CPU time, and the power utilization without battery system, we study a set of general instances. We obtain the PV output

profile from CAISO in CAI [2014]. The document records the average solar output for a 24 hour horizon. We randomly chose 20 sample days and scaled the output level to 1 MW. Given that the PV values are the average value of the local PV generation, which has a smoothing effect, we add some random noise to some periods of the profile to simulate the actual PV output level. We randomly choose 2-5 hours during the day and add noise to $\widetilde{PV}_t$, the average PV feed-in at time t. The actual PV output level for the case study is $\overline{PV}_t = \widetilde{PV}_t + \epsilon_t$. The noise ϵ is defined as follows:

$$\epsilon_t = \begin{cases} 0 & \text{if } \omega_t > 0 \\ \epsilon_t & \text{if } \widetilde{PV}_t \leq \omega_t \leq 0 \\ -\widetilde{PV}_t & \text{if } n \text{ is even} \end{cases}$$

Where $\omega_t \sim N(0, 300)$. In this noise function we take a randomly generated normal number. If the number is positive, then we leave the PV value intact; if the number is negative and is smaller than the original PV output, then the actual output level is set to 0; otherwise, the actual PV value is set to be the sum of the observed PV plus the random noise.

Two assumptions we made for the case study are listed below:

1. The round trip efficiency is 0.8, which is incurred during charge, e.g. $\eta_c = 0.8$, $\eta_d = 1.0$.

2. The qusai-stationary phase lasts for at least 4 hours, i.e., $L = 4$.

We sampled 20 days, with 5 minute resolution. We studied 2 different storage sizes, a 500 kW, 1 hour system and 500 kW, 2 hour system. In addition, we compare the solutions to the non-battery case to see how much more solar energy is utilized with a storage system. During the case study, we report the CPU time for both model A and model B.

We first compare the CPU times of model A and B. The average run time for model A is 45.06 seconds and the average run time for model B is 54.06 seconds. A more detailed comparison is displayed in the form of a performance profile.

To generate the performance profiles, let i represent the instance solved and x represent the model used, and let the CPU time value obtained by solving instance i with

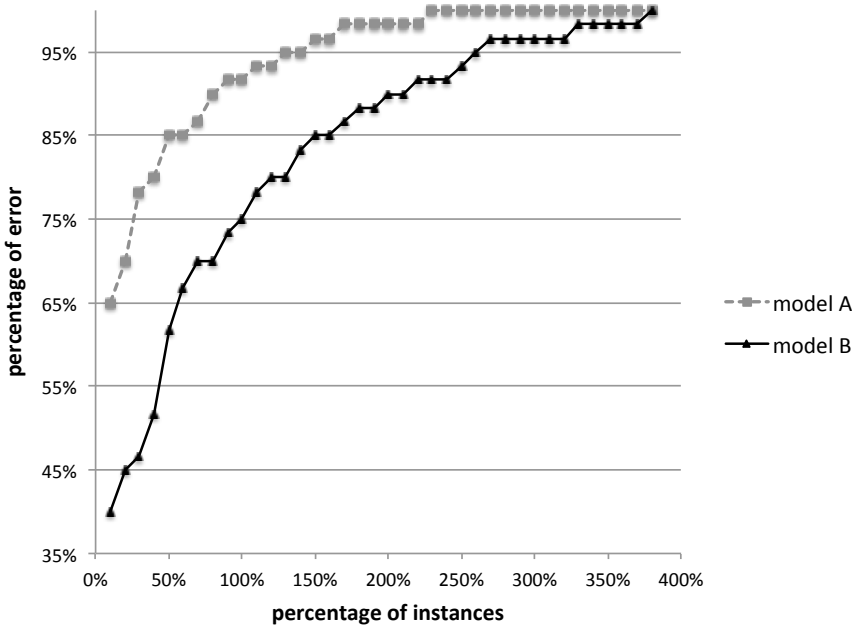


Figure 4.10: Performance profile for models A and B

model x be denoted $J(i, x)$. Then the performance profile is generated by the following procedure:

1. For each instance i solved, find the minimum CPU time between model A and B, $J_{\min}(i) = \min_x J(i, x)$.

2. Compute the relative time difference between each model and the minimum-time model:
$$D(i, x) = \frac{J(i, x) - J_{\min}(i)}{J_{\min}(i)}$$

3. For each model x, calculate $G(x)$, the percentage of instances for which $J(i, x) \leq \alpha$, for different α.

4. Plot $G(x)$ as a function of α.

The result is shown in Figure 4.10. It is clear from the plots that the CPU time of model A is in general less than the CPU time of model B. This shows that different formulations can yield different performance. For MIP problems, it may be worthwhile to model the problem with more constraints and variables.

Day	500 kW, 2 hour	500 kW, 1 hour	No battery
1	3649.33	3374.30	2757.17
2	4125.95	3725.95	2494.49
3	3859.00	3459.00	1910.00
4	5049.45	4649.82	4241.85
5	5015.77	4615.77	4058.59
6	4567.96	4170.26	2690.98
7	736.33	736.35	687.71
8	4828.91	4428.91	3995.33
9	4474.83	4153.03	3719.89
10	3229.25	3199.02	2080.52
11	4224.26	4149.33	2911.13
12	5262.01	4862.86	4092.37
13	5259.82	4859.82	4459.82
14	4765.75	4367.06	3685.74
15	3289.23	3284.63	1977.02
16	5335.94	4937.26	4430.31
17	4133.38	4133.38	2240.82
18	4668.50	4256.48	3453.95
19	5332.27	4932.27	3398.93
20	5182.08	4782.08	3777.49
Total average	4349.50	4053.88	3153.21

Table 4.2: Energy feed-in (kWh) for different systems

Now we compare the solar power utilization from two different battery systems, a 500 kW, 1 hour system and a 500 kW, 2 hour system, compared to the no-battery system. Since the difference between the PV feed-in value (objective value) of model A and model B is $< 0.1\%$, in Table 4.2, we show the results of model A only.

It is obvious from the results that a battery system can enhance PV feed-in significantly compare to the no-battery case. For the 500 kW, 2 hour system, the increase in PV feed-in level compared due to the no-battery case is 1196.30 kWh per day on average, which is 37.93% overall. For the 500 kW, 1 hour system, the value is 900.67 kWh per day on average, which is 28.56% overall.

4.4 Stochastic PV Feed-in Problem

The previous section addresses the dispatch problem when the PV profile is deterministic. In practice, the PV generation changes in real time according to the weather. In this case, the dispatch planner should consider the stochastic features of the PV generation, and make decisions accordingly.

Moving from the deterministic case to the stochastic case is not straightforward, given that complex constraints need to be considered. Therefore, we first relax the constraints imposed by the grid operator, and work toward the more complete problem later.

If the constraints from the grid are ignored, then the goal becomes maximizing the energy sold, making use of the battery storage system. Here we assume the price of energy can be different from time to time. To model this problem, we introduce the following notations:

Parameters:

π_t = revenue per unit of power injected to the grid (\$/kW);

$\underline{U}, \overline{U}$ = minimum and maximum allowable power dis/charge rate;

$\underline{B}, \overline{B}$ = minimum and maximum allowable battery state of charge;

η_c, η_d the charge and discharge efficiency of the battery, $(\eta_c \leq 1, \eta_d \leq 1)$.

Random Variable:

$\overline{PV_t}$ = PV output available at time t;

Decision Variables:

PV_t = the amount of PV power being used at time t (either feed-in or stored into battery);

P_t = the amount of PV power sold at time period t;

u_t = the amount of power dis/charged from/to the battery;

b_t = battery state-of-charge at time t;

$\overline{PV_t}$ is a random variable. We assume that $\overline{PV_t}$ for $t = 1 \cdots T$ are independent with distribution $\overline{PV_t} \sim N(\mu_t, \sigma_t^2)$.

4.4.1 Dynamic Programming Model of Energy Sold Maximization

The goal is to optimize the the PV energy sold assuming the PV output is stochastic. We made the following assumptions:

- the planner observes the PV output first, and then makes decisions in period t;
- there are no constraints on the amount of power that can be feed into the grid at any time;
- the system cannot buy power from the grid;
- the battery has perfect efficiency of, i.e., $\eta_c = \eta_d = 1$.

Under this setting, at each time period, the PV owner need to evaluate what's the optimal amount of power sell to the grid and what's the battery charge/discharge action associate with it.

First, it's not hard to see that if the payment rate π_t is flat, then at any time, the system will sell all the PV output power to the grid. This solution is the optimal solution. This is true because of the following two facts:

1. battery won't produce any power, therefore the total amount of selling back to the grid is the same as the total PV production;
2. the rate is flat, there is no incentive to delay feed-in of a portion of the current PV output to later.

Therefore, the problem is trivial when the payment rate is flat. On the other hand, if the rate is not flat, then it can be benefit to arrange more feed-in during the period when the payment rate is high with the battery system. The dynamic programming model for the problem is discussed next:

Let $\theta_{T+1}(b)$ be the terminal cost incurred if the state-of-charge of the battery at the end of the time horizon is b. Note since there is no feed-in limit at any time, no PV output power will be wasted; therefore we have the relationship

$$P_t = \overline{PV}_t + u_t, \quad \forall t = 1 \cdots T$$

Define the state to be

$$x_t = (b_t, \overline{PV}_t).$$

Then the dynamic programming formulation for this problem is:

$$\theta_t(b, \overline{PV}) = \max_u \left\{ \pi_t(\overline{PV} + u)^+ \Delta_t + \mathbb{E}_{\overline{PV}_{t+1}}[\theta_{t+1}(b - u\Delta_t, \overline{PV}_{t+1})] \;\middle|\; \begin{array}{l} \underline{U} \le u \le \overline{U} \\ \underline{B} \le b - u\Delta_t \le \overline{B} \\ \overline{PV} + u \ge 0 \end{array} \right\} \tag{4.30}$$

If a battery action u is given, the PV feed-in value will be fixed: $\overline{PV}+u$, and the incremental cost at the current period is $\pi_t(\overline{PV}+u)\Delta_t$. The expected cost from time t to T is $\mathbb{E}[\theta_{t+1}(b-u\Delta_t, \overline{PV}_{t+1})]$, where the expectation is with respect to the next state random variable $\overline{PV}_{t+1}$. That is,

$$\mathbb{E}[\theta_{t+1}(b - u\Delta_t, \overline{PV}_{t+1})] = \int_{-\infty}^{+\infty} \theta_{t+1}(b - u\Delta_t, x) f(x)\, dx.$$

where $f(x)$ is the *pdf* of the random variable $\overline{PV}_{t+1}$.

The first two constraints in (4.30) are the dis/charge rate constraints and the capacity constraint. The third constraint makes sure there is no buying power from the grid. According to the three constraints in the model, we have the following relationship:

$$u_{min} = -\min\{-\underline{U}, (b - \overline{B})/\Delta_t\};\ u_{max} = \min\{\overline{U}, (b - \underline{B})/\Delta_t\},$$

where u_{min} and u_{max} are the maximum charge and discharge level of the battery, and the battery SOE level is b.

Once the PV level is observed, according to the no-buying-power constraint, the range of feasible battery actions is shown below:

- if $-\overline{PV} \le u_{min}$: $u \in [u_{min}, u_{max}]$;
- else: $u \in [-\overline{PV}_t, u_{max}]$.

4.4.2 Dynamic Programming Algorithm

Based on the DP model (4.30), the DP algorithm is given in Algorithm 8.

Algorithm 8 DP algorithm for problem with no grid feed-in constraints

1: Let $t = T + 1$, $\theta_T(b, \overline{PV}) = b$.
2: For every given state $x_t = (b_t, \overline{PV}_t)$, evaluate the range of feasible battery actions.
3: For any u within the range, compute the cost incurred by dis/charge:

$$f(b_t, \overline{PV}_t) = \pi_t(\overline{PV}_t + u)\Delta_t + \mathbb{E}_{\overline{PV}_{t+1}}[\theta_{t+1}(b - u, \overline{PV}_{t+1})$$

4: Find u_t^* that minimizes the cost:

$$u_t^* = \arg\min_{u \in U_t}\{f(b_t, \overline{PV}_t)\}, \text{and } \theta_t(x_t) = \min_{u_t \in U_t}\{f(b_t, \overline{PV}_t)\}$$

5: Fill in the cell $(t, (b_t, \overline{PV}_t))$ of the action list with the best action u_t^*, fill in the cell $(t, (b_t, \overline{PV}_t))$ of the cost list with the optimal cost $J_t(x_t)$.
6: if $t = 1$, STOP; else $t = t - 1$, go to step 2.

We implemented the DP algorithm in Matlab 10.8. on a 2.4GHz Intel Core i5 processor. We first tried a small scale problem for testing purposes. Below is a summary of the data inputs:

1. battery size: 50 kW 2 hour, efficiency η=1;

2. time horizon = 24 hours;

3. $\sigma_t = 30$ for $t = 1, \cdots, T$; the mean of the PV output is shown in Figure 4.11.

4. at time T, the salvage cost of the power in the battery is \$0.1 kW

The DP algorithm is based on a set of discretizations. For this instance, we choose the following discretizations: The state space of the battery SOE assumes values from $\underline{b} = 0$ to $\bar{b} = 100$, discretized into 40 different values. The state space of the PV is also discretized into 40 different values. The maximum output level is set to be $\max_t\{\mu_t + 3\sigma_t\}$. For the action space, we discretize the action space into 40 different values.

The CPU time of the DP algorithm on this instance is 220.4 seconds. We test the result on a set of randomly generated instances which follow the distribution specified in the data input. Compared to the no-battery case, where we feed-in available PV to the

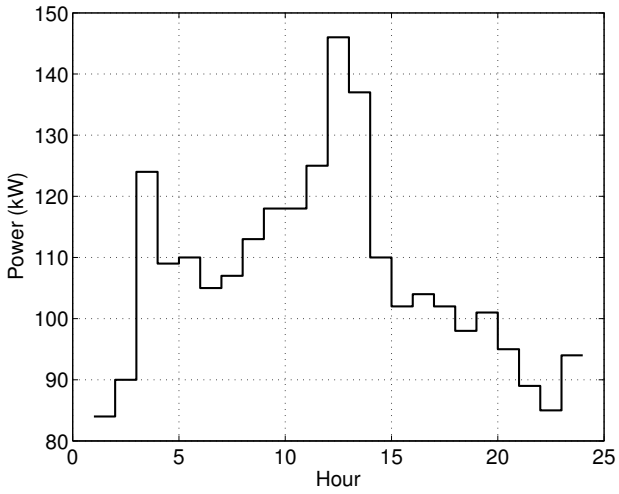


Figure 4.11: The Mean PV output for Small Scale Instance

grid, the overall revenue increase is 11.87%, which on average is $691.36 per instance. In Figure 4.12 we show the results for one day. We show the trend of the price in the same plot. It is clear from this plot that when the price is high, especially during periods 19 and 21, the battery discharges its power near the maximum rate to increase the feed-in power to the grid.

4.4.3 Incorporating Battery Efficiency

In Section 4.4.2 we studied the performance of a battery with perfect efficiency. We observe from the plots in Figure 4.12, that the battery sometimes makes a round-trip charge and discharge even if the price is flat. This is feasible in the perfect efficiency case since there is no cost associated with using the battery, and making a round trip when the price is flat won't hurt the objective. However, one can expect that this phenomenon will disappear when the battery efficiency becomes less than 1, since the a portion of the power will be lost due to the resistance of the battery itself. As a result, the system will charge or discharge the battery only if it is necessary.

We slightly modify the DP model to account for the battery efficiency. Let u be the internal power of the battery, and E be the terminal power of the battery. Their relationship is given below:

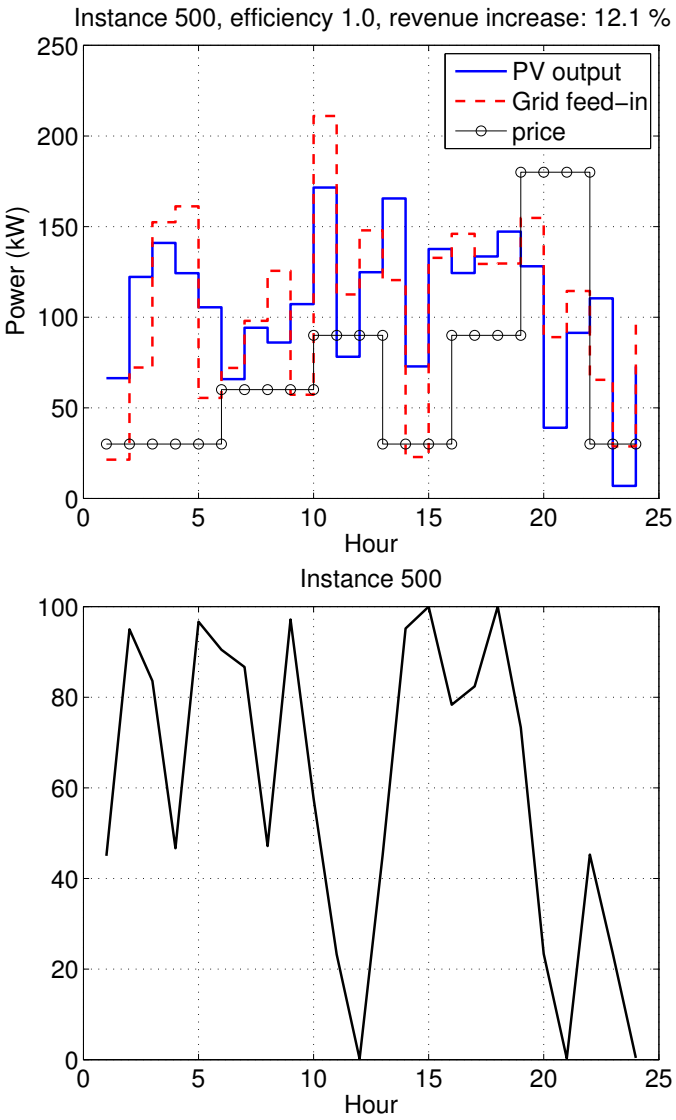


Figure 4.12: Feed-in comparison

$$\text{if } u < 0,\ E = u/\eta_c, \text{ else } E = u\eta_d.$$

The new DP model is shown below:

$$\theta_t(b, \overline{PV}) = \max_u \left\{ \pi_t(\overline{PV} + E)^+ \Delta_t + \mathbb{E}_{\overline{PV}_{t+1}}[\theta_{t+1}(b - u, \overline{PV}_{t+1})] \;\middle|\; \begin{array}{l} \underline{U} \le u \le \overline{U} \\ \underline{B} \le b - u\Delta_t \le \overline{B} \\ \overline{PV} + E \ge 0 \end{array} \right\} \tag{4.31}$$

We implement the DP algorithm according to this model for the instance we used in Figure 4.11. Here we choose $\eta_c = \eta_d = 0.9$. We show the result on trajectory 500 in Figure 4.13. It is clear from the result that even with a relatively good efficiency of 0.9, there are is no round trip actions during the period when the price is flat. Also one can see that the revenue increase dropped a little lower, from 12.1% to 12.0%. We also ran the DP algorithm with efficiency 0.9 on the same set of instances specified in Section 4.4.2. The results indicate an increase of 11.83%, compared to the perfect battery efficiency case of 11.87%; the difference is 0.04%.

It also can be interesting to study how a change in battery efficiency will affect the revenue increase. Figure 4.14 shows the relationship between the battery efficiency and the revenue increase. The revenue increase (in percentage) is not sensitive if $\eta > 0.6$.

4.4.4 Case Study

Next we study the performance of the dynamic programming model on a set of 20 instances. We use the same set of PV data from Section 4.3.5. The resolution of the deterministic case is 5 mins (288 periods per day), but in this case study, we choose the resolution to be 30 minutes, e.g., we record the power output of the PV farm every 30 minutes. We take the observed PV value to be the mean of the random PV output. We assume the standard deviation of the PV output is 20% of its mean value plus some random noise, $\sigma_t = 0.2\overline{PV}_t + \epsilon$, where $\epsilon \sim N(30, 10)$. For each instance, the price over time is generated from a uniform distribution $U(10, 40)$. In the case study, we test the performance of the DP algorithm on 3 different sizes of battery system: 250 kW, 1 hour system, 500 kW, 1 hour

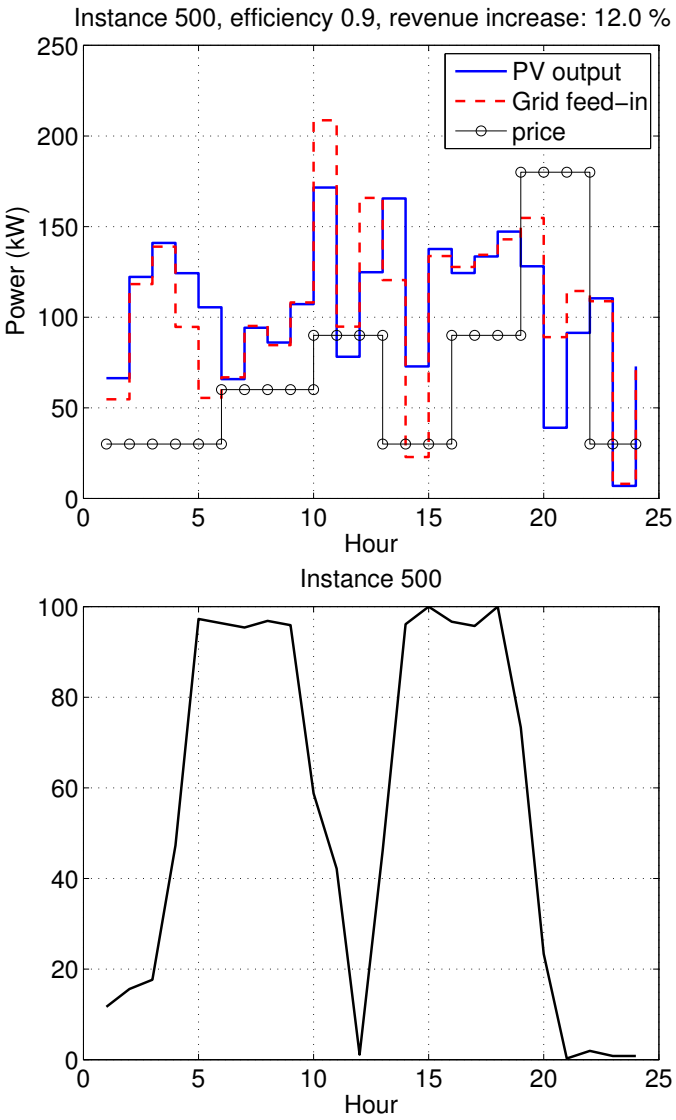


Figure 4.13: Feed-in comparison

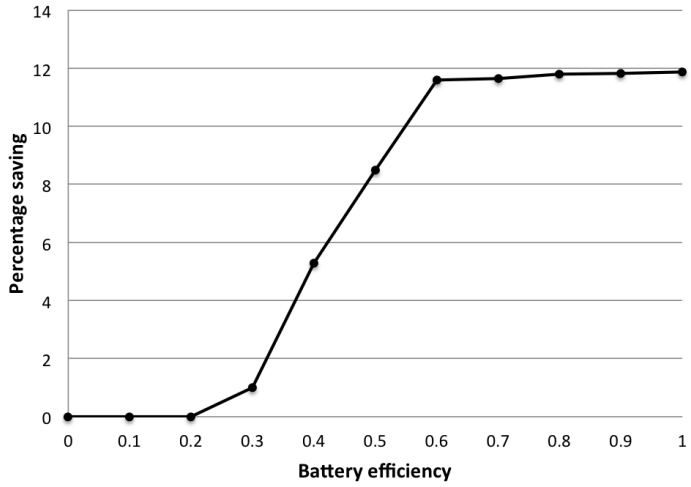


Figure 4.14: Revenue increase vs. battery efficiency

system and 500 kW, 2 hour system. The battery is assumed to have round trip efficiency of 0.8 and the the resistance is being accounted for during charge ($\eta_c = 0.8$, $\eta_d = 1$).

We choose the following discretizations: The state space of the battery SOE is discretized into 40 different values. The state space of the PV is also discretized into 40 different values. The action space is discretized into 40 different values.

For each instance, to study the performance of the DP algorithm, we randomly generate 500 PV output trajectories according to the assumptions on the mean and variance. Each trajectory represents a 24 hour, 30 minute resolution PV sample path. For each trajectory, we execute the optimal solution from the DP. We define the performance metric to be the following:

$$\rho_{k,j} = \frac{\sum_i \theta_k^{i,j} - \sum_i \theta_k^{i,n}}{\sum_i \theta_k^{i,n}}$$

where k is the index for the instances, j represents the size of the battery system, n means the case with no battery, and i is the index for different trajectories. $\theta_k^{i,j}$ is the total power sale revenue for trajectory i under battery size j from instance k. $\rho_{j,k}$ represents the percentage increase in the revenue after we apply the DP algorithm for instance k with battery size j.

We implement the case study in Matlab 10.8, on a 2.4GHz Intel Core i5 processor. The time depends largely on the discretization level. The average CPU time for each

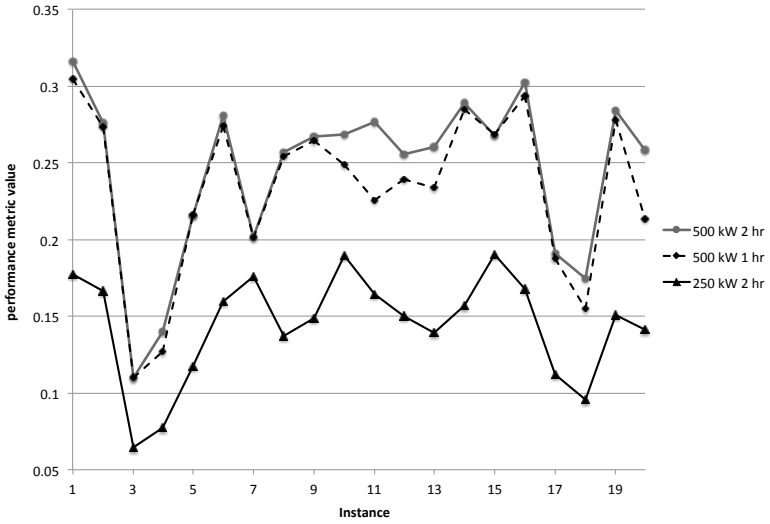


Figure 4.15: Performance of different batteries

instance is 182.30 seconds. In Figure 4.15 we plot the performance metric value for the 3 different battery systems. The black curve plots the the performance metric for the 250 kW, 2 hour system for each instance. The black dashed curve is for the 500 kW, 1 hour system and the grey curve is for the 500 kW, 2 hour system. The observation we make is that the 250 kW, 2 hour system achieves less than the 500 kW 1 hour system, about half of the revenue increase. This is interesting since the volume of the battery systems are the same, 500 kWh; however, they perform differently. The main reason is that the problem is of 30 minute resolution, and the battery is supposed to react to the difference in the price and choose actions accordingly. Here the maximum instantaneous power of the battery system is more important, and a one hour battery system is sufficient here. This explanation is further supported by another observation, that the performance of the 500 kW, 1 hour system is very similar to the performance of the 500 kW, 2 hour system. It is clear from this experiment that the sizing of the battery system plays a very important role in this problem. The optimal sizing of the battery system is out of the scope of this chapter.

4.4.5 Incorporating the Three Phase Phase Constraints

Now we consider the problem with the 3 phase constraints imposed by the grid operator. The original problem assumes that the PV operator may choose t_s, t_f as well as the reference value $Pref$ in advance. However, choosing optimal values for this decision variable is not straightforward in the stochastic case. Therefore, we first assume that they are fixed.

Let γ be the rate limit for ramp up and the ramp down phase ($0.6\%Pmax$ for the deterministic case), and let δ be the allowable level of oscillation during the qusai-stationary phase, i.e., the feed-in power is within the range $[Pref - \delta, Pref + \delta]$. ($\delta = 0.25\%Pmax$ for the deterministic case.)

The assumptions for developing the DP formulation for this problem are:

1. the choice of t_s and t_f is fixed;
2. if any of the constraints are violated, the penalty is proportional to the amount of violation, and the violation cost is h per kW;
3. to avoid large arbitrary violations, we assume the maximum feed-in level is mP;
4. during the qusai-stationary phase, the reference power feed-in level ($Pref$) is given.

To track whether the feed-in value is violated, we'll need to introduce an additional state variable $\hat{P}$, which accounts for the feed-in value of the previous time period. In the previous problem, since there is no constraint on the feed-in value, no PV will be wasted. In contrast, in this problem, since there is a penalty cost associated with feeding in too much or too little, not all the PV output will be utilized at optimality. Therefore the amount of PV used is a decision variable. We will start with a generic model and discuss the details later. According to the assumptions given above, we have the following DP model:

$$\theta_t(b, \overline{PV}, \hat{P}) = \max_{u,P} \left\{ \pi_t P \Delta_t - g(t, P, \hat{P}) + \mathbb{E}[\theta_{t+1}(b - u\Delta_t, \overline{PV}_{t+1}, P)] \;\middle|\; \begin{array}{l} \underline{U} \le u \le \overline{U} \ (a) \\ \underline{B} \le b - u\Delta_t \le \overline{B} \ (b) \\ P = PV + E \ (c) \\ 0 \le PV \le \overline{PV} \ (d) \\ 0 \le P \le mP \ (e) \end{array} \right\} \tag{4.32}$$

where the expectation is over $\overline{PV}_{t+1}$. For this problem the decision variables are the battery actions u and the power feed-in P. Once the choice of u and P is made, the incremental cost consists of two parts, the feed-in revenue, $\pi_t P \Delta_t$, and the violation cost, $g(t, P, \hat{P})$. The violation cost depends on the power feed-in level at time $t-1$ and the choice of feed-in level P. Now we show how the violation cost is computed.

$$g(t, P, \hat{P}) = \begin{cases} h\{(P - \hat{P} - \gamma\Delta)^+ + (P - \hat{P})^-\}, & t \le t_s \\ h\{(P - pref - \delta)^+ + (P - pref + \delta)^-\}, & t_s + 1 \le t \le t_f \\ h\{(P - \hat{P})^+ + (P - \hat{P} - \gamma\Delta)^-\} & t \ge t_f + 1 \end{cases} \tag{4.33}$$

Equation (4.33) states that for the ramp up or the ramp down phase, a penalty will be incurred if the system ramps up too fast or too slow. For the quasi stationary phase, a penalty will be incurred if the system goes below/over the band limit.

Compared to the model described in Section 4.4.1, this DP model has one additional state variable and one additional decision variable. Given the range of P and u, we are looking for (P^*, u^*) that provides the optimal cost value. Here we briefly discuss how to find feasible combinations of P and u.

In the constraints, the utilized power level PV is also a decision variable, but it is constrained by the P and u. Assume the feasible range of actions related to the battery constraints (4.32 a) and (4.32 b) is A_1. Now fix a power feed-in level specified in (4.32 e) (P_{fix}), then according to (4.32 c) and (4.32 d), there will be range of feasible battery actions; lets call this A_2. Therefore $A = A_1 \cap A_2$ is the a range for the feasible battery

actions. If $A \neq \emptyset$, then for any $u \in A$, we evaluate the value function and record the one with maximum cost. We repeat this for all possible $P_{fix} \in [0, mP]$ and record the optimal cost and the optimal u^* and P^*.

As in the previous model, both the battery SOE b and $\overline{PV}$ are needed as state variables for this problem. In addition, we also need $\hat{P}$ in the state variable to evaluate the penalty cost for the problem. Both b and $\overline{PV}$ represent the current condition, which carries no information from the past.

Incorporating the Violation Cost

Now we consider the case in which if there is any violation in the phase constraints, then there is no revenue earned from t to $t + \hat{t}$. Suppose there are k periods between time t and $\hat{t}$. We introduce another state variable, called flag, which takes values from $0, 1 \cdots, k-1$:

$$0, \text{ if there was no violation in periods } t-k+1, \cdots, t. \tag{4.34}$$

$$\text{flag} = \quad i, i = 1, \cdots k-1 : \text{if there was a violation in periods } t-k+1, \cdots, t, \tag{4.35}$$

$$\text{that lasted } i \text{ periods, } i = 1, \cdots, k-1 \tag{4.36}$$

The DP formulation for this problem is:

$$\theta_t(b, \overline{PV}, \hat{P}, \text{flag}) = \max_{u,P} \left\{ R_t + \mathbb{E}_{\overline{PV}_{t+1}}[\theta_{t+1}(b - u\Delta_t, \overline{PV}_{t+1}, P, \text{ new flag})] \;\middle|\; \begin{array}{l} \underline{U} \leq u \leq \overline{U} \ (a) \\ \underline{B} \leq b - u \leq \overline{B} \ (b) \\ P = PV + E \ (c) \\ 0 \leq PV \leq \overline{PV} \ (d) \\ 0 \leq P \leq mP \ (e) \end{array} \right\} \tag{4.37}$$

where R_t is the incremental revenue at time period t. Denote the cost inside the braces as f. Then at each time t, we have:

1. if flag =0, for any $P_{fix} \in [0, mP]$,

- $f = 0 + \mathbb{E}_{\overline{PV}_{t+1}}[\theta_{t+1}(b - u\Delta_t, \overline{PV}_{t+1}, P, 1)]$, if any of the phase constraint is violated at time t.
- $f = \pi_t P + \mathbb{E}_{\overline{PV}_{t+1}}[\theta_{t+1}(b - u\Delta_t, \overline{PV}_{t+1}, P, 0)]$ if no violation exists

2. if flag $= i$, $i = 1 \cdots k - 1$.

- $f = 0 + \mathbb{E}_{\overline{PV}_{t+1}}[\theta_{t+1}(b - u\Delta_t, \overline{PV}_{t+1}, P, (i + 1) \bmod k)]$ if no violation exists,
- $f = 0 + \mathbb{E}_{\overline{PV}_{t+1}}[\theta_{t+1}(b - u\Delta_t, \overline{PV}_{t+1}, P, 1)]$, if any of the phase constraint is violated at time t.

According to the above relationship, we can build a DP model under the new penalty condition. As one can see, the values of R_t and flag are affected by the value of P.

In this problem, we assume that the penalty lasts for exactly k periods. In the model we need a state variable that indicates whether there exists a penalty, or how long the penalty has lasted; therefore the system needs to have the "flag" as the state variable. In this case, the dimension of the state space is 4. Since the solution time increases sharply due to the curse of dimensionality, the model becomes less time efficient once the state space becomes large.

In this DP model, the choice of t_s, t_f and $Pref$ is given ahead of time. It is very difficult to evaluate those parameters optimally with the DP model. One could consider using a heuristic to find these values.

4.5 The Search of Structured Solutions

In the previous section, we discussed the stochastic problem. Solving the problem optimally while considering all the constraints is very difficult. We now consider whether there ever exists a structured solution if we relax some of the constraints. Let's go back to the problem setting in Section 4.4.1; that is, relax all the constraints set by the grid operator, with the goal of maximizing the power sold. We start with a discussion for the deterministic problem. We assume the battery has an efficiency of 1.

4.5.1 Deterministic Case

Under the deterministic setting, the problem to be solved is:

$$\begin{aligned} \text{maximize} \quad & \sum_t \pi_t (p_t + u_t)\Delta_t \\ \text{subject to} \quad & p_t + u_t \geq 0, \ \forall t = 1 \cdots, T, \\ & p_t \leq \overline{PV}_t, \ \forall t = 1 \cdots, T, \\ & \underline{B} \leq S_o - \sum_{j=1}^{t} u_j \Delta_j \leq \overline{B}, \ \forall t = 1 \cdots, T, \\ & \underline{U} \leq u_t \leq \overline{U}, \ \ \forall t = 1 \cdots, T, \end{aligned} \tag{4.38}$$

where p_t is the power sold the grid directly without routing through the battery. The first constraint is the no-power-sell back constraint. The second constraint makes sure the total power sold directly to the grid is less than the power available. The last two are the capacity constraint and the charge/discharge power rate constraint.

This is a linear programming problem. The objective is to optimize the overall feed-in level. Since all of the π_t's are positive, at optimality, we have $p_t = \overline{PV}_t$. Therefore it is equivalent to optimizing:

$$\begin{aligned} \text{maximize} \quad & \sum_t \pi_t u_t \Delta_t \\ \text{subject to} \quad & \overline{PV}_t + u_t \geq 0, \ t = 1 \cdots, T, \\ & \underline{B} \leq S_o - \sum_{j=1}^{t} u_j \Delta_t \leq \overline{B}, \ \forall t = 1 \cdots, T, \\ & \underline{U} \leq u_t \leq \overline{U}, \ \ \forall t = 1 \cdots, T, \end{aligned} \tag{4.39}$$

Proposition 1: If the first constraint in problem (4.39) is not violated, e.g., battery charge/discharge rate is relatively small compared to the PV output power, the optimal solution for problem 4.38 at each time instance is to charge at the maximum rate, discharge at the maximum rate or do nothing. Here the maximum charge rate is defined as $-\min\{(\overline{B} - S_t)/\Delta_t, -\underline{U}\}$, and maximum discharge rate is defined as $\min\{(S_t - \underline{B})/\Delta_t, \overline{U}\}$.

Proof: If the first constraint in problem (4.39) is satisfied, then at every time instance, the range of battery power is given by $u_t \in [-\min\{(\overline{B} - S_t)/\Delta_t, -\underline{U}\}, \min\{(S_t - \underline{B})/\Delta_t, \overline{U}\}]$, and the optimal u_t can always be achieved at one of the end points.

Note this is also true when the battery efficiency is less than 1. When the battery efficiency is incorporated, the problem becomes:

$$\begin{aligned}
\text{maximize} \quad & \sum_t \pi_t (p_t + E_t)\Delta_t \\
\text{subject to} \quad & PV_t + E_t \geq 0,\ t = 1 \cdots, T, && (a) \\
& p_t \leq \overline{PV}_t,\ t = 1 \cdots, T, && (b) \\
& \underline{B} \leq S_o - \sum_{j=1}^{t} u_j \Delta_t \leq \overline{B},\ \forall t = 1 \cdots, T, && (c) \\
& \underline{U} \leq u_t \leq \overline{U},\ \ \forall t = 1 \cdots, T, && (d) \\
& E_t = \eta_d u_t^+ - \frac{1}{\eta_c} u_t^-,\ \forall t = 1 \cdots, T, && (e) \\
& u_t = u_t^+ - u_t^-,\ \forall t = 1 \cdots, T, && (f) \\
& u_t^+, u_t^- \geq 0,\ \forall t = 1 \cdots, T, && (g)
\end{aligned} \tag{4.40}$$

which is equivalent to:

$$\begin{aligned}
\text{maximize} \quad & \sum_t \pi_t E_t \Delta_t \\
\text{subject to} \quad & PV_t + E_t \geq 0,\ t = 1 \cdots, T, && (a) \\
& \underline{B} \leq S_o - \sum_{j=1}^{t} u_j \Delta_j \leq \overline{B},\ \forall t = 1 \cdots, T, && (b) \\
& \underline{U} \leq u_t \leq \overline{U},\ \ \forall t = 1 \cdots, T, && (c) \\
& E_t = \eta_d u_t^+ - \frac{1}{\eta_c} u_t^-,\ \forall t = 1 \cdots, T, && (d) \\
& u_t = u_t^+ - u_t^-,\ \forall t = 1 \cdots, T, && (e) \\
& u_t^+, u_t^- \geq 0,\ \forall t = 1 \cdots, T, && (f)
\end{aligned} \tag{4.41}$$

where E_t is the battery's terminal power, and u_t is the battery's internal power. Constraint (4.41 d) captures the relationship between the terminal and internal power,

where η_d and η_c are the battery's charge and discharge efficiency ($0 < \eta_d \leq 1$, $0 < \eta_c \leq 1$). u_t^+, and u_t^- are the battery's discharge power and charge power (absolute value), respectively.

Lemma 1: At optimality, one of the following must be true: $u_t^- = 0$, $u_t^+ = 0$.

Proof: The objective is to maximize the total power feed-in. Suppose the battery internal power is fixed to u_t^o. If $u_t^o \geq 0$ (battery is discharging), then according to constraints (4.41 d) and (4.41 e), we have $E_t = (\eta_d - \frac{1}{\eta_c})u_t^- + u_t^o\eta_d$, since $\eta_d - \frac{1}{\eta_c} \leq 0$, to maximize E_t, we have $u_t^- = 0$, and $E_t = u_t^o\eta_d$. Similarly, we can prove tjat when $u_t^o \leq 0$, $u_t^+ = 0$ and $E_t = \frac{1}{\eta_c}u_t^o$.

Lemma 1 in general states that if $\eta_d < 1$ and $\eta_c < 1$, there is no optimal solution such that the battery charges and discharges at the same time.

Proposition 2: If the first constraint in problem (4.41 a) is not violated, e.g., the battery charge/discharge rate is relatively small compared to the PV output power, the optimal solution for problem (4.41) at each time instant is to charge at the maximum rate, discharge at the maximum rate or do nothing. Here the maximum charge rate is defined as $-\min\{(\overline{B} - S_t)/\Delta_t, -\underline{U}\}$, and the maximum discharge rate is defined as $\min\{(S_t - \underline{B})/\Delta_t, \overline{U}\}$.

Proof: If the battery rate is small, then the no-power-sell back constraint will always be satisfied. Then at every time instant, the range of battery internal power is given by $u_t \in [-\min\{(\overline{B} - S_t)/\Delta_t, -\underline{U}\}, \min\{(S_t - \underline{B})/\Delta_t, \overline{U}\}]$. According to Lemma 1, the range of battery terminal power is: $E_t \in [-\frac{1}{\eta_c}\min\{(\overline{B} - S_t)/\Delta_t, -\underline{U}\}, \eta_d \min\{(S_t - \underline{B})/\Delta_t, \overline{U}\}]$, and the optimal E_t can always be achieved at one of the end points.

If the battery power limits are comparable to the PV output level, then the optimal battery power is constrained not only by its own characteristics, but also by the PV power available, e.g., the amount of PV power output that can be injected into the battery.

Here we introduce two examples, one with a smaller battery, and the other with a larger battery. The PV output level and the feed-in price level are the same for these two instances. Table 4.3 provides a summary of the results for a small battery; in this case, the battery size is 30 kW, 2 hour. It is clear from the result that the battery either charges at

Period	Price ($)	PV output (kW)	Battery action (kW)	Battery SOE (kWh)
1	2.9	107	0	0
2	2	113	-30	30
3	2	118	-30	60
4	3	118	30	30
5	3	125	-30	60
6	3.8	146	30	30
7	6	137	30	0
8	1	110	-30	30
9	1	102	-30	60
10	3	104	30	30
11	3	102	0	30
12	3	98	-30	60
13	6	101	30	30
14	6	95	0	30
15	9	89	30	0
16	1	85	-30	30
17	1	94	0	30
18	1	94	30	0

Table 4.3: Data and solution for example with small battery

the rate limit or discharges at the rate limit or does nothing. The idea is to take advantage of the price differences and make battery decisions accordingly.

Table 4.4 includes results for a large battery which is 150 kW, 1 hour, and it is close to the PV output level. From this example, we see that at period 2, according to the battery characteristic, it can charge up to its limit of 150 kW; however, the PV output available is only 113 kW. This also happens in other periods; for example, in period 4, the battery could discharge 150 kW, but it doesn't. We see the actual action is to discharge 125 kW and charge 125kW in period 5 and therefore the battery is fully charged at the end in period 4. If we were to discharge at the maximum rate of 150 in period 4, the battery cannot get fully charged at period 5, since the PV power is limited. Therefore, in this case, the system not only needs to make decisions according to the price and battery characteristic, but also has to factor in the PV output availablity.

4.5.2 Stochastic Case

Now assume the PV output power is stochastic. We are especially interested in whether there exists any threshold structure for the battery inventory. We first examine the convexity of the value function for the a priori case, where the action is made before the PV

Period	Price ($)	PV output (kW)	Battery action (kW)	Battery SOE (kWh)
1	2.9	107	0	0
2	2	113	-113	113
3	2	118	-37	150
4	3	118	125	25
5	3	125	-125	150
6	3.8	146	0	150
7	6	137	150	0
8	1	110	-110	110
9	1	102	-40	150
10	3	104	150	0
11	3	102	-52	52
12	3	98	-98	150
13	6	101	95	55
14	6	95	-95	150
15	9	89	150	0
16	1	85	-85	85
17	1	94	-65	150
18	1	94	150	0

Table 4.4: Data and solution for example with large battery

output level is observed. We start off with this setting because the state space has only one dimension — the battery SOE level.

The DP model for this problem is:

$$\theta_t(b) = \max_{u \in U(b)} \left\{ \pi_t \mathbb{E}_{\overline{PV}_t} (\overline{PV}_t + u)^+ \Delta_t + \mathbb{E}_{\overline{PV}_t} [\theta_{t+1}(b - max\{u, -\overline{PV}_t\}\Delta_t)] \right\} \tag{4.42}$$

where

$$U(b) = [-\min\{(\overline{B} - b)/\Delta_t, -\overline{U}\}, \min\{(b - \underline{B})/\Delta_t, \underline{U}\}].$$

We need to see if there exists a base stock level for this problem. Let's assume the battery power salvage cost is 0 at the end of time T. Then at time T, the problem that we consider is

$$\theta_T(b) = \max_{u \in U(b)} \left\{ \pi_T \mathbb{E}_{\overline{PV}_T} (\overline{PV}_T + u)^+ \Delta_T \right\}. \tag{4.43}$$

Let $f_T(u) = \pi_t \Delta_t \mathbb{E}_{\overline{PV}_t} (\overline{PV}_t + u)^+ \Delta_T = \pi_t \Delta_T n(-u)$, where $n(x)$ is the probability loss function. Then

$$f_T'(u) = \pi_T \Delta_T (1 - F(-u)) \geq 0,$$

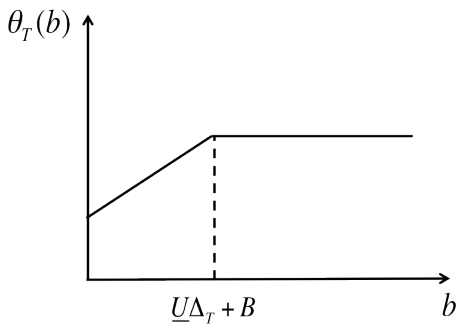


Figure 4.16: The revenue function at time T

and

$$f''_T(u) = \pi_T \Delta_T f(-u) \geq 0;$$

therefore $f_T(u)$ is non-decreasing and convex. And the maximum revenue at this stage is achieved when $u^* = \min\{(b - \underline{B})/\Delta_t, \underline{U}\}$. (discharge as much as possible). Therefore,

$$\theta_T(b) = \pi_T \Delta_T \mathbb{E}_{\overline{PV}_T}(\overline{PV}_T + u^*)^+.$$

Since both u^* and $\overline{PV}_T$ are non-negative,

$$\theta_T(b) = \pi_T \Delta_T \mathbb{E}_{\overline{PV}_T}(\overline{PV}_T + u^*) \tag{4.44}$$

$$\begin{aligned} \theta_T(b) &= \pi_T \Delta_T \mathbb{E}_{\overline{PV}_T}(\overline{PV}_T + u^*) \\ &= \begin{cases} \pi_T \Delta_T E(\overline{PV}_T) + \underline{U}\Delta_T, & \text{if } b \geq \underline{U}\Delta T + B \\ \pi_T \Delta_T E(\overline{PV}_T) + b - \underline{B}, & \text{otherwise} \end{cases} \end{aligned} \tag{4.45}$$

Now $\pi_T \Delta_T E(\overline{PV}_T) + \underline{U}\Delta_T$ is invariant with respect to the battery SOE level b, $\underline{U}$, and $\pi_T \Delta_T E(\overline{PV}_T) + b - \underline{B}$ is a linear function with respect to b. Therefore, $\theta_T(b)$ is a piecewise linear and concave function with respect to b. Figure 4.16 shows a plot of $\theta_T(b)$.

Now let's consider period $t = T - 1$. At this stage, the DP problem to be solved is:

$$\theta_{T-1}(b) = \max_{u \in U(b)} \left\{ \pi_{T-1} \Delta_{T-1} \mathbb{E}_{\overline{PV}_{T-1}}[(\overline{PV}_{T-1} + u)^+] + \mathbb{E}_{\overline{PV}_{T-1}}[\theta_T(b_{t+1})] \right\} \tag{4.46}$$

Where $b_{t+1} = b - max\{u, -\overline{PV}_{T-1}\}\Delta_{T-1}$.

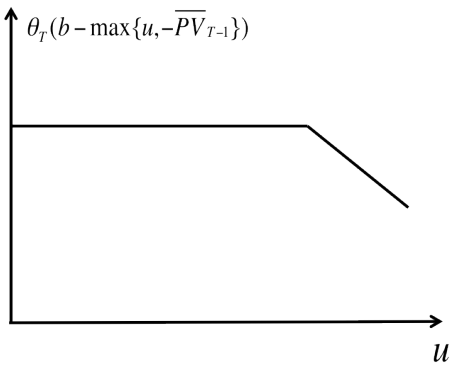


Figure 4.17: Future cost at time $T-1$ for fixed $\overline{PV}$

Let

$$f_{T-1,1}(u) = \pi_{T-1}\Delta_{T-1}\mathbb{E}_{\overline{PV}_{T-1}}(\overline{PV}_{T-1}+u)^+$$

and

$$f_{T-1,2}(u) = \mathbb{E}_{\overline{PV}_{T-1}}[\theta_T(b - max\{u, -\overline{PV}_{T-1}\}\Delta_{T-1})]$$

therefore

$$f_{T-1}(u) = f_{T-1,1}(u) + f_{T-1,2}(u).$$

by previous proof that $f_{T-1,1}$ is non-deceasing but convex with respect to u. What about $f_{T-1,2}(u)$? From the expression we have

$$\theta_T(b - max\{u, -\overline{PV}_{T-1}\}\Delta_{T-1}) = \begin{cases} \theta_T(b - u\Delta_{T-1}) & \text{if } u \geq \overline{PV}_{T-1} \\ \theta_T(b + \overline{PV}_{T-1}\Delta_{T-1}) & \text{if } u < \overline{PV}_{T-1} \end{cases} \tag{4.47}$$

By the previous proof that $\theta_T(x)$ is non-decreasing and concave, therefore,
$\theta_T(b - max\{u, -\overline{PV}_{T-1}\}\Delta_{T-1})$ is non-increasing and concave with respect to u. Figure 4.17 shows the plot of this function. $f_{T-1,2}$ is the expectation over the random variable $\overline{PV}_{T-1}$. Since concavity is preserved by expectation, $f_{T-1,2}$ is concave with respect to u. And f_{T-1} is a sum of a concave and a convex function. Therefore, f_{T-1} is neither convex nor concave in general, as shown in the example below.

Here we show a simple example of the behavior of $f_{T-1,1}$, $f_{T-1,2}$ and f_{T-1}. Let's consider a problem with $T = 2$, and assume the sell back prices at times 1 and 2 are 6 and

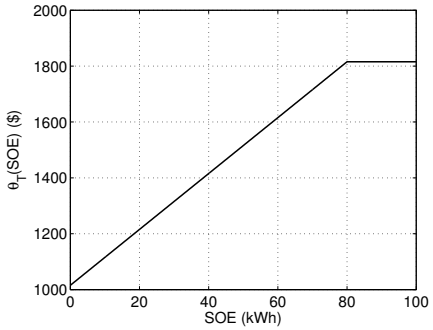


Figure 4.18: The revenue function at time T

10, respectively. Assume the PV profiles at time 1 and 2 are normally distributed with mean 20 and 100 and standard deviation is 10 and 30. Assume the battery is 80 kW, 100 kWh. The battery efficiency is 1. At time 2, we plot $\theta_T(b)$. At time 1, we assume a given battery SOE level $b = 60$, and we plot the functions $f_{T-1,1}$, $f_{T-1,2}$ and f_{T-1}.

Figure 4.18 shows the revenue at time T as a function of the battery SOE. This function is piecewise linear and concave as proved. Figure 4.19 plots the curves of $f_{T-1,1}$ and $f_{T-1,2}$ as a function of the feasible actions u (where u ranges from -60 kW to 40 kW in this case.) Clearly, $f_{T-1,1}$ is convex and $f_{T-1,2}$ is concave. The sum of these two curves is neither convex nor concave, as shown in Figure 4.20. The optimal battery pseudo action before observing the PV output is to charge for 20 kW. Note that it is not optimal to discharge the battery since the current price is lower than the future price, and it won't charge too much due to the limitation of the current PV profile. If the PV generation at time 1 turns out to be 30 kW, the deterministic case, the optimal decision is to charge 30 kW to the battery. However, in this example, according to Figure 4.19, the system will choose to charge 20 kW power into the battery. Therefore, similar optimal battery behavior is not true for stochastic case.

If the function can be proven to be concave, then there would exist an optimal SOE level b that maximizes $\theta(b)$. However, we prove that f is be neither convex nor concave in general. The above example shows that tracking the behavior of the objective value $\theta_t(\cdot)$ for different SOE levels b by examining the convexity of f is difficult.

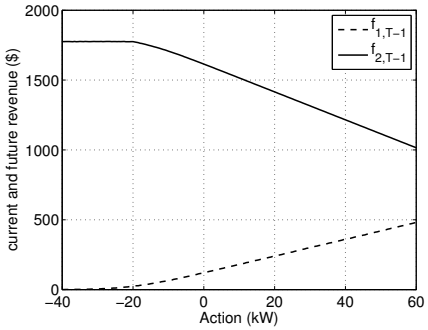


Figure 4.19: The current cost and future cost $T-1$, b=60 kWh

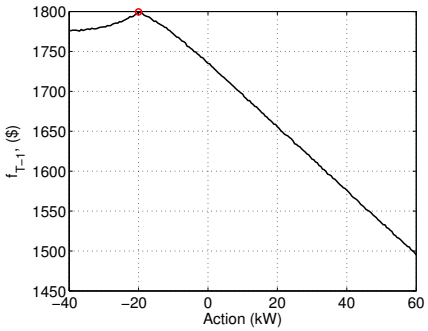


Figure 4.20: The revenue function at time $T-1$, b= 60 kWh

A New Direction

According to Zhou et al. [2014], if the electricity feed-in rate is also stochastic, it can be proved that there exists a dual-threshold inventory action structure. In this case, the DP formulation becomes:

$$\theta_t(b, \overline{PV}, \pi) =$$

$$\max_u \left\{ \pi(\overline{PV} + E)\Delta_t + \mathbb{E}[\theta_{t+1}(b - u\Delta_t, \overline{PV}_{t+1}, \pi_{t+1})] \;\middle|\; \begin{array}{l} \underline{U} \le u \le \overline{U} \ (a) \\ \underline{B} \le b - u\Delta_t \le \overline{B} \ (b) \\ \overline{PV} + E \ge 0 \ (c) \end{array} \right\} \quad (4.48)$$

where the expectation is over random variables $\overline{PV}_{t+1}$ and π_{t+1}. The authors prove that $\theta_t(b, \overline{PV}, \pi)$ is concave in b given any $\overline{PV}$ and π, and that there exists a dual threshold inventory actions, with thresholds $0 < \underline{X_t}(b_t, \pi_t) \le \overline{X_t}(b_t, \pi_t) < \underline{B}$, abbreviated as for $\underline{X_t}$,

$\overline{X_t}$. Define $y_t = x_t - u\Delta_t$, the thresholds are:

$$\underline{X_t} = \arg \max_{y_t \in [\underline{B}, \overline{B}]} \mathbb{E}[\theta_{t+1}(y_t, \overline{PV}_{t+1}, \pi_{t+1})|S_t] - \pi_t y_t/\eta_c \tag{4.49}$$

$$\overline{X_t} = \arg \max_{y_t \in [\underline{B}, \overline{B}]} \mathbb{E}[\theta_{t+1}(y_t, \overline{PV}_{t+1}, \pi_{t+1})|S_t] - \pi_t y_t \eta_d \tag{4.50}$$

The optimal action associated with the inventory position is as follows:

$$u(S_t) = \begin{cases} -\min\{\underline{X_t} - b\Delta_t, \overline{PV}_t \eta_c, \underline{U}\} & \text{if } 0 \le b \le \underline{X_t} \\ 0 & \text{if } \underline{X_t} \le b \le \overline{X_t} \\ -\max\{\overline{X_t} - b\Delta_t, \overline{U}\} & \text{if } b \ge \overline{X_t} \end{cases}$$

Note that the thresholds $\overline{X_t}$ $\underline{X_t}$ are functions of t, $\overline{PV}_t$, but not for b_t. And if $\eta_c = \eta_d = 1$, $\overline{X_t} = \underline{X_t}$.

Our problem assumes that the price is deterministic, which is a special case of the stochastic price. Therefore, similar results also hold for this problem. To demonstrate that, we give two different examples. (The proof can be deduct directly from Zhou et al. [2014], with a deterministic assumption on the price.) For the first example, the battery has perfect efficiency, $\eta_c = \eta_d = 1$; for the second example, the battery has imperfect efficiency, $\eta_c < 1$, $\eta_d < 1$.

Example 1: Battery with perfect efficiency

Assume the PV output is observed before the battery action is taken. Refer to Figure (4.11) for the mean of the PV output. The standard deviation of the PV output is 30% of the mean PV output. The price variation is given in Figure 4.21. We choose a fine discretization level. (Both SOE and PV value are discretized to 100 different values, and the battery action space is discretized to 50 different values.)

For some examples, multiple optimal solutions could exist. In practice, we observe that the optimal action changes dramatically if the state value differs slightly; this is counter-intuitive. We will give one example to explain this problem: assume the prices over time are [1,1,1,3], and the battery is 100 kW, 1 hour. The optimal action is to charge

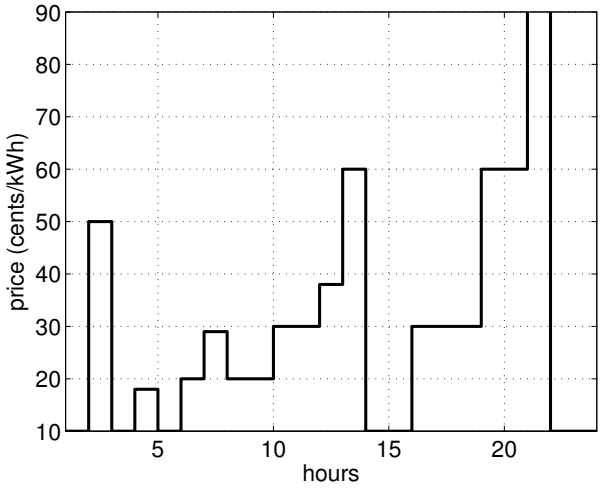


Figure 4.21: The price variation

the battery any time between time 1 and 3, and discharge the battery at hour 4 at its maximum rate of 100 kW. Therefore, the system may choose to charge the battery at time 2, and for time 1 and 3, there is no action, or the system can charge the battery at time 1 and do nothing at time 2 and 3. Clearly, there may be multiple solutions to the problem. Note this is true even if the battery has an imperfect efficiency. This could prevent us from observing any structured solutions or policies.

A good way around this is to introduce a discount factor γ to the future state b_{t+1}. The DP model under this setting is:

$$\theta_t(b, \overline{PV}) = \max_u \left\{ \pi_t \Delta_t \mathbb{E}_{\overline{PV}_t}(\overline{PV}_t + E) + \mathbb{E}_{\overline{PV}_t}[\theta_{t+1}(\gamma(b-u)\Delta_t, \overline{PV}_{t+1})] \;\middle|\; \begin{array}{l} \underline{U} \le u \le \overline{U} \\ \underline{B} \le b - u\Delta_t \le \overline{B} \\ \overline{PV} + E \ge 0 \end{array} \right\} \tag{4.51}$$

Where $E = u^+\eta_d - u^-\eta_c$.

After running the DP algorithm, for each time period, we plot the optimal battery action as a function of the battery SOE state values for a fixed PV value (120 kW). The

period	threshold value	period	threshold value
1	100	13	0
2	0	14	55
3	100	15	100
4	0	16	0
5	100	17	55
6	60	18	100
7	0	19	0
8	54	20	56
9	100	21	0
10	55	22	0
11	100	23	0
12	56	24	0

Table 4.5: A summary of the threshold values

plots are shown in Figure 4.22.

The threshold structure can be easily observed from the result. For example, when $t = 1$, the action is to charge the battery fully if possible. The threshold value is 100 kW. At time $t = 2$, we discharge the battery fully if possible, therefore, the threshold value is 0 kWh. This is not difficult to understand since the sellback price is \$50/kW, which is higher than that of time 1 and time 3. We also observe that for some time periods, the optimal battery inventory threshold is somewhere between fully charged and fully discharged. For example, when $t = 6$, the threshold is 60 kWh. Similar cases include period 10, 12 etc. Table 4.5 gives a summary of the threshold values.

This example shows that for the a posteriori case, under perfect battery efficiency, the single threshold structure also holds.

Example 2: Battery with imperfect efficiency

If the battery has imperfect efficiency, according to (4.49) and (4.50), the two threshold values are different, with $\underline{X}_t < \overline{X}_t$. We should be able to observe two different threshold values. Figure 4.24 provides a demonstration of the relationship between the next battery SOE state b_{t+1} and the current SOE state b_t given t and π_t. This plot shows that there exist two threshold values, a store-up-to level and sell-down-to level. In between these two threshold values, the inventory action is to do nothing. Next we show one example to support this claim.

We assume the same settings as in example 1. We modify the charge and discharge

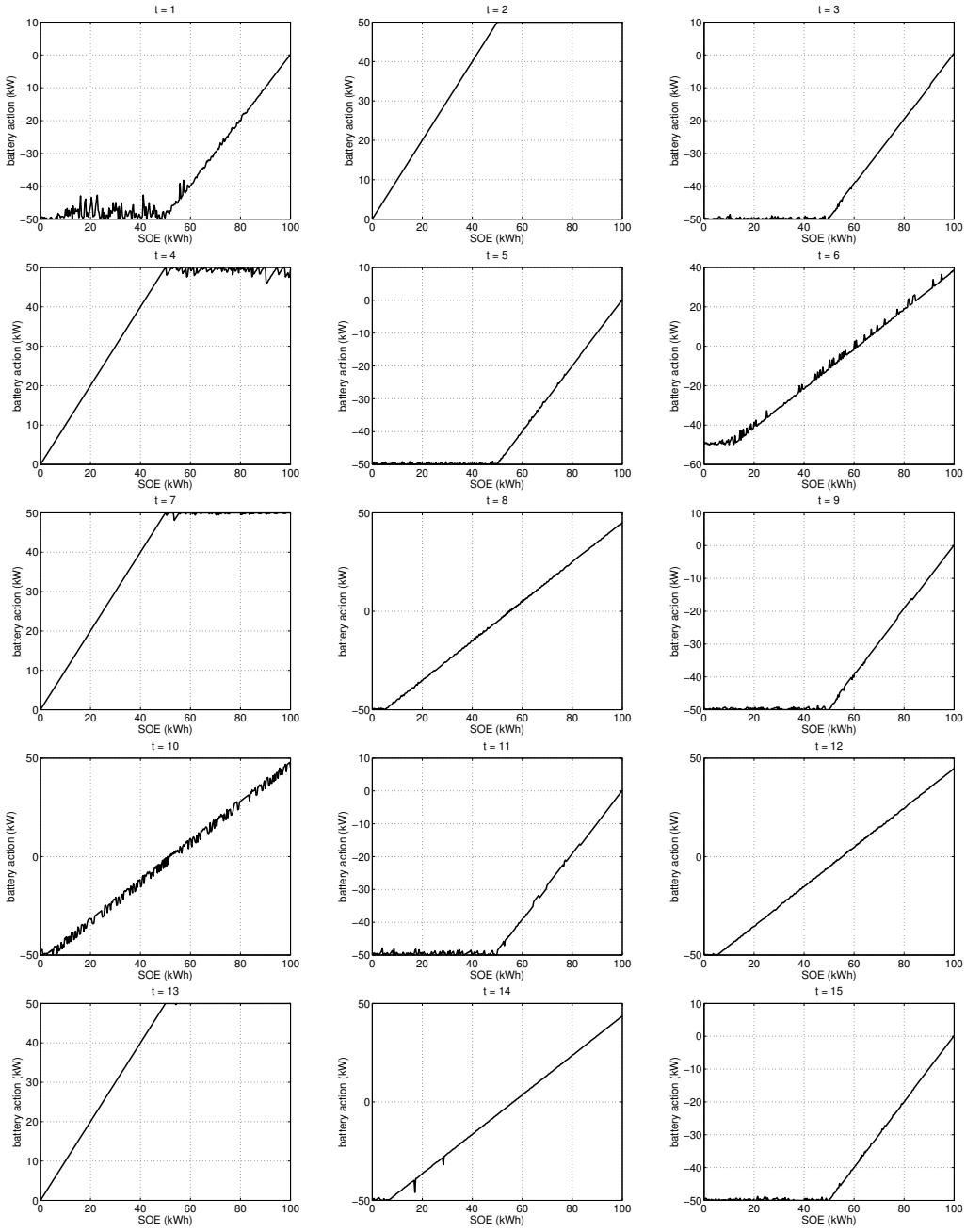


Figure 4.22: Demonstration of single threshold policy, $\gamma = 0.9$

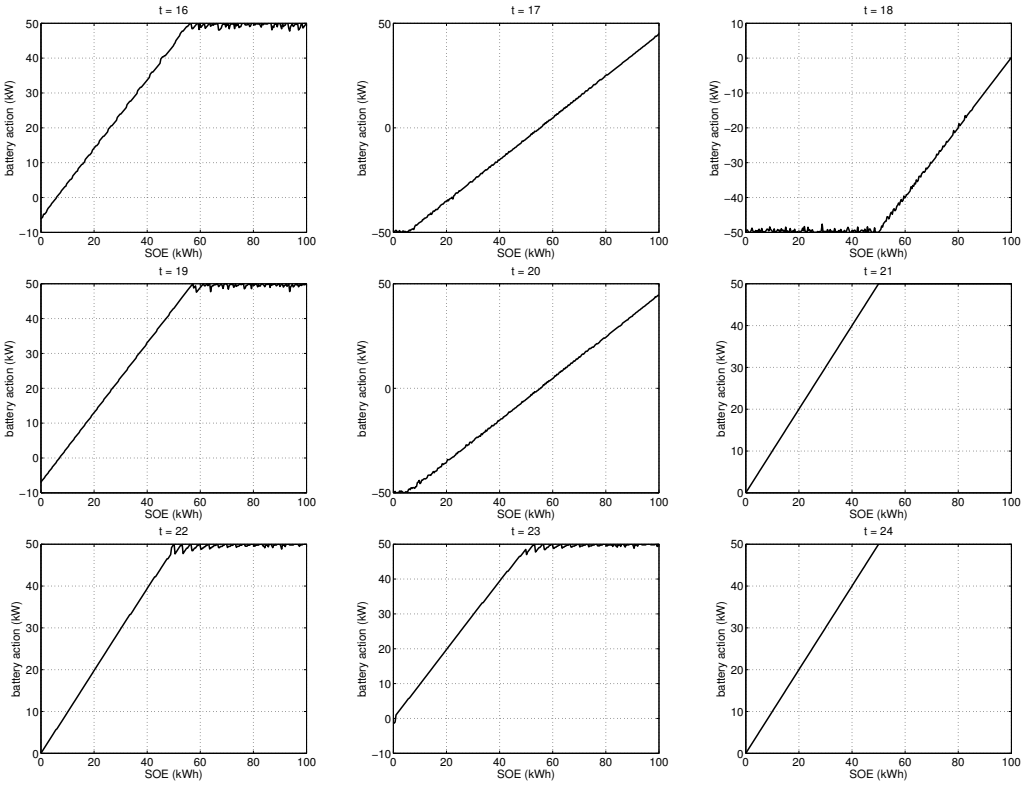


Figure 4.23: Demonstration of single threshold policy-con't.

efficiency to $\eta_c = 0.7$ and $\eta_d = 0.5$, respectively. For the same fixed PV value (120 kW), the relationship between battery SOE and optimal action is given in figure 4.25. Here the inventory behavior is very different from the previous example. For instance, at time period 1, if $b_t < 55$ kWh, the inventory action is to store up to 55 kWh; if $b_t > 55$ kWh, the battery action is to do nothing. In this case, we only observe the store-up-to and the do-nothing phase, and $\overline{X} = 55$ kWh. A similar phenomenon can be observed when $t = 5$. When $t = 12$, the optimal inventory action is to do nothing if the SOE is less than 55 kWh and discharge to 55 kWh if the SOE is greater than 55 kWh. In this case, there are two phases: do-nothing and sell-down-to $\overline{X} = 55$ kWh. Similar cases are shown for $t = 12$, 19 or 20. Note that cases are not observed when the battery has perfect efficiency.

To sum up, we argued that for the PV feed-in problem without the three phase constraints, there exists a threshold inventory action structure. When the battery has perfect efficiency, there is a single threshold value; and when the battery has imperfect efficiency, there exists a dual threshold structure. Note that the threshold value is not

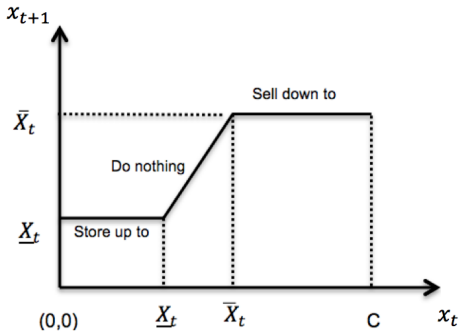


Figure 4.24: Demonstration of dual threshold structure

known beforehand. Computation is also needed in order to obtain the threshold values. For a 5 minute resolution problem, the system needs to determine $2 \times 12 \times 24 = 576$ parameters for a daily cycle.

4.6 Conclusions and Future Work

We study a PV feed-in problem with a storage system where the power purchaser imposes complex constraints on the power feed-in profile. We approach this problem under both deterministic settings and stochastic settings. Under the deterministic setting, we model this problem as a mixed integer programming (MIP) formulation. Two different variants of the formulation are introduced. We then evaluate the performance of these two variants on a set of instances, where the PV output level is obtained from CAISO. We show that the PV power feed-in can be enhanced significantly by a properly sized battery, and that formulating this problem with more constraints and variables can reduce the solution time. Under the stochastic setting, we first relax the constraints incurred by the power purchaser, and we study the performance of the DP algorithm under this setting. Later we show that when battery have perfect efficiency, the battery inventory has a single threshold structure, whereas a dual threshold structure holds when the battery has imperfect efficiency. In the end we study the case where the phase constraints are included, and provide a DP formulation for this problem.

Some avenues for future work include developing efficient/near optimal algorithms

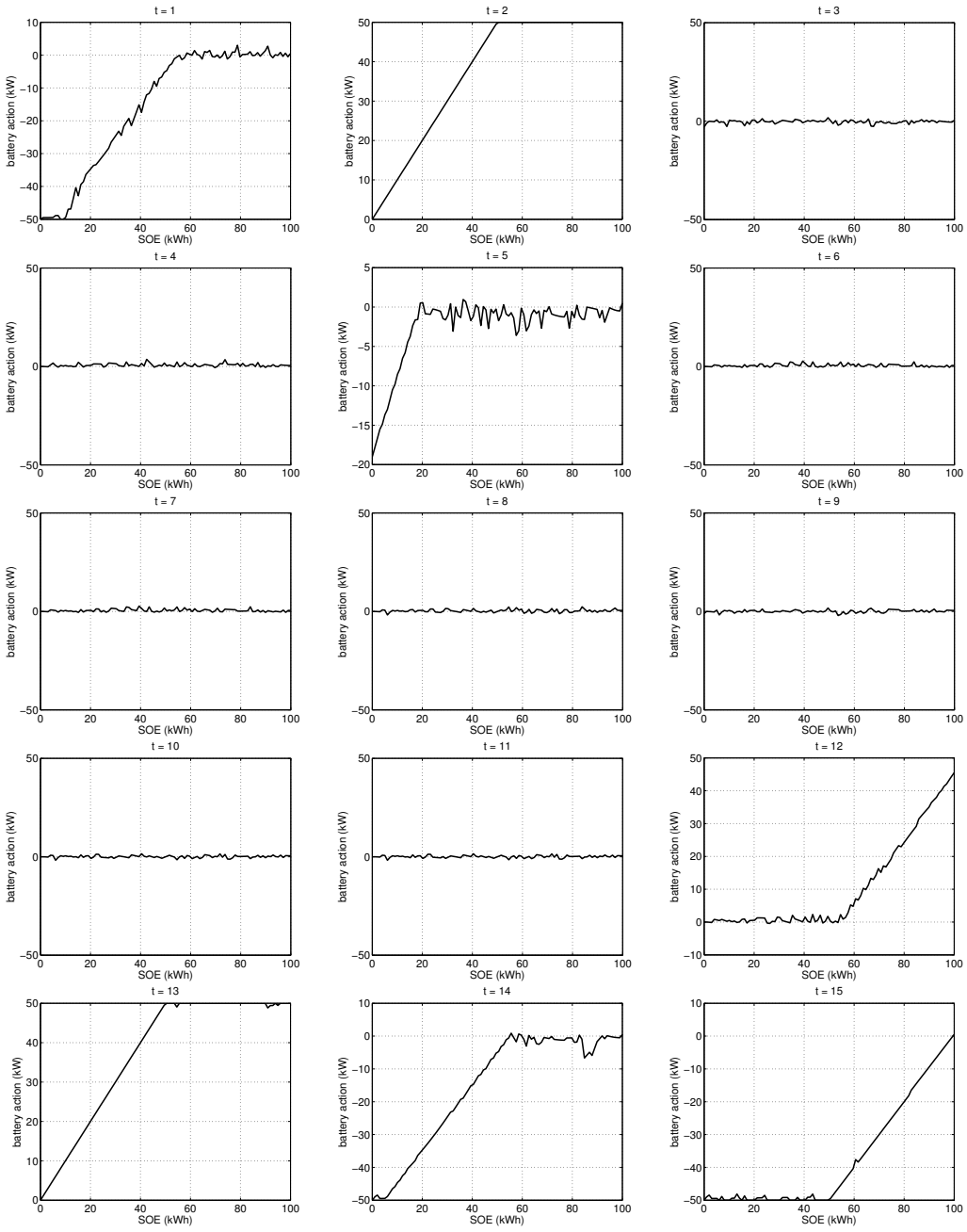


Figure 4.25: Demonstration of dual threshold policy, $\gamma = 0.9$

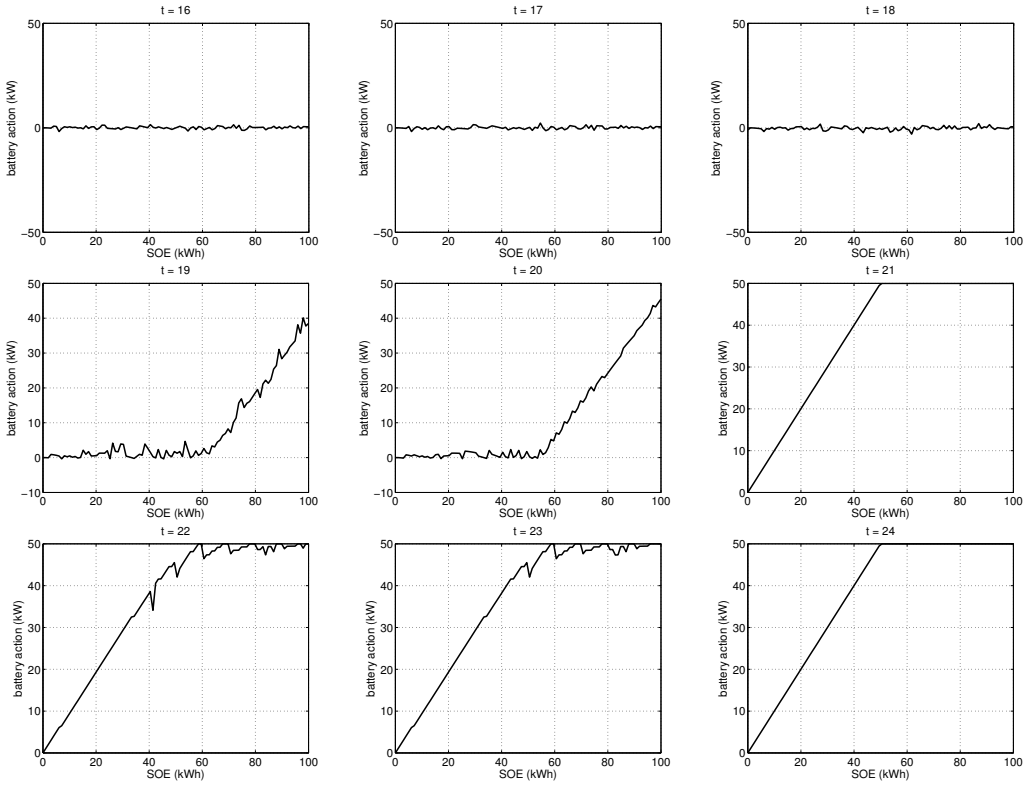


Figure 4.26: Demonstration of threshold policy-con't.

with respect to the 3 phase constraints for the stochastic case. It is also interesting to study the algorithm for real time implementation, and see how the algorithm reacts to changes in the PV inputs.

Chapter 5

Conclusions and Future Work

Today, the energy industry faces many looming challenges. One main challenge is the ability to increase energy efficiency. This requires efforts from two entities within the energy system — the rule/policy maker from the upper level who creates standards to properly regulate energy usage and energy trading processes, and the rule follower, who reacts to the rules or polices wisely to maximize its benefit taking into consideration economic and quality-of-life issues. Ideally, these two parties should coordinate well, and the rule follower is assumed to make decisions smartly enough. In reality, many such applications are not as straightforward, and making decisions without analyzing the problem systematically is very difficult. This thesis studies on three problems to help the rule follower increase its benefit in the energy system.

In the first problem, a power consumer aims to dispatch its power usage over time given different electricity price rate, appliance characteristics, and power limit constraints. That information is collected by an Energy Management Controller (EMC). This problem is challenging because that the number of appliances is large, the operation of each appliance is stochastic, and decisions have to be made very frequently. In this case, we propose an approximate dynamic programming (ADP) algorithm, which work well for small instances but cannot solve instances of realistic size sufficiently quickly. We also propose several scheduling policies for very fast solutions of large scale problems. We discover that sorting the requested appliances according their operating urgency improves the cost. This result

allows the power consumer to dispatch power usage properly and quickly.

Our work on the EMC problem provides a scheduling scheme for future smart grid systems. We work under the assumption that the power consumer will specify the delay penalty of each appliance. In reality, customers themselves may not have a clear sense about the delay penalties, or the penalties may not be large enough to make the algorithm effective. It would be helpful to create a model without having the delay penalties as inputs, but instead replacing them it with other more powerful measures. Our work also makes sure that the power usage is distributed more evenly without creating any power usage peaks or rebound peaks by specifying a power usage limit at each time period. It is worthwhile to understand how to set those limits automatically and effectively. In general, it would be interesting to understand whether in a facility with smart grid infrastructure, it is possible to optimally dispatch power usage to reduce energy payments while considering the loss of social welfare. What models/policies are more effective? What parameters are needed? What assumptions should be made to model the appliances? These are interesting questions to be studied in the future.

In the second problem, the power consumer needs to consider the electricity tariff imposed by the grid. To be more specific, a demand charge is incurred based on the maximum demand that happened within a billing cycle. In this case, a battery system is introduced to reduce the peak demand. The power consumer needs to decide the power dispatch plan within the entities, meaning how much to charge/discharge the battery at each period of time, and how much power to purchase directly from the grid in order to reduce the peak demand. The planning system we develop can assist the power consumer to make decisions under the load uncertainty. First, we develop a minimax dynamic programming model for the deterministic load problem. In light of the deterministic DP model, we then developed a stochastic DP model to solve the problem optimally. Later we developed a sample average approximation algorithm for real time implementation. The SAA Algorithm is demonstrated to solve the problem quickly while producing satisfactory peak reduction. We also introduce several naive algorithms for comparison with the DP and SAA Algorithm. In the end, we introduce a real-time SAA Algorithm and test its

performance on a data set of 1 year, 365 days. We observe that this algorithm is very effective in terms of generating real time power dispatch plans.

For the demand charge reduction problem, our work show that the demand charge can be effectively reduced with a battery storage system. In this project, we assume that the size of the battery system is given. From an economic standpoint, it would be helpful to consider the cost of the battery itself, e.g., purchase cost, maintenance cost and other cash flows in the long run, and design a system that captures the tradeoff between investment and return to optimally size the battery system. Another extension of this problem is to handle cases when loads are correlated, or during the time when loads cannot be forecasted properly. Is there an algorithm that is more robust and requires less information of the load profile?

In the third problem, we look at the solar power trading problem. The owner of the PV farm sells power to the grid operator. Standard PV power is subject to uncertainty. To better regulate the power feed-in profile, the grid operator has imposed complex constraints on the shape of the power profile; and a battery system needs to be deployed to buffer out the volatility of the PV output. In this case, the PV farm needs to plan the PV feed-in profile systematically, given the PV output uncertainty. We help the PV owner achieve the following tasks. First, we develop a Mixed Integer Programming model for the deterministic case, which assumes the PV output is fixed. In this model, we take into consideration all the constraints. The case study show that a 1 day, 5 minutes resolution problem can be solve by our model within minutes. Then we relax some of the constraints, and develop a dynamic programming model under this condition. We show that a threshold structure battery inventory solution exists for the simplified problem. Then we propos a dynamic programming model with respect to the key constraints from the grid operator. The level of complexity increases after we take into consideration more constraints.

This particular PV feed-in problem is challenging since the constraints imposed by the grid operator are complex and specific. In the deterministic setting, we solve it effectively by modeling it as a Mixed Integer Programming problem. In the stochastic setting, we relax the grid constraints and show that the relaxed problem has an optimal threshold

inventory structure. Because of the complexity of the constraints, solving the original problem optimally in the stochastic setting is a hard problem. A future direction could be developing a powerful algorithm that incorporates all the constraints in light of the threshold inventory structure. In this case, what assumptions need to be made? Do there exist any optimal battery inventory structures? How can we generalize the solution to a more generic PV power trading process? These are the questions that are attractive for future research.

Bibliography

California iso solar output profile, 2014. URL `http://www.caiso.com/green/renewableswatch.html`.

2014. URL `http://www.eia.gov`.

Daniel Adelman and Canan Uçkun. Smart homes with price-responsive thermostats. Technical report, University of Chicago, 2013.

R. Baldick, S. Kolos, and S. Tompaidis. Interruptible electricity contracts from an electricity retailer's point of view: Valuation and optimal interruption. *Operations Research*, 54 (4):627–642, 2006.

Oded Berman and Gabriel Y. Handler. Optimal minimax path of a single service unit on a network to nonservice destinations. *Transportation Science*, 21(2):115–122, 1987.

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Belmont, MA, 1995.

Bogdan S. Borowy and Ziyad M. Salameh. Methodology for optimally sizing the combination of a battery bank and pv array in a wind/pv hybrid system. *Energy conversion, ieee transactions*, 11(2):367–375., 1996.

T. H. Chang, M. Alizadeh, and A. Scaglione. Coordinated home energy management for real-time power balancing. In *IEEE Power and Energy Society General Meeting*, 1988.

Fang Chen, Lawrence V. Snyder, and Shalinee Kishore. Efficient algorithms and policies for demand response scheduling. *Energy engineering, ASCE*, 2014.

Claudio G. Codemo, Tomaso Erseghe, and Andrea Zanella. Energy storage optimization strategies for smart grids. *Communications (ICC), 2013 IEEE International Conference on. IEEE*, 2013.

Antonio J. Conejo, Miguel A. Plazas, Rosa Espinola, and Ana B. Molina. Day-ahead electricity price forecasting using the wavelet transform and ARIMA models. *IEEE Transactions on Power Systems*, 20(2):1035–1042, 2005.

Antonio J. Conejo, Juan M. Morales, and Luis Baringo. Real-time demand response model. *Smart Grid, IEEE Transactions*, 1(3):236–242, 2010.

Anna Danandeh, Long Zhao andBo Zeng, and Mehrnaz Abdollahian. Optimal job scheduling with day-ahead price and random local distributed generation: A two-stage robust approach. *IEEE PES Innovative Smart Grid Technologies Conference*, 2012.

Muhamad Zalani Daud, Azah Mohamed, and M. A. Hannan. An improved control method of battery energy storage system for hourly dispatch of photovoltaic power sources. *Energy Conversion and Management*, 73:256–270, 2013.

Peter Van de Ven, Nidhi Hedge, Laurent Massoulié, and Theodoros Salonidis. Optimal control of residential energy storage under price fluctuations. *ENERGY*, 2011.

Boris Defourny, Hugo P. Simao, and Warren B. Powell. Robust forecasting for unit commitment with wind. *System Sciences (HICSS), 46th Hawaii International Conference, IEEE*, pages 2337–2344, 2013.

S. Diaf, D. Diaf, M. Belhamel, M. Haddadi, and A. Louche. A methodology for optimal sizing of autonomous hybrid pv/wind system. *Energy Policy*, 35(11):5708–5718, 2007.

S. Diaf, D. Diaf, M. Belhamel, M. Haddadi, and A. Louche. Technical and economic assessment of hybrid photovoltaic/wind system with battery storage in corsica island. *Energy Policy*, 36(2):743–754, 2008.

Xuzhu Dong, Guannan Bao, Zhigang Lu, Zhichang Yuan, and Chao Lu. Optimal battery

energy storage system charge scheduling for peak shaving application considering battery lifetime. *In Informatics in Control, Automation and Robotics*, pages 211–218, 2012.

Abraham Ellis, David Schoenwald, Jon Hawkins, Steve Willard, and Brian Arellano. Pv output smoothing with energy storage. *Photovoltaic Specialists Conference (PVSC), 38th IEEE*, pages 001523–001528, 2012.

André Even, Jan Neyens, and A. Demouselle. Peak shaving with batteries. In *Electricity Distribution,CIRED. 12th International Conference on, IET*, pages 5–17, 1993.

Hicham Fakham, Di Lu, and Bruno Francois. Power control design of a battery charger in a hybrid active pv generator for load-following applications. *Industrial Electronics, IEEE Transactions*, 58(1):85–94, 2011.

Lingwen Gan, Adam Wierman, Ufuk Topcu, Niangjun Chen, and Steven H. Low. Real-time deferrable load control: handling the uncertainties of renewable generation. *Fourth International Conference on Future Energy Systems, ACM*, pages 131–124, 2013.

Samuele Grillo, Mattia Marinelli, Stefano Massucco, and Federico Silvestro. Optimal management strategy of a battery-based storage system to improve renewable energy integration in distribution networks. *Smart Grid, IEEE Transactions*, 3(2):950–958, 2012.

Longbo Huang, Jean Walrand, and Kannan Ramchandran. Optimal demand response with energy storage management. *In Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on, IEEE*, pages 61–66, 2012.

Matthew P. Johnson, Amotz Bar-Noy, Ou Liu, and Yi Feng. Energy peak shaving with local storage. *Sustainable Computing: Informatics and Systems 1*, 3:177–188, 2011.

Jae Ho Kim and Warren B. Powell. Optimal energy commitments with storage and intermittent supply. *Operations research*, 59(6):1347–1360, 2011.

Shalinee Kishore and Lawrence V. Snyder. Control mechanisms for residential electricity demand in SmartGrids. *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 443–448, 2010.

Koutsopoulos, Iordanisand Vassiliki Hatzi, and Leandros Tassiulas. Optimal energy storage control policies for the smart power grid. *Smart Grid Communications (SmartGridComm), IEEE International Conference on, IEEE*, pages 475–480, 2011.

Matt Kraning, Yang Wang, Ekine Akuiyibo, and Stephen Boyd. Operation and configuration of a storage portfolio via convex optimization. *18th IFAC World Congress*, 2011.

Na Li, Lijun Chen, and Steven H. Low. Optimal demand response based on utility maximization in power networks. In *IEEE PES General Meeting*, July 2011a.

Na Li, Lijun Chen, and Steven H. Low. Optimal demand response based on utility maximization in power networks. *In Power and Energy Society General Meeting, 2011 IEEE*, pages 1–8, 2011b.

Xiangjun Li, Hui Dong, and Xiaokang Lai. Battery energy storage station (bess)-based smoothing control of photovoltaic (pv) and wind power generation fluctuations. *Sustainable Energy, IEEE Transactions*, 4(2):464–473, 2013.

Y. Liang and Z. J. M. Shen. Stochastic control for smart grid users with flexible demand. Working paper, September 2011.

Jeff Linderoth, Alexander Shapiro, and Stephen Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241, 2006.

Douglas K. Maly and Khang-Shen Kwan. Optimal battery energy storage system (bess) charge scheduling with dynamic programming. In *IEEE Proceedings-Science, Measurement and Technology 142*, volume no. 6, pages 453–458, 1995.

A. Mohsenian-Rad, V. W. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia. Optimal and autonomous incentive-based energy consumption scheduling algorithm for smart grid. In *IEEE Innovative Smart Grid Technologies (ISGT)*, Jan 2010a.

A. H. Mohsenian-Rad and A. Leon-Garcia. Optimal residential load control with price

prediction in real-time electricity pricing environments. *IEEE Transactions on Smart Grid*, 1(2):120–133, 2011.

Amir-Hamed Mohsenian-Rad, Vincent W.S. Wong, Juri Jatskevich, Robert Schober, and Alberto Leon-Garcia. Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid. *Smart Grid, IEEE Transactions*, 1(3):320–331, 2010b.

NYISO. New york independent system operator, 2012. URL `http://www.nyiso.com/public/markets_operations/market_data/pricing_data/index.jsp`.

D. O'Neill, M. Levorato, A. Goldsmith, and U. Mitra. Residential demand response using reinforcement learning. In *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 409–414, 2005.

Alexandre Oudalov, Daniel Chartouni, Christian Ohler, and Gerhard O. Linhofer. Value analysis of battery energy storage applications in power systems. *Power Systems Conference and Exposition*, pages 2206–2211, 2006.

Alexandre Oudalov, Rachid Cherkaoui, and Antoine Beguin. Sizing and optimal operation of battery energy storage system for peak shaving application. *Power Tech, IEEE Lausanne*, pages 621–625, 2007.

Peter Palensky and Dietmar Dietrich. Demand side management: Demand response, intelligent energy systems, and smart loads. *Industrial Informatics, IEEE Transactions*, 7 (3):381–388, 2011.

PGE. Electricity tariff, 2013. URL `http://www.pge.com/tariffs/ERS.SHTML#ERS`.

Warren B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley-Interscience, 2001.

Anindita Roy, Shireesh B. Kedare, and Santanu Bandyopadhyay. Optimum sizing of wind-battery systems incorporating resource uncertainty. *Applied Energy*, 87(8):2712–2727, 2010.

Daniel F. Salas and Warren B. Powell. Benchmarking a scalable approximate dynamic programming algorithm for stochastic control of multidimensional energy storage problems. *Working Paper,Department of Operations Research and Financial Engineering, Princeton, NJ*, 2013.

Tomonobu Senjyu, Manoj Datta, Atsushi Yona, Toshihisa Funabashi, and Chul-Hwan Kim. Pv output power fluctuations smoothing and optimum capacity of energy storage system for pv power generator. *International Conference on Renewable Energies and Power Quality (ICREPQ08)*, 2008.

Shengnan Shao, Tianshu Zhang, Manisa Pipattanasomporn, and Saifur Rahman. Impact of TOU rates on distribution load shapes in a smart grid with PHEV penetration. In *IEEE PES Transmission and Distribution Conference and Exposition*, Apr 2010.

Alexander Shapiro. Monte carlo simulation approach to stochastic programming. *33nd Conference on Winter Simulation, IEEE Computer Society*, pages 428–431, 2001.

Zhen Shu and Panida Jirutitijaroen. Optimal operation strategy of energy storage system for grid-connected wind power plants. *IEEE Transactions on Sustainable Energy*, 1(1), 2014.

G. K. Singh. Solar power generation by pv (photovoltaic) technology: a review. *Energy*, 53:1–13., 2013.

Chua-Liang Su and Daniel Kirschen. Quantifying the effect of demand response on electricity markets. *Power Systems, IEEE Transactions*, 24(3):1199–1207, 2009.

Anand Subramanian, Manuel Garcia, Alejandro D. Domínguez-García, Duncan S. Callaway, Kameshwar Poolla, and Poolla Varaiya. Real-time scheduling of deferrable electric loads. In *IEEE American Control Conference (ACC)*, pages 3643–3650, Jun 2012.

Chee Wei Tan, Tim C. Green, and Carlos A. Hernandez-Aramburo. A stochastic method for battery sizing with uninterruptible-power and demand shift capabilities in pv (photovoltaic) systems. *Energy*, 35(12):5082–5092., 2010.

Sercan Teleke, Mesut E. Baran, Subhashish Bhattacharya, and Alex Q. Huang. Rule-based control of battery energy storage for dispatching intermittent renewable sources. *Sustainable Energy, IEEE*, 1(3):117–124, 2010.

U. S. Department of Energy. Estimating appliance and home electronic energy use, 2013. http://energy.gov/energysaver/articles/estimating-appliance-and-home-electronic-energy-use.

Sergio Vazquez, Srdjan M. Lukic, Eduardo Galvan, Leopoldo G. Franquelo, and Juan M. Carrasco. Energy storage systems for transport and grid applications. *Industrial Electronics, IEEE Transactions*, 57(12):3881–3895, 2010.

Chandu Venu, Yann Riffonneau, Seddik Bacha, and Yahia Baghzouz. Battery storage system sizing in distribution feeders with distributed photovoltaic system. *PowerTech, 2009 IEEE Bucharest, IEEE*, pages 1–5, 2009.

G. Xiong, C. Chen, S. Kishore, and A. Yener. Residential demand response using reinforcement learning. In *IEEE PES Innovative Smart Grid Technologies (ISGT)*, 2011.

Yangfang Zhou, Alab Scheller-Wolf, Nichola Secomandi, and Stephen Smith. Managing wind-based electricity generation with storage and transmission capacity. *SSRN eLibrary*, 2014.

Biography

Fang Chen is a Ph.D. candidate in the Department of Industrial and Systems Engineering at Lehigh University. She received her B.S. degree in Transportation Engineering in 2008 from Beijing Institute of Technology, and her M.S. degree in Mathematics and Statistics in 2010 from University of Minnesota Duluth. She joined Lehigh University in 2010. Her research focuses on mathematical programming with applications to energy systems and supply chain management. She is a member of INFORMS and IEEE, and the Lehigh University INE cluster. Fang will join Bosch North America as a Control and Optimization Engineer.