8-14-2015

Energy Management and Privacy in Smart Grids

Sergio Alfonso Salinas Monroy

Energy management and privacy in smart grids

By

Sergio A. Salinas Monroy

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Electrical and Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

August 2015

Energy management and privacy in smart grids

By

Sergio A. Salinas Monroy

Approved:

Pan Li
(Major Professor)

Qian (Jenny) Du
(Committee Member)

James E. Fowler
(Committee Member and Graduate Coordinator)

Yong Fu
(Committee Member)

Erdem Topsakal
(Committee Member)

Jason M. Keith
Interim Dean
Bagley College of Engineering

Name: Sergio A. Salinas Monroy

Date of Degree: August 14, 2015

Institution: Mississippi State University

Major Field: Electrical and Computer Engineering

Major Professor: Dr. Pan Li

Title of Study: Energy management and privacy in smart grids

Pages of Study: 124

Candidate for Degree of Doctor of Philosophy

Despite the importance of power systems in today's societies, they suffer from aging infrastructure and need to improve the efficiency, reliability, and security. Two issues that significantly limit the current grid's efficient energy delivery and consumption are: load-following generation dispatch, and energy theft. A load-following generation dispatch is usually employed in power systems, which makes continuous small changes so as to account for differences between the actual energy demand and the predicted values. This approach has led to an average utilization of energy generation capacity below 55% [49]. Moreover, energy theft causes several billion dollar losses to U.S. utility companies [31] [16], while in developing countries it can amount to 50% of the total energy delivered [48]. Recently, the Smart Grid has been proposed as a new electric grid to modernize current power grids and enhance its efficiency, reliability, and sustainability. Particularly, in the Smart Grid, a digital communication network is deployed to enable two-way communications between users and system operators. It thus makes it possible to shape the users' load demand curves by means of demand response strategies. Additionally, in the Smart Grid, traditional meters will be replaced with cyber-physical devices, called smart meters, capable of recording and transmitting users' real-time power consumption. Due to their monitoring capabilities, smart meters offer a great opportunity to detect energy theft in smart grids, but also raise serious concerns about users' privacy. In this dissertation, we design optimal load scheduling schemes to enhance system efficiency, and develop energy theft detection algorithms that can preserve users' privacy.

DEDICATION

To my parents.

ACKNOWLEDGEMENTS

I would like to give my sincerest gratitude to Prof. Pan Li, my advisor, whose guidance, and encouragement, have been crucial to my Ph.D study. Much of the work presented herein is a result of intellectual and enlightening discussions I had with him over the last few years. He is an outstanding mentor and researcher, from whom I have learned not only how to do research but also lifelong lessons.

I also would like to acknowledge my other committee members, Prof. Jenny Q. Du, Prof. James E. Fowler, Prof. Yong Fu, and Prof. Erdem Topsakal, for serving on my supervisory committee and for their help during my Ph.D. study.

I would like to extend my thanks to all my colleagues in our Networking, Energy, Security and big daTa (NEST) group for providing me with an encouraging research environment, and for their friendship and collaboration. I would like to specially acknowledge my colleagues and good friends, Sheng Cai, Xuhui Chen, Jinlong Ji, Ming Li, Changqing Luo, Weixian Liao, Arun Thapa, and Kaijin Zhang, for many valuable discussions and all the good memories.

Finally, I owe a special debt of gratitude to my parents who have made enormous sacrifices for me and my brother during the earlier years of our lives, and more importantly have always been there when we needed them the most. I owe all of my accomplishments to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Electric power systems were first installed as a luxurious novelty in the 1880's providing electric power to only a few people. Since then the power grid has been expanded to reach almost every person on the planet, and it is essential to support critical systems, such as telecommunications networks, stock markets, and health care facilities. Despite the importance of power systems in today's societies, they suffer from aging infrastructure and need to improve the efficiency, reliability, and security. Two issues that significantly limit the current grid's efficient energy delivery and consumption are: load-following dispatch, and energy theft.

Specifically, the load demand in a power system depends on users' daily activities which is very dynamic and exhibits a peak-valley pattern. A load-following generation dispatch is usually employed where power plants are turned on and off according to load forecasts, and then minor adjustments are continuously made to account for differences between the actual demand and the predicted values. For this approach to be feasible, enough generation capacity is required to be available to meet the peak load plus a security margin, which has led to an average utilization of energy generation capacity below 55% [49]. Besides, energy users are usually several hundred miles away from power plants, which results in a significant amount of energy loss due to transmission inefficiencies. The Energy Information Administration estimates that these losses amount to 7% of the total generated energy in the U.S. [55].

Another significant problem in power systems is energy theft, which causes several billion dollar losses to U.S. utility companies [31] [16], and in developing countries it can amount to 50% of the total energy delivered [48]. Legitimate users are also affected in the sense that utility companies impose higher energy rates to amortize losses due to energy theft. In addition to financial losses, energy theft enables criminal activities, such as illegal substance production [16], and jeopardizes the stability of the power system.

Recently, the Smart Grid (SG) has been proposed as a new electrical grid to modernize current power grids and enhance its efficiency, reliability, and sustainability. Specifically, in the SG, a digital communication network is deployed to enable two-way communications between users and system operators. It thus makes it possible to shape the users' load demand curves by means of demand response (DR) strategies, i.e., to encourage customers to change their usual electricity consumption patterns by financial incentives. One such strategy is real time pricing (RTP), in which system operators charge users a price that varies according to real-time energy generation cost. Since usually generation cost increases as the amount of generated energy increases, users may want to shift their load demands from peak hours to other times. Therefore, RTP can reduce the peak-hour load demand in the power system, which in turn lowers the requirement on system generation capacity. Another key feature of the smart grid is distributed generation (DG), where users install and take advantage of renewable generation resources, and energy storage devices.

Thus, DG can help reduce the energy loss due to transmission inefficiencies and alleviate congestion during peak hours.

Moreover, in the Smart Grid, traditional meters will be replaced with cyber-physical devices, called smart meters, capable of recording and transmitting users' real-time power consumption. Due to their monitoring capabilities, smart meters offer a great opportunity to detect pirate users in smart grids. However, since they are vulnerable to more types of attacks compared to traditional mechanical meters, i.e., cyber-attacks, energy theft may be an even more serious problem in smart grids. Although some schemes have been proposed for system operators to detect energy theft in smart grids, they require users to send their private information, e.g., load profiles, to the system operators, which violates users' privacy. In particular, users' private information may be sold to interested third-parties. Insurance companies may buy load-profiles from the utility companies to make premium adjustments on the users' policies. For example, they could find electricity consumption patterns that increase the risk of fire in a property and increase insurance premiums accordingly. Marketing companies may also be interested in this data to identify potential costumers. Moreover, criminals may utilize such private information to commit crimes. For instance, the robbers may analyze the energy consumption pattern of the potential victims to deduce their daily behavior. They can even know if a robbery alarm has been set at their target location [43].

The goal of this work is to design practical algorithms capable of realizing the Smart Grid vision of a more efficient, secure, and sustainable power grid. To this end, we propose algorithms for optimal energy management under RTP and DG, and privacy-preserving energy theft detection.

In Chapter 2, we consider a third-party managing the energy consumption of a group of users, and formulate the load scheduling problem as a constrained multi-objective optimization problem (CMOP). The optimization objectives are to minimize energy consumption cost and to maximize a certain utility, which can be conflicting and non-commensurable. We then develop an evolutionary algorithms (EA) to obtain the Pareto-front solutions. To further improve the algorithm efficiency, we present an ϵ-approximate EA that obtains ϵ-Pareto fronts of the objective space. The algorithms are validated by extensive simulation results.

In Chapter 3, we investigate the optimal energy management problem in the smart grid under uncertain energy user demands, distributed renewable energy resources, and energy storage devices. We aim to optimally schedule the usage of all the energy resources in the system and minimize the long-term time averaged expected total cost of supporting all users' load demands. In particular, we first formulate an optimization problem, which turns out to be a time-coupling problem and prohibitively expensive to solve. Based on Lyapunov optimization theory for event-driven queueing systems, we reformulate the problem and develop a dynamic energy management scheme that can dynamically solve the problem in each time slot based only on the current system state. We conduct extensive simulations to evaluate the performance of the proposed dynamic energy management scheme.

In Chapter 4, we address the energy theft problem with distributed, privacy-preserving energy theft detection algorithms. Specifically, utilizing peer-to-peer (P2P) computing with a neighborhood's smart meters as nodes, we solve a linear system of equations (LSE) for users' "honesty coefficients". If a user's honesty coefficient is equal to 1, this user is honest. Otherwise, if the honesty coefficient is larger than 1, then this user has reported less consumed energy and hence is committing fraud. The users' privacy can be preserved because they do not need to disclose any of their energy consumption data to others. Extensive simulations are carried out and the results show that the proposed algorithms can efficiently and successfully identify the fraudulent users in the system.

In Chapter 5, we conclude this dissertation and discuss future work.

CHAPTER 2

MULTI-OBJECTIVE ENERGY CONSUMPTION SCHEDULING IN SMART GRIDS

2.1 Introduction

In power grids, generation capacity is required to meet peak-hour load demand plus a security margin. However, according to recent studies, the average utilization of the generation capacity is below 55% [50]. This leads to inefficient operation of power grids because a portion of generation plants is largely unused or underutilized, but must still be maintained and supervised to guarantee its reliability. On the other hand, as energy demand, and peak load demand as well, continue increasing, additional generation capacity will be needed to accommodate future load demand, which requires a large investment and might lead to even lower utilization.

Recently, the Smart Grid (SG) has been proposed as a new type of electrical grid to modernize current power grids to efficiently deliver reliable, economic, and sustainable electricity services. One of the key features of the SG is the replacement of conventional mechanical meters with smart meters to enable two-way communications between users and grid operators. Using the communication infrastructure of the SG, it is possible to shape the users' load demand curves by means of demand response (DR) strategies. One promising DR strategy is real-time pricing (RTP), where utility companies charge users with a price that varies according to the generation cost, i.e., the higher the generation cost, the higher the price. The advantage of RTP is threefold. First, users may reduce their energy consumption when the price is high, and hence lower their electric bills. Second, peak-hour load demand can be reduced, thus reducing the redundant generation capacity needed to meet reliability requirements. Third, off-peak load demand can be increased, which can increase the utilization of the available generation capacity.

Most current research on real-time pricing focuses on how to optimally schedule all users' energy consumption given their predefined energy demand. In particular, Mohsenian-Rad et al. [33] propose an autonomous load scheduling algorithm based on cooperative game theory, where each user is a player and their load schedules are the strategies. Agarwal and Cui [2] propose a load scheduling noncooperative game among users that can be reduced to a congestion game. In both studies, the single optimization objective is to minimize the electric bill of the users, while the reduction of the peak-hour consumption is considered as a desirable secondary effect. Moreover, Samadi et al. [45] propose an auction based scheme where users provide their utility functions and energy constraints to the utility company, who then replies with a set of prices that maximizes users' utility functions. A similar auction scheme is also proposed by Li et al. [28].

Notice that previous study mostly aims at a single objective, e.g., to minimize users' cost. In this chapter, we formulate the load scheduling problem as a constrained multi-objective optimization problem (CMOP). Specifically, we consider a third-party managing the energy consumption of a group of smart grid users. All users submit their energy requests to the third-party, which then optimally schedules their energy consumption so that its two objectives can be satisfied. The first objective is to minimize the total energy consumption cost, while the second one is to maximize its utility measured by a certain utility function. This third party can be a company, who schedules its departments' energy consumption in order to minimize the cost and maximize its gross income. Or, it can be a community manager, who schedules the residents' energy consumption so that the total energy cost is minimized and its utility (e.g., life comfortness living in this community) is maximized.

We note that these two objectives considered in this study are conflicting and non-commensurable. In the literature, evolutionary algorithms (EAs) have been proven to be effective in finding good approximations of optimal solutions to multi-objective optimization problems [10, 12, 22, 25, 56, 59]. In particular, EAs aim to find a set of solutions that approximate the Pareto-optimal front in the objective space, which all follow two basic steps iteratively: variation and selection. Variation consists of choosing some solutions from the existing (maybe random) solutions to be combined and produce new ones. Then, selection is performed to keep the good solutions and discard the bad ones. Different ways for selecting the best solutions and storing them have been proposed in the literature. In this study, to solve the formulated CMOP, we first develop an evolutionary algorithm, called LSEA, to retrieve a set of Pareto-optimal solutions and show the trade-offs between energy consumption cost and the utility. Then, in order to further improve the algorithm efficiency, we present an ϵ-approximate evolutionary algorithm, called ϵ-LSEA, to obtain ϵ-Pareto fronts of the objective space. Extensive simulations have also been conducted to evaluate the performance of the two proposed algorithms.

The rest of this chapter is organized as follows. Section 3.2 introduces system models considered in this study. We describe the constrained multi-objective optimization problem in Section 2.3. Section 2.4 details the proposed evolutionary algorithms for solving the CMOP. Simulation results are presented in Section 4.6. Finally, we conclude this chapter in Section 4.7.

2.2 System Model

In this section, we briefly describe smart grids, and energy cost model and utility function model in smart grids.

2.2.1 Smart Grids

Smart grids have been promoted by many governments as a way of addressing energy independence and sustainability, global warming, and emergency resilience issues [53]. In smart grids, the energy consumption of each user is monitored by a smart meter (SM), which is also capable of controlling the user's appliances (e.g., turning them on or off, adjusting their settings). Due to their communication capability, SMs also enable two-way communications between users and utility companies, via multihop wireless, wired, or hybrid networks.

In this study, we consider a third-party managing the energy consumption of a group of smart grid users. Each user submits its energy request to the third-party, e.g., 2 kilowatt-hour (kWh) between 10:00 and 18:00, before a day starts (0:00). Then, the third party optimally schedules all users' energy consumption (either locally or via cloud computing) so that its objectives can be satisfied, which are first, to minimize the total energy consumption cost, and second, to maximize its utility measured by a certain utility function. For example, this third party can be a company, who schedules its departments' energy consumption in order to minimize the cost and maximize its gross income. The third party can also be a community manager, who schedules the residents' energy consumption so that the total energy cost is minimized and its utility (e.g., life comfortness living in this community) is maximized.

2.2.2 Energy Cost Model

We discretize a day into H time slots of equal length, which are denoted by a set $\mathcal{H}$. A complete energy consumption schedule for user u ($u \in \mathcal{U}$) during one day is given by a vector $\mathbf{x}_u = [x_u^1, x_u^2, ..., x_u^H]$, where x_u^h is user u's energy consumption in the hth time slot, and $\sum_{h=1}^{H} x_u^h = e_u$, i.e., user u's required energy consumption during one day. Then, the total energy consumption of all users in time slot h ($1 \leq h \leq H$), denoted by E_h, is

$$E_h = \sum_{u=1}^{U} x_u^h \tag{2.1}$$

where $U = |\mathcal{U}|$ is the cardinality of the set $\mathcal{U}$, i.e., the number of users in this area.

Besides, we assume that the energy price functions are known to the third party. One example for such a price function is given below:

$$C_i(E_r) = a_i E_r^2 + b_i E_r, \text{ for } 0 \leq E_r < G_i^{max} \tag{2.2}$$

where E_r is the total energy consumption of all users, a_i and b_i are non-negative coefficients, and G_i^{max} is a upper bound on the energy consumption for this price function to hold.

Furthermore, in practice, the energy price function may be piecewise. In this chapter, we consider a two-piece price function without loss of generality, which is composed of two functions denoted by C_1 and C_2, respectively. Assume that $a_2 > a_1$ and $b_2 > b_1$, i.e., the energy price increases even faster once the energy consumption exceeds a certain threshold. Consequently, the overall cost function of consuming E_r energy, denoted by $C(E_r)$, is

$$C(E_r) = \begin{cases} C_1(E_r), & \text{for } 0 \leq E_r < G_1^{max} \\ C_1(G_1^{max}) + C_2(E_r - G_1^{max}) + M_1, & \\ \qquad \text{for } G_1^{max} \leq E_r < G_1^{max} + G_2^{max} & \\ \infty, & \text{for } E_r \geq G_1^{max} + G_2^{max} \end{cases} \tag{2.3}$$

where $M_1 > 0$ accounts for a marginal cost. Notice that when the total energy consumption exceeds a certain threshold, i.e., $E_r \geq G_1^{max} + G_2^{max}$, the cost goes to infinity. It means that the third party is only allowed to use this much energy (i.e., $G_1^{max} + G_2^{max}$) at most, which could be a constraint to ensure the stability of the neighboring areas considered from the whole grid perspective.

2.2.3 Utility Function Model

In addition to low cost, the third party also intends to achieve high utility, which is calculated by a utility function. As mentioned before, the utility could be a company's gross income, or a community's living comfortness, and so on. Usually, the utility functions are non-decreasing with respect to the consumed power, concave, and results in a zero utility value given zero power consumption [45]. For simplicity, we use the following utility function, denoted by $V(E_r)$, in this study:

$$V(E_r) = \sqrt{E_r} \tag{2.4}$$

where E_r is the total energy consumption of all the users. Note that the utility value may not have the same unit as the energy cost.

2.3 Constrained Multi-Objective Optimization Problem Formulation

In general, a constrained multi-objective optimization problem (CMOP) is defined as follows [8]

$$\begin{aligned}
\underset{\mathbf{x}}{\text{minimize}} \quad & F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})) & \\
\text{subject to} \quad & g_i(\mathbf{x}) \leq 0, & i = 1, \ldots, m \\
& h_j(\mathbf{x}) = 0, & j = 1, \ldots, p \\
& x_q^l \leq x_q \leq x_q^u, & q = 1, \ldots n
\end{aligned} \tag{2.5}$$

where $F(\mathbf{x})$ is the set of objective functions, $g_i(\mathbf{x})$ is the set of inequality constraints, $h_j(\mathbf{x})$ is the set of equality constraints, and x_q^l and x_q^u are the minimum and maximum values of each decision variable x_q, respectively. A CMOP minimizes k objective functions *simultaneously*, where the objective functions represent (usually) competing or conflicting objectives.

In this study, we consider two objective functions, and formulate a CMOP as follows:

$$\underset{\mathbf{x}}{\text{minimize}} \quad \left(\sum_{h=1}^{H} C\Big(\sum_{u=1}^{U} x_u^h\Big), \; -\sum_{h=1}^{H} V\Big(\sum_{u=1}^{U} x_u^h\Big) \right) \tag{2.6}$$

$$\text{subject to} \quad \sum_{u=1}^{U} x_u^h \leq G_1^{max} + G_2^{max}, \; \forall h \in [1, H] \tag{2.7}$$

$$x_u^h \begin{cases} \geq 0, & S_u \leq h \leq T_u \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

$$e_u - \bar{e}_u \leq \sum_{h=1}^{H} x_u^h \leq e_u + \bar{e}_u \tag{2.9}$$

$$1 \leq S_u \leq T_u \leq H \tag{2.10}$$

In the above CMOP, the first objective function minimizes the total energy generation cost during one day, and the second objective function maximizes the utility function. Constraint (2.7) guarantees that in each time slot the total energy consumption does not exceed the maximum generation capacity of the system. Constraint (2.8) indicates that each user u has certain energy demand which needs to be satisfied between a required starting time S_u and a required stopping time T_u. Constraint (2.9) represents a user' tolerance of its daily energy consumption, i.e., the user is fine with consuming $e_u - \bar{e}_u$ to $e_u + \bar{e}_u$ energy in one day. Constraint (2.10) simply means that the starting time is no later than the stopping time for each user, which are both between time slots 1 and H.

2.4 Solving CMOPs by Evolutionary Algorithms

Evolutionary algorithms (EAs) have been proven to be effective in finding good approximations of CMOPs' optimal solutions. The basic idea is to use the crossover, mutation and selection principles of Darwinian evolution to combine, modify and choose possible solutions iteratively until a good approximation of the optimal solution to a CMOP is found. Specifically, crossover and mutation are probabilistic procedures that combine solutions in order to make (possibly better) new solutions. Selection is a deterministic procedure that discards the bad solutions found so far and keeps the good ones. Besides, selection procedures are based on the solutions' fitness, which is usually assigned by an EA based on Pareto dominance and the distance to its nearest neighbors in the objective space. Before we dive into the details, we give some definitions as follows.

2.4.1 Definition 1

In a CMOP, a solution vector $\mathbf{x}$ *is said to Pareto dominate another solution vector* $\mathbf{y}$ *(*$\mathbf{x} \succ \mathbf{y}$*), if* $x_i \leq y_i$ *for all* $i \in [1, k]$ *and there exists some* $i \in [1, k]$ *such that* $x_i < y_i$*, where* k *is the dimension of the solution vectors.*

EAs are usually applied to unconstrained optimization problems. Some different penalty functions and definitions of dominance have been proposed in the literature to handle constraints. Penalty functions are functions of the infeasibility of a solution, where larger values are assigned to solutions farther away from the feasible space of the problem while smaller values are assigned to solutions closer to the feasible space. In this chapter, we adopt the dominance definition given by Deb et al. [12], which takes constraints into consideration and is described below.

2.4.2 Definition 2

A solution vector $\mathbf{x}$ *is said to constraint-dominate another solution vector* $\mathbf{y}$ *(*$\mathbf{x} \succ \mathbf{y}$*) if any of the following conditions is true:*

1. $\mathbf{x}$ *is feasible but* $\mathbf{y}$ *is not.*
2. *Both* $\mathbf{x}$ *and* $\mathbf{y}$ *are feasible and* $\mathbf{x}$ *Pareto dominates* $\mathbf{y}$*, as described by Definition 1 in Section 2.4.1.*
3. *Both* $\mathbf{x}$ *and* $\mathbf{y}$ *are infeasible, but* $\mathbf{x}$ *has lower overall constraint violation.*

After an EA is executed, several non-dominated solutions, in the Pareto sense, are obtained. Each of these solutions is a compromise between the multiple objective functions. In what follows, we first propose an evolutionary algorithm to find Pareto optimal solutions to the load scheduling problem formulated in Section 2.3, and then develop an ϵ-approximate evolutionary algorithm to obtain ϵ-Pareto fronts of the solutions.

2.4.3 Load Scheduling with an Evolutionary Algorithm (LSEA)

An evolutionary algorithm is usually composed of several important processes, including initialization, selection, crossover, and mutation. In the following, we describe such processes, respectively.

In the beginning, N random solutions, called individuals, are created to form the initial population P_0. The initial individuals satisfy constraints (2.8)-(2.10) but may not meet constraint (2.7). Next, all individuals are compared to each other using the constraint-dominance definition (Definition 2 in Section 2.4.2) and each individual is assigned a rank according to the number of individuals by which it is dominated. For example, non-dominated individuals receive a rank of 1, individuals dominated by only one individual receive a rank of 2, and so on. Individuals with the same rank form a front. Besides, a crowding distance [12] is assigned to each individual within the same front. The crowding distance is a measure of how close an individual is to other individuals in the objective space, where a larger crowding distance indicates the individual is farther away from other individuals. Specifically, crowding distance is computed in D steps, where D is the objective space dimensionality. In each dimension d, the individuals are sorted according to their dth objective value. Then, we obtain for each individual the aggregate distance to its two adjacent neighbors with respect to the dth objective. The first and last individuals in each dimension d are assigned a crowding distance of ∞ to preserve diversity. Finally, an individual's crowding distance is calculated as its total aggregate distances in all dimensions. Please refer to Function 1 in Fig. 2.1 for more details.

Once all individuals are assigned a rank and crowding distance, the next step is to select some individuals from P_0, to create a mating pool for crossover and mutation. The selection is done using binary tournament, i.e., randomly selecting two individuals from P_0 and comparing their ranks. The individual with the smaller rank will be selected for the mating pool. If the two individuals have the same rank, then the one with larger crowding distance is selected. If both individuals have the same rank and the same crowding distance, then either one is selected with a probability of 0.5. After the mating pool is filled, the crossover process starts. Each time two random individuals are taken from the mating pool, called parents, to create two more individuals, called offsprings, with probability p_c. Then, the offspring are mutated with probability p_m. Usually, p_c is large and p_m is small. After N offspring individuals have been created, they are grouped in Q_0.

The ith ($i \geq 1$) iteration will start by creating an aggregated population $R_i = P_{i-1} \cup Q_{i-1}$. Then all individuals in population R_i will be assigned a rank and crowding distance. Individuals with rank 1 are added to P_i. Recall P_i has a fixed size of N. If there are less than N individuals with rank 1, all individuals with rank 1 will be added to the new population P_i. To fill in the remaining spots in P_i individuals with rank 2 are considered, and so on. When the last front is considered, and its size is larger than the remaining spots, individuals with larger crowding distances will be included in P_i. All other individuals are discarded. Finally, a new offspring population is created by selecting individuals from P_i for the mating pool, as described previously, and performing crossover and mutation. When the number of iterations reaches a predefined threshold, say G, the algorithm stops and the non-dominated individuals can be extracted from P_G to form a Pareto-front.

Notice that the above description does not specify how to conduct crossover and mutation. Next, we introduce these two processes, respectively. In particular, we adopt the simulated binary crossover (SBX) [3] scheme for the crossover process. This procedure creates two offsprings, y and $\tilde{y}$, from two parents x and $\tilde{x}$ as follows. For any $u \in [1, U]$, $h \in [1, H]$, we get

$$\begin{aligned} y_u^h &= \tfrac{1}{2}[(1-\beta^h)x_u^h + (1+\beta^h)\tilde{x}_u^h] \\ \tilde{y}_u^h &= \tfrac{1}{2}[(1+\beta^h)x_u^h + (1-\beta^h)\tilde{x}_u^h] \end{aligned} \tag{2.11}$$

where y_u^h and $\tilde{y}_u^h$ are the elements of vectors y and $\tilde{y}$, respectively, x_u^h and $\tilde{x}_u^h$ are the elements of vectors x and $\tilde{x}$, respectively, and β^h is a sample generated by a random number generator shown below:

$$\beta(v) = \begin{cases} \tfrac{1}{2}(2v)^{\frac{1}{\eta_c+1}}, \ v \leq \tfrac{1}{2} \\ \tfrac{1}{2}[2(1-v)]^{-\frac{1}{\eta_c+1}}, \ v > \tfrac{1}{2} \end{cases} \tag{2.12}$$

where v is a random variable uniformly distributed in $[0, 1]$, and η_c is a predefined parameter.

Besides, we perform the mutation process shown in the following. For any $u \in [1, U]$, $h \in [1, H]$, we have

$$y_u^h = x_u^h(\frac{1}{2} + \delta) \tag{2.13}$$

where δ is uniformly distributed between 0 and 1.

In the case that the kth decision variable of an offspring after crossover and mutation fall outside the lower and upper bounds specified in the CMOP constraints, they are reset as follows:

$$y_u^h = \begin{cases} x_u^{h,lo}, \text{ if } y_u^h \leq x_u^{h,lo}, \\ x_u^{h,up}, \text{ if } y_u^h \geq x_u^{h,up} \\ y_u^h, \text{ if } x_u^{h,lo} \leq y_u^h \leq x_u^{h,up} \end{cases} \tag{2.14}$$

Algorithm 1 in Fig. 2.2 further details the evolutionary algorithm for load scheduling, which is called LSEA.

2.4.4 Load Scheduling with an ϵ-approximate Evolutionary Algorithm (ϵ-LSEA)

The evolutionary algorithm proposed above provides a dense and diverse set of solutions on the Pareto front (i.e., the Pareto optimal solutions). However, a dense set of solutions may not be necessary because adjacent solutions provide similar trade-offs. In the following, we develop an ϵ-approximate evolutionary algorithm for the load scheduling problem.

We first give some definitions as follows [58].

2.4.4.1 Definition 3

Let $\mathbf{a}$ *and* $\mathbf{b}$ *be two vectors of dimension* k' *in the objective space. Then* $\mathbf{a}$ *is said to* ϵ*-dominate* $\mathbf{b}$ *for some* $\epsilon > 0$*, denoted as* $\mathbf{a} \succ_\epsilon \mathbf{b}$*, if*

$$\epsilon \cdot a_i \geq b_i \qquad \forall i \in \{1, ..., k'\}. \tag{2.15}$$

2.4.4.2 Definition 4

Let $\mathbf{Y}$ *be the objective space and* $\epsilon > 0$*. Then a set* $\mathbf{Y}_\epsilon$ *is called an* ϵ*-approximate Pareto front of* $\mathbf{Y}$*, if any vector* $\mathbf{b} \in \mathbf{Y}$ *is* ϵ*-dominated by at least one vector* $\mathbf{a} \in \mathbf{Y}_\epsilon$*, i.e.,*

$$\forall \mathbf{b} \in \mathbf{Y}, \quad \exists \mathbf{a} \in \mathbf{Y}_\epsilon : \mathbf{a} \succ_\epsilon \mathbf{b}. \tag{2.16}$$

The set of all ϵ*-approximate Pareto fronts of* $\mathbf{Y}$ *is denoted as* $\mathbf{P}_\epsilon(\mathbf{Y})$*.*

2.4.4.3 Definition 5

Let $\mathbf{Y}$ *be the objective space and* $\epsilon > 0$*. Then a set* $\mathbf{Y}^*_\epsilon \subseteq \mathbf{Y}$ *is called an* ϵ*-Pareto front of* $\mathbf{Y}$ *if*

1. $\mathbf{Y}^*_\epsilon$ *is an* ϵ*-approximate Pareto front of* $\mathbf{Y}$*, i.e.,* $\mathbf{Y}^*_\epsilon \in \mathbf{P}_\epsilon(\mathbf{Y})$*, and*
2. $\mathbf{Y}^*_\epsilon$ *contains Pareto points of* $\mathbf{Y}$ *only, i.e.,* $\mathbf{Y}^*_\epsilon \subseteq \mathbf{Y}^*$*.*

The set of all ϵ*-Pareto fronts of* $\mathbf{Y}$ *is denoted as* $\mathbf{P}^*_\epsilon(\mathbf{Y})$*.*

The main idea of ϵ-LSEA is to choose a parent from a variable size population A, called the archive, and another parent from a fixed size population P. After crossover, the resulting offspring may be accepted into the archive depending on whether or not it ϵ-dominates any individual in A. Similarly, the offspring may be accepted into the population depending on its dominance relation to individuals in P. After a predefined number of offsprings have been generated, the solutions in the archive form a diverse $\epsilon-$approximate Pareto front. In what follows, we explain in details the archive acceptance and population acceptance algorithms as well as ϵ-LSEA.

Regarding the archive acceptance algorithm, we adopt the selection strategy proposed by Deb et al. [11] to find ϵ-Pareto fronts with guaranteed convergence and diversity, which is described by Procedure 1 in Fig. 2.3. This algorithm divides the two-dimensional objective space into boxes of size $\epsilon \times \epsilon$ and stores in an archive only one non-dominated solution per box on the ϵ-Pareto fronts. Using a generalized dominance relation on these boxes, the algorithm maintains a set of non-dominated boxes, and hence guaranteeing the ϵ-approximation property. In particular, Procedure 1 in Fig. 2.3 accepts or rejects an offspring as follows. We first identify the solutions in the archive that are dominated by the current offspring. Here, dominance relation is determined using the vector b of each solution obtained with Function in 2 Fig. 2.4. If the offspring dominates any solution, the dominated solution is removed and the offspring is added to the archive. When there are no box-dominated solutions in the archive, we further check two cases. First, if the offspring lies inside a box occupied by an archive solution, then the dominating solution in the Pareto-sense is kept in the archive and the dominated solution is discarded. Second, if the offspring lies inside a box where there is no archive solution, the offspring is added to the archive. Moreover, since in each box there is only one non-dominated solution, the convergence property can be guaranteed, too.

In addition, we have the following theorem [26].

2.4.4.4 Theorem 1

Let $\mathbf{Y}^{(t)} = \bigcup_{j=1}^{t} \mathbf{y}^{(j)}$, $1 \leq \mathbf{y}_i^{(j)} \leq B$, *be the set of all objective vectors created by a multi-objective evolutionary algorithm and given to the selection operator defined in Procedure 1 in Fig. 2.3. Then* $\mathbf{A}^{(t)}$ *is an* ϵ*-Pareto set of* $\mathbf{Y}^{(t)}$ *with bounded size, i.e.,*

$$\mathbf{A}^{(t)} \in \mathbf{P}_{\epsilon}^{*}(\mathbf{Y^{(t)}}) \tag{2.17}$$

$$|\mathbf{A}^{(t)}| \leq \left(\frac{\log B}{\log \epsilon}\right)^{k-1} \tag{2.18}$$

Our population acceptance mechanism, described by Procedure 2 in Fig. 2.6, uses dominance relations and crowding distances to accept an offspring into the population or reject it. In particular, the algorithm works as follows. First, a crowding distance is assigned to each population individual p in P_{g-1} using Function 1 in Fig. 2.1. Next, it is determined if offspring q dominates any p. If it does, the algorithm replaces the dominated p that has the lowest crowding distance CD with q. In case q is dominated by any p, it is rejected. On the other hand, if q does not dominate any p and it is also non-dominated, the p with the lowest CD among all individuals in P_{g-1} is replaced by q. If several individuals have the same lowest CD, then a randomly chosen one is replaced by q. Finally, the procedure returns the updated population P_g. Notice that this procedure only compares the offspring with all members of the population P_{g-1}, rather than compare it with all members of the whole population as in Algorithm 1 in Fig. 2.2. This keeps the computational cost low, and the use of crowding distances maintains a well spread population.

Finally, we describe in details the ϵ-approximate evolutionary algorithm (ϵ-LSEA) for the load scheduling problem detailed by Algorithm 2 in Fig. 2.5. Initially, a random population P_0 is created satisfying constraints (2.8)-(2.10) specified in the CMOP. Then, the non-dominated individuals in P_0 are copied into archive A. In the gth iteration, an individual p is randomly selected from the population P_{g-1} using binary tournament and another solution a is randomly chosen from the archive A to form the mating pool. The parent individuals, p and a, are used for crossover, and the resulting offspring q is subject to mutation. Unlike that in the previous algorithm, only one offspring q is generated per iteration. Next, offspring q is accepted or rejected from the population using Procedure 2 in Fig. 2.6. Lastly, Procedure 1 in Fig. 2.3 is used to decide whether or not offspring q is added into the archive A. The algorithm stops after a predefined number of offsprings G have been generated. Since fewer solutions are needed to converge to the Pareto-front, this algorithm has a shorter computation time than Algorithm 1 in Fig. 2.2.

2.5 Simulation Results

In this section, we conduct simulations to evaluate the performance of the proposed two algorithms, i.e., Load Scheduling with an EA (LSEA, Algorithm 1 in Fig. 2.2) and Load Scheduling with an ϵ-approximate EA (ϵ-LSEA, Algorithm 2 in Fig. 2.5), respectively. The proposed algorithms are implemented in Matalb2011b on a general purpose computer with a 3.4GHz CPU and 4GB RAM memory. The parameters for the cost function in equation (2.3) are presented in Table 2.1, and some parameters indicated in constraints (2.7), (2.8) and (2.9) are given in Table 2.2 which are the same for all users. Besides, when two parents are selected for reproduction, the crossover process (SBX) will be applied with probability

$p_m = 0.9$ and $\eta_c = 0$, and each offspring will mutate with probability $p_m = e^{-g/G}$, where g is the number of the current iteration and G is the predefined iteration number.

2.5.1 LSEA

We first evaluate the performance of LSEA with 5, 15, 25 and 50 users, respectively. In particular, each user has a daily energy requirement e_u, which is uniformly distributed between 0 and 24 kWh, to be scheduled throughout 24 hours. Fig. 2.7(a) shows the obtained Pareto-front for 5 users. Each cross in the graph represents a solution found by LSEA and its position is determined by the values of the corresponding objective functions. We can observe that the range of the cost objective goes from \$2 to \$48 and the utility function spans from 10 to 70. These solutions in objective space provide us with a wide set of trade-offs between the total energy consumption cost and the overall utility. Moreover, we notice that the Pareto-front is densely populated, i.e., adjacent solutions are very close to each other. Fig. 2.7(b)-2.7(d) show similar results for the cases of 15, 25 and 50 users, respectively.

2.5.2 ϵ-LSEA

Next, we show the performance of ϵ-LSEA with 5, 15, 25 and 50 users, respectively. The same as before, we assume that each user has a daily energy requirement e_u, which is uniformly distributed between 0 and 24 kWh, to be scheduled throughout 24 hours. As shown in Fig. 2.8(a), we can easily see there is an ϵ-Pareto front with only a few solutions, which can make the final decision easier. Fig. 2.8(b)-2.8(d) also show an ϵ-Pareto front that can be easily identified. Moreover, in these three cases the results are obtained using a large number of iterations. However, as we will show in the next section, in fact a lot fewer generations are enough to obtain an ϵ-Pareto front. Here, we show the results with a large number of iterations after an ϵ-Pareto front has been identified to be sure that the algorithm has converged.

Moreover, the time and the number of iterations needed for obtaining the results shown in Fig. 2.7 and Fig. 2.8 are presented in Table 2.3. We can see that the efficiency of ϵ-LSEA is higher than that of LSEA, and the efficiency improvement gets more significant when the number of users becomes larger.

2.5.3 Convergence of LSEA and ϵ-LSEA

Finally, we compare the convergence speed of LSEA and ϵ-LSEA by looking into the evolution of the population of LSEA and of the archive of ϵ-LSEA, when the number of users is 25. Fig. 2.9(a)-2.9(d) show the progress of the population of LSEA when the running time is equal to 15, 90, 240, and 600 minutes, respectively. We can find that a good Pareto front can be found only after 600 minutes. Compared to that, we can see in Fig. 2.10(a)-2.10(d) that a good ϵ-Pareto front can be achieved after 120 minutes, which is much faster. Besides, considering the modest capability of the computer used to run these simulations, the third party usually would have more computing resources and thus even shorter computation time. It can also employ cloud computing to accomplish the load scheduling tasks, which would further reduce the computation time.

2.6 Conclusions

In this chapter, we consider a third-party managing the energy consumption of a group of smart grid users, and formulate the load scheduling problem as a constrained multi-objective optimization problem. The first objective is to minimize the total energy consumption cost, while the second is to maximize its utility measured by a certain utility function. To solve the problem, we first develop an evolutionary algorithm, called LSEA, to retrieve a set of Pareto-optimal solutions and show the trade-offs between energy consumption cost and the utility. Then, in order to further improve the algorithm efficiency, we present an ϵ-approximate evolutionary algorithm, called ϵ-LSEA, to obtain ϵ-Pareto fronts of the objective space. Extensive simulations have also been conducted to evaluate the performance of the two proposed algorithms. We can observe that ϵ-LSEA is more efficient compared to LSEA.

```
Input: Individuals $p_k$'s in front $Z$, objective space dimension $D$
1: Calculate for each individual $p_k$ the objective values $f_{k,1}, \ldots, f_{k,D}$ in the objective
   space
2: Set $I_k$ to 0 for each individual $p_k$
3: for $d = 1$ to $D$ do
4:     Sort individuals $p_k$'s in $Z$ in ascending order according to $f_{k,d}$
5:     The crowding distance of the first and of the last individual are set to infinity
6:     for $k = 2$ to the size of $Z$ minus 1 do
7:         $I_k = I_k + (f_{k-1,d} - f_{k+1,d})/(\max_k\{f_{k,d}\} - \min_k\{f_{k,d}\})$
8:     end for
9: end for
Output: Crowding distances $I_k$'s
```

Figure 2.1

Function 1: Crowding Distance Assignment

Table 2.1

Cost Function Parameters (U: the number of users, $H = 24$)

a_1	b_1	a_2	b_2	G_1^{max}	G_2^{max}	M_1
0.2	0.3	0.4	0.6	$8U/H$	$16U/H$	1

```
Input: $N$
 1: Create an random initial population, $P_0$ of size $N$, satisfying constraints (2.8)-
    (2.10) in the CMOP
 2: Apply non-dominating sorting to $P_0$
 3: Apply binary tournament to $P_0$ to fill mating pool
 4: Crossover individuals in mating pool to fill offspring set $Q_0$
 5: Apply mutation to $Q_0$
 6: Set the maximum number of generations, $G$
 7: for $g = 1$ to $G$ do
 8:     $R_g = P_{g-1} \cup Q_{g-1}$
 9:     Apply non-dominating sorting to $R_g$
10:     Apply binary tournament to $R_g$ to fill mating pool
11:     Apply crossover to individuals in mating pool to generate $Q_g$
12:     Apply mutation to individuals in $Q_g$
13:     Create $P_g$
14: end for
Output: Non-dominated individuals in $P_G$
```

Figure 2.2

Algorithm 1: Load Scheduling with an EA (LSEA)

```
Input: $A, f$
1: $D := \{f' \in A | box(f) \succ box(f')\}$
2: if $D \neq \emptyset$ then
3:     $A' = A \cup f \setminus D$
4: else if $\exists f' : (box(f') = box(f) \wedge f \succ f')$ then
5:     $A' = A \cup f \setminus f'$
6: else if $\nexists f' : box(f') = box(f) \vee box(f') \succ box(f)$ then
7:     $A' = A \cup f$
8: else
9:     $A' = A$
10: end if
Output: $A'$
```

Figure 2.3

Procedure 1: Selection Process for ϵ-Pareto Front

Table 2.2

Parameters in Constraints (2.7)-(2.9)

S_u	T_u	$\bar{e}_u$
0	24	0.5kW

```
Input: $f$
1: for all $i \in \{1, \ldots, m\}$ do
2:     $b_i = \lfloor \frac{\log f_i}{\log 1+\epsilon} \rfloor$
3: end for
4: $b = (b_i, \ldots, b_m)$
Output: $b$
```

Figure 2.4

Function 2: box

Table 2.3

Completion Time

	LSEA		ϵ-LSEA	
Users	**Time(mins)**	**Generations**	**Time(mins)**	**Generations**
5	25	92×10^3	15	26×10^3
15	548	1×10^6	495	5×10^6
25	1312	1.5×10^6	632	13×10^6
50	10033	1×10^6	4978	14×10^6

```
1: Create a random initial population, $P_0$ of size $N$, satisfying constraints (2.8)-(2.10)
   in the CMOP
2: Copy non-dominated individuals in $P_0$ to $A$
3: for $g = 1$ to $G$ do
4:     Choose a solution $p$ from $P_{g-1}$ using binary tournament, and a solution $a$
       from $A$ at random
5:     Use $p$ and $a$ as parents to create one offspring $q$.
6:     Apply mutation to $q$ resulting in $q'$
7:     Run Procedure 2 to decide if $q'$ is included in population $P_g$
8:     Run Procedure 1 to decide if $q'$ is included in the achieve $A$
9: end for
Output: $\epsilon$-Pareto fronts in $A$
```

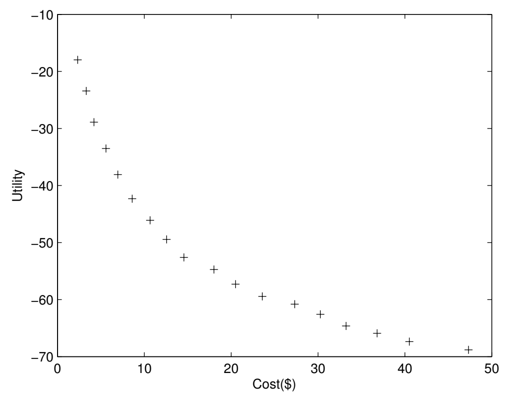
Figure 2.5

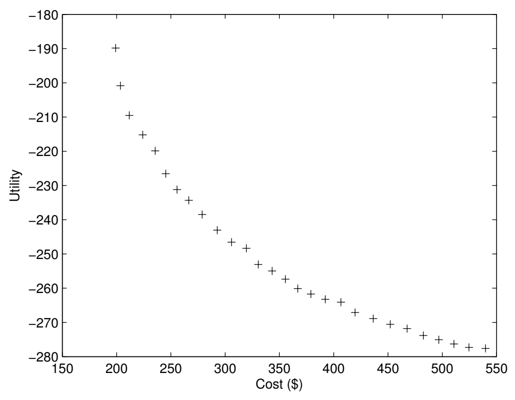
Algorithm 2: Load Scheduling with an ϵ-Approximate EA (ϵ-LSEA)

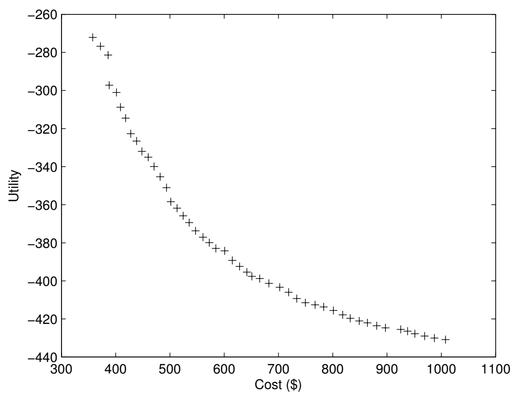
```
**Input:** population $P_{g-1}$, offspring $q$
1: Apply Function 1 to $P_{g-1}$ to assign crowding distances $CD$ to each population individual $p$
2: if $\exists\, p : q \succ p$ then
3:     Replace the individual $p$ that is dominated by the offspring $q$ and has the smallest $CD$ with $q$ (or break ties randomly).
4: else if $\exists\, p : p \succ q$ then
5:     Discard $q$
6: else
7:     Replace the $p$ with the smallest $CD$ with offspring $q$ (or break ties randomly).
8: end if
**Output:** $P_g$
```

Figure 2.6

Procedure 2: Population Acceptance Procedure for ϵ-LSEA


(a) 5 users


(b) 15 users


(c) 25 users

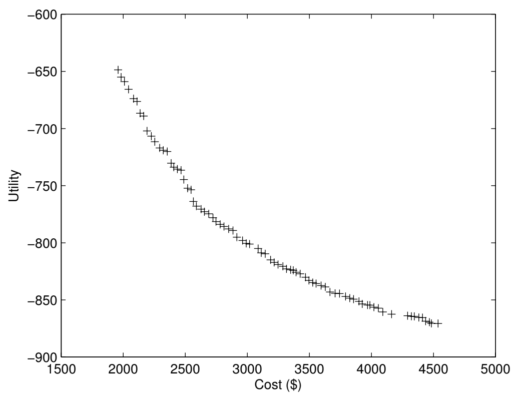
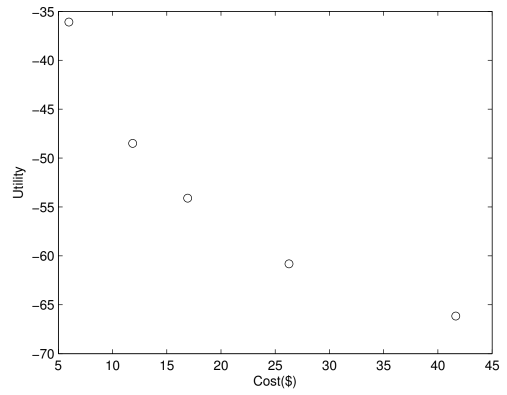
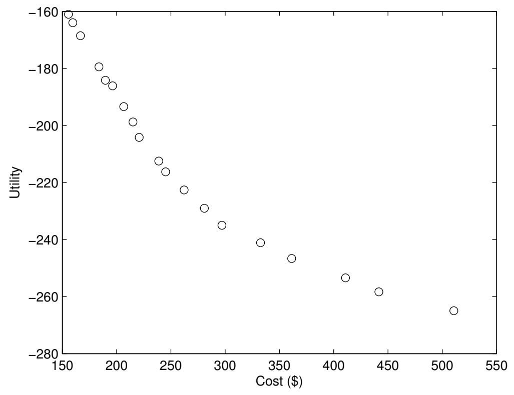

(d) 50 users

Figure 2.7

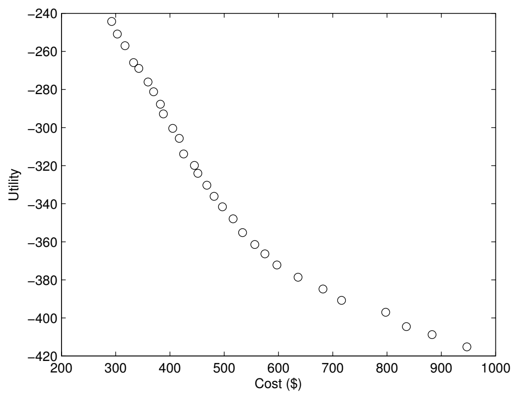
Pareto front for 5, 15, 25, and 50 users respectively, using LSEA.



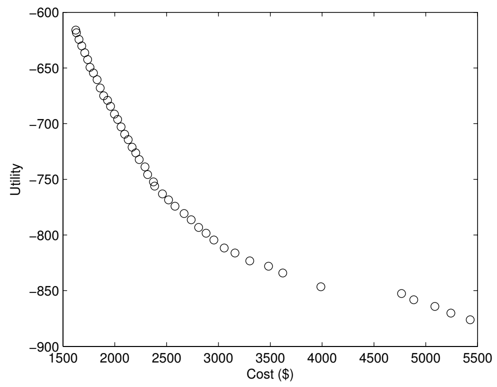
(a) 5 users



(b) 15 users



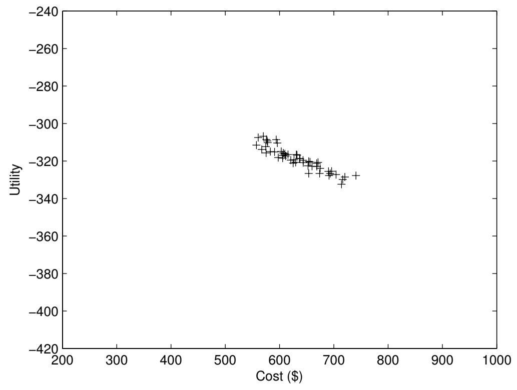
(c) 25 users



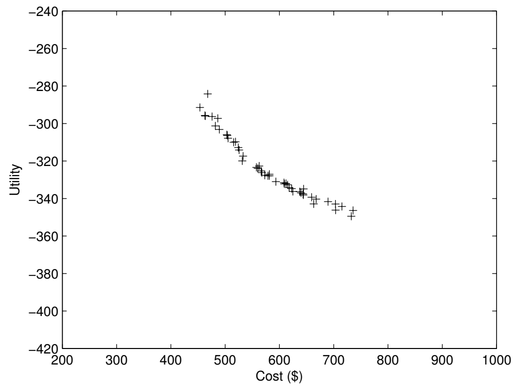
(d) 50 users

Figure 2.8

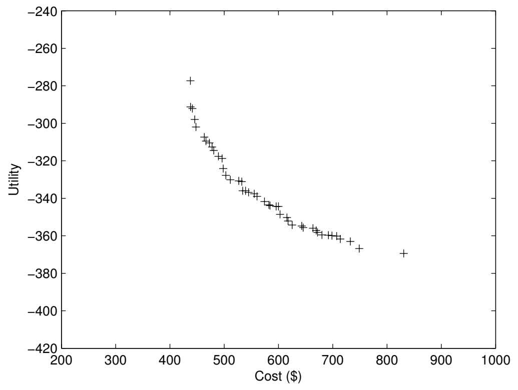
ϵ-Pareto front for 5, 15, 25, 50 users, respectively, using ϵ-LSEA.



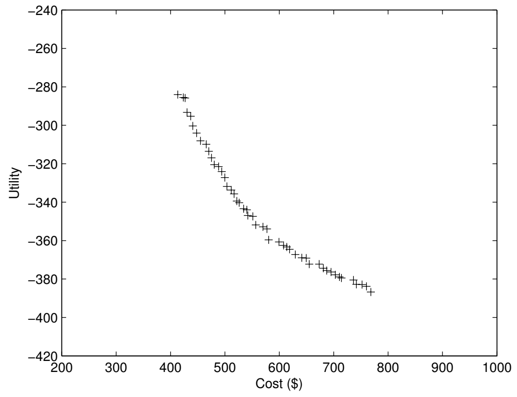
(a) 15 min (5×10^3)



(b) 90 min (23×10^3Iterations)



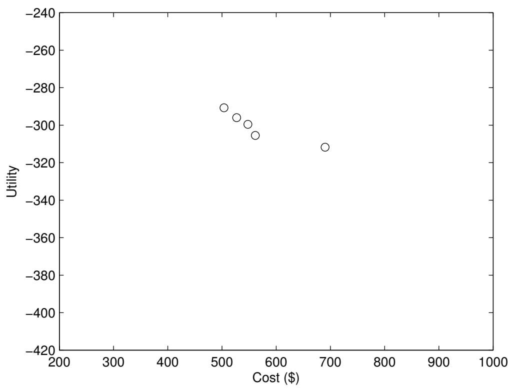
(c) 240 min (80×10^3 Iterations)



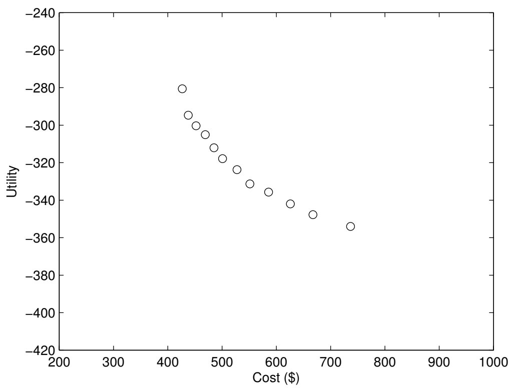
(d) 600 min (185×10^3 Iterations)

Figure 2.9

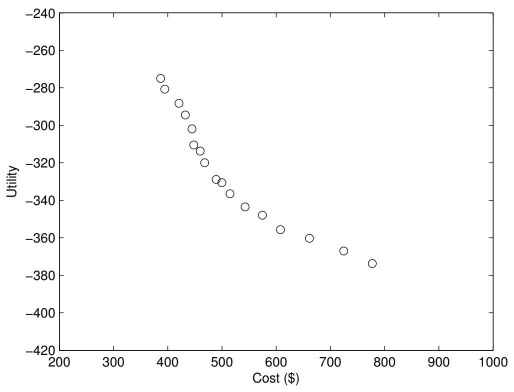
Population evolution using LSEA at different generations for 25 users.



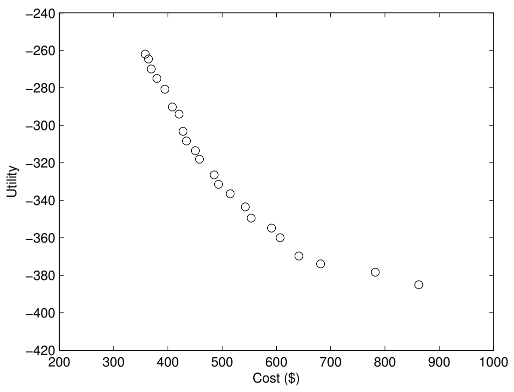
(a) 1 min (11×10^3 Iterations)



(b) 15 min (152×10^3 Iterations)



(c) 60 min (578×10^3 Iterations)



(d) 120 min (1×10^6 Iterations)

Figure 2.10

Population evolution using ϵ-LSEA at different generations for 25 users.

CHAPTER 3

DYNAMIC ENERGY MANAGEMENT FOR THE SMART GRID WITH DISTRIBUTED ENERGY RESOURCES

3.1 Introduction

Largely underutilized generation capacity and high transmission losses are two major sources of system inefficiency in traditional power grids. Recent studies show that the average utilization of the generation capacity is below 55% [49] and 7% of generated energy is lost due to transmission inefficiencies [55]. In particular, since enough generation capacity is required to be available to meet peak-hour load demand plus a security margin, some power plants are largely unused or underutilized. Besides, energy users are usually several hundreds of miles away from power plants, which inevitably results in a significant amount of energy loss due to transmission inefficiencies. Moreover, overall electricity consumption is projected to increase by about 14% in the next 20 years [54], which will require a big investment to expand the generation and transmission capacity to accommodate the new demand.

Recently, the Smart Grid (SG) has been proposed as a new electrical grid to modernize current power grids and enhance its efficiency, reliability, and sustainability. Particularly, in the SG, a digital communication network is deployed to enable two-way communications between users and system operators. It thus makes it possible to shape the users' load demand curves by means of *demand response (DR)* strategies, i.e., to encourage customers to change their usual electricity consumption patterns by incentives [4]. One such strategy is real time pricing (RTP), in which system operators charge users a price that varies according to real-time energy generation cost. Since usually generation cost increases as the amount of generated energy increases, users may want to shift their load demands from peak hours to other times. Therefore, RTP can reduce the peak-hour load demand in the power system, which in turn lowers the requirement on system generation capacity. It can also reduce users' electricity bills by encouraging them to consume more power during hours with lower electricity prices. Another feature of the Smart Grid is *distributed generation (DG)*, where users install and take advantage of renewable generation resources (such as solar panels and wind turbines), and energy storage devices (e.g., batteries). In DG, users determine whether to immediately consume their own (generated or stored) energy, store it, or sell it to the grid. Thus, DG can help reduce the energy loss due to transmission inefficiencies, alleviate congestion during peak hours, reduce the system's carbon footprints, and lower users' electricity bills.

Due to unpredictable realtime prices and distributed energy resources, the Smart Grid poses great challenges for energy management (or load scheduling) with RTP and DG. Most previous studies focus on obtaining load schedules for customers in day-ahead scenarios based on the their load requirement. In particular, Goudarzi et al. [19] propose a mixed-integer optimization problem to find a load schedule that minimizes a customer's energy consumption cost plus an inconvenience function. Du et al. [15] present a two-step optimization algorithm to minimize a user's energy cost to run thermostatically controlled appliances. Gatsis and Giannakis [17] develop a day-ahead scheduling scheme considering imperfect information between the utility company and the customers due to packet loss. Mohsenian-Rad et al. [33] employ game theory to find an optimal daily load schedule for each user that minimizes the total energy generation cost. Shinwari et al. [47] design a water-filling based algorithm, which results in almost flat total power consumption of a neighborhood so as to minimize the changes in load demand per hour and reduce the utility company's operational costs. Salinas et al. [44] investigate a constrained multi-objective optimization problem (CMOP) to manage the energy consumption of a group of users. They develop two evolutionary algorithms to obtain the Pareto-front solutions and the ϵ-Pareto front solutions to the CMOP, respectively. Joe-Wang et al. [23] formulate a linear optimization problem to maximize the utility company's revenue. Note that all these studies require users to know exactly their load demands ahead of time, which may not be always predicted and can be uncertain. Besides, none of the above studies considers DG, energy storage management, or the possibility of users selling energy to the grid, which are essential and appealing features of the SG. In contrast, Neely et al. [37] develop an algorithm to minimize the long- term average expected cost of a utility company, which supplies power by a traditional power plant and a renewable energy resource. Individual user's load demand and energy storage devices are not considered. In [52], Urgaonkar et al. study a similar problem for a data center with an uninterruptible power supply that acts as an energy storage device. Guo et al. [20] propose an algorithm to minimize one user's long term expected energy cost considering a renewable energy resource and a battery. Note that essentially these works deal with one single load demand.

In this chapter, we investigate the optimal energy management problem in the smart grid, taking into account customers' uncertain load demands, and distributed renewable energy resources and energy storage devices. Specifically, we consider an electric power distribution network consisting of a set of energy users, who have two-way real-time communications with a utility company. Each user has a renewable energy resource, an energy storage device, and a connection to the power grid, which collaboratively satisfy its load demand. The utility company provides energy to the users from both a traditional power plant (e.g., coal, gas) and a renewable energy resource (e.g., solar bank, wind farm). We model users' load demands and all renewable energy resources' as stochastic processes to account for their uncertainty. Besides, we consider that the system works in a time-slotted fashion. We aim to optimally schedule the usage of all the energy resources in the network and minimize the utility company's long-term time averaged expected total cost of supporting all users' load demands.

Moreover, we study two cases of users' load demands: first, users have delay intolerant (DI) load demands which need to be satisfied in the same time slot when they are requested, and second, users have both DI and delay tolerant (DT) load demands, the latter of which can tolerate being served within user-defined deadlines. In each case, we first formulate an optimization problem, which turns out to be a time-coupling problem. Previous approaches usually solve such problems based on Dynamic Programming [24, 41] and suffer from the "curse of dimensionality" problem [6]. They also require full statistical information of the random variables in the problem, which may be difficult to obtain in practice. Instead, we reformulate the problem using Lyapunov optimization theory for event-driven queueing systems [36]. We develop a dynamic energy management scheme that can dynamically solve the problem in each time slot based on the current system state only, i.e., without any information about the future or past system states, and hence is more efficient than previous approaches. With the results of our dynamic energy management scheme, we are then able to obtain both a lower and an upper bound on the optimal result of the original optimization problem. Furthermore, in the case of both DI and DT load demands, we also show that DT load demands are guaranteed to be served within user-defined deadlines. Extensive simulations have been conducted to evaluate the performance of the proposed dynamic energy management scheme. Results show that the proposed scheme can lead to tight lower and upper bounds on the optimal result, and can significantly reduce the utility company's cost.

The rest of the chapter is organized as follows. Section 3.2 introduces system models considered in this study. We study dynamic energy management with DI load demands in Section 3.3 and with both DI and DT loads in Section 3.4. Simulations are conducted in Section 3.5. We finally conclude this chapter in Section 3.6.

3.2 System Model

In this section we describe the considered smart grid network and our mathematical models for users' delay intolerant load demand, distributed renewable energy generation, distributed energy storage, load serving, and the utility company's energy generation cost. Note that we only introduce delay intolerant load demand model here. Delay tolerant load demand model will be discussed in Section 3.4.

3.2.1 Smart Grid Network

We consider an electric power distribution network consisting of a set of residential and business energy users, denoted by $\mathcal{I} = \{1, 2, \ldots, n\}$, who have two-way real-time communications with a utility company. Each user has a renewable energy resource, an energy storage device, and a connection to the power grid, which collaboratively satisfy its load demand. The utility company provides energy to the users from both a traditional power plant (e.g., coal, gas) and a renewable energy resource (e.g., solar bank, wind farm). It aims to optimally schedule the usage of all the energy resources in the network and minimize its total cost of supporting all users' load demands. Besides, we consider that the system works in a time-slotted fashion. Energy management decisions are made dynamically by the utility company in each time slot. In particular, in each time slot, users transmit their load requests along with other state variables to a control center deployed by the utility company. Based on the collected data, the control center computes a load servicing schedule and transmits to each user his/her corresponding actions needed to be executed in the current time slot. Each user then follows the instructions and updates some of his/her state variables.

3.2.2 Delay Intolerant Load Demand Model

DI load demands are very common in our daily life, such as lighting and using electronic devices, and need to be satisfied in the same time slot. Denote user i's delay intolerant (DI) load demand in time slot t by $l_i(t)$. We assume $\{l_i(t)\}_{t=0}^{\infty}$ is an independent and identically distributed (i.i.d.) non-negative stochastic process, which is deterministically bounded, i.e., $0 \leq l_i(t) \leq l_i^{max}$.

3.2.3 Distributed Renewable Energy Generation

Each user is equipped with a renewable energy resource, which can be a set of solar panels or a wind turbine. The output of a renewable energy resource is dynamic and difficult to predict because it depends on meteorological conditions. In this work, we assume that the output of user i's renewable energy resource, denoted by $e_i(t)$, is an i.i.d. stochastic process and satisfies $0 \leq e_i(t) \leq e_i^{max}$, where e_i^{max} is the maximum energy output of user i's renewable energy resource and a constant.

In addition to serving user i's load, $e_i(t)$ can be used to charge the user's energy storage device, or sold to the power grid. In particular, we have

$$e_i(t) = r_i^l(t) + r_i^g(t) + c_i^r(t) \tag{3.1}$$

where $r_i^l(t)$ is the energy used to satisfy user i's load demand $l_i(t)$, $r_i^g(t)$ is the energy sold to the grid, and $c_i^r(t)$ is the energy used to charge user i's energy storage device.

3.2.4 Distributed Energy Storage

Each user i has an energy storage device which can store some energy that can be used at a later time. Since the energy storage device acts as an energy buffer, we can model its energy level as a queue, i.e.,

$$B_i(t+1) = B_i(t) + C_i(t) - D_i(t). \tag{3.2}$$

In particular, $C_i(t)$ is the energy charging the energy storage device, i.e.,

$$C_i(t) = c_i^g(t) + c_i^r(t) \tag{3.3}$$

where $c_i^g(t)$ and $c_i^r(t)$ are the energy drawn from the grid and from the renewable energy resource, respectively. $D_i(t)$ is the energy discharged from the energy storage device, i.e.,

$$D_i(t) = d_i^g(t) + d_i^l(t) \tag{3.4}$$

where $d_i^g(t)$ is the energy sold to the grid, and $d_i^l(t)$ is the energy serving user i's DI load demand.

Notice that it is more efficient to serve user i's load demand $l_i(t)$ by directly using energy from the grid or from the renewable energy resource, than by first charging the energy storage device and then discharging it. Thus, we have the following two constraints

$$\mathbf{1}_{d_i^l(t)>0} + \mathbf{1}_{c_i^r(t)>0} \leq 1 \tag{3.5}$$

$$\mathbf{1}_{d_i^l(t)>0} + \mathbf{1}_{c_i^g(t)>0} \leq 1 \tag{3.6}$$

where the indicator function $\mathbf{1}_A$ is equal to 1 when the event A is true, and zero otherwise.

On the other hand, it is more efficient to sell energy to the grid by directly selling the output of the renewable energy resource, than by first charging the energy storage device and then discharging it. Thus, we have

$$\mathbf{1}_{d_i^g(t)>0} + \mathbf{1}_{c_i^r(t)>0} \leq 1 \tag{3.7}$$

Similarly, discharging the energy storage device to sell energy to the grid and charging it by drawing energy from the grid cannot take place at the same time, i.e.,

$$\mathbf{1}_{d_i^g(t)>0} + \mathbf{1}_{c_i^g(t)>0} \leq 1 \tag{3.8}$$

The above constraints (3.5)-(3.8) will always hold when the following one holds:

$$\mathbf{1}_{C_i(t)>0} + \mathbf{1}_{D_i(t)>0} \leq 1 \tag{3.9}$$

Besides, denote by B_i^{max} the maximum amount of energy that can be stored by user i's energy storage device. Then, we need

$$0 \leq B_i(t) \leq B_i^{max}. \tag{3.10}$$

Denote by C_i^{max} the maximum amount of energy that user i's energy storage device can be charged with during a single time slot, and D_i^{max} the maximum amount of energy that can be discharged from user i's energy storage device during a single time slot. Thus, we have

$$\begin{aligned} C_i(t) &\leq \min[C_i^{max}, B_i^{max} - B_i(t)] && (3.11) \\ D_i(t) &\leq \min[D_i^{max}, B_i(t)]. && (3.12) \end{aligned}$$

From (3.11) and (3.12), we get $C_i(t) + D_i(t) \leq B_i^{max} - B_i(t) + B_i(t) = B_i^{max}$, which should hold for any $C_i(t)$ and $D_i(t)$ that satisfy (3.11) and (3.12). Since $C_i(t) \leq C_i^{max}$ and $D_i(t) \leq D_i^{max}$, we also have the following constraint:

$$C_i^{max} + D_i^{max} \leq B_i^{max}. \tag{3.13}$$

3.2.5 Load Serving

The utility company needs to supply enough energy to the grid to satisfy all users' load demands. The amount of energy supplied by the utility company in time slot t, denoted by $P(t)$, can be calculated as

$$P(t) = \sum_{i\in\mathcal{I}} \big(l_i(t) + c_i^g(t) - r_i^l(t) - r_i^g(t) - d_i^g(t) - d_i^l(t)\big). \tag{3.14}$$

User i's load demand is satisfied by the energy from the power grid, its local renewable energy resource, and its own energy storage device. Particularly, we have

$$l_i(t) = g_i^l(t) + r_i^l(t) + d_i^l(t) \tag{3.15}$$

where $g_i^l(t)$ is the amount of energy drawn from the power grid to satisfy user i's load demand in time slot t. Note that user i's connection to the power grid can only be in one of three states: drawing energy from the grid, providing energy to the grid, and idle, i.e., cannot draw and provide energy at the same time. Therefore, we get

$$\mathbf{1}_{g_i^l(t)+c_i^g(t)>0} + \mathbf{1}_{d_i^g(t)+r_i^g(t)>0} \leq 1. \tag{3.16}$$

In addition, the total amount of energy that user i draws from the power grid in time slot t, denoted by $G_i(t)$, satisfies

$$0 \leq G_i(t) = g_i^l(t) + c_i^g(t)c \leq G_i^{max} \tag{3.17}$$

where G_i^{max} is a constant determined by the physical characteristics of user i's connection to the grid. Similarly, the total amount of energy that user i provides to the power grid in time slot t, denoted by $M_i(t)$, satisfies

$$0 \leq M_i(t) = r_i^g(t) + d_i^g(t) \leq M_i^{max} \tag{3.18}$$

where M_i^{max} is also a constant.

3.2.6 Energy Generation Cost

As mentioned before, the utility company provides energy to the users from both a traditional power plant and a renewable energy resource. Assume the output of the utility company's renewable energy resource, denoted by $R(t)$, is an i.i.d. non-negative stochastic process. The cost of generating such renewable energy is considered to be negligible. Thus, the utility company will first use renewable energy and then traditional energy to satisfy users' load demands. The amount of traditional energy the utility company needs in time slot t, denoted by $N(t)$, is

$$N(t) = P(t) - R(t) \tag{3.19}$$

If $R(t) > P(t)$, then the utility company is able to sell the excess power to other utility companies.

Consequently, a utility company's energy generation cost can be calculated as

$$U(t) = f\left(N(t)\right) \tag{3.20}$$

where $f\left(N(t)\right)$ is a non-decreasing and convex function[1].

3.3 Dynamic Energy Management with Delay Intolerant Load Demands

In this section, we study the dynamic energy management for the smart grid when users have delay intolerant (DI) load demands.

3.3.1 Problem Formulation

Let $\mathbf{H}(t) = \{H_1(t), H_2(t), \ldots, H_n(t)\}$ be the vector of decision variables in the system, where $H_i(t) = \{g_i^l(t), d_i^g(t), d_i^l(t), c_i^r(t), c_i^g(t), r_i^g(t), r_i^l(t)\}$. We also denote the system state by a vector of random variables, i.e., $\mathbf{S}(t) = \{S_1(t), S_2(t), \ldots, S_n(t), R(t)\}$

[1]Note that our analysis herein still holds if we assume a concave cost function f. In that case, our objective function can be set to $\max\{-f(N(t))\}$

where $S_i(t) = \{l_i(t), e_i(t)\}$. Thus, the utility company's objective is to design a dynamic energy management algorithm, which can optimally control the decision vectors $\mathbf{H}(t)$ $(t \geq 0)$ to minimize the following long-term time averaged expected total cost, i.e.,

$$\overline{U} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{U(t)\}, \tag{3.21}$$

under uncertain system states $\mathbf{S}(t)$ $(t \geq 0)$[2]. We call this problem $\mathbf{P1}$ and formally formulate it as follows:

$$\mathbf{P1}: \quad \textbf{Minimize} \quad \overline{U} \tag{3.22}$$

We denote the optimal result, i.e., the minimum of the objective function, of $\mathbf{P1}$ by $\mathbf{P1}^*$. We can see that $\mathbf{P1}$ is a time-coupling optimization problem due to constraints (3.2), (3.10)-(3.12). Previous approaches usually solve such problems based on Dynamic Programming and suffer from the "curse of dimensionality" problem [6]. They also require detailed statistical information of the random variables in the problem, which may be difficult to obtain in practice. Next, we reformulate this problem using Lyapunov optimization theory for queueing systems [36] so that it can be solved in each time slot based on the current system state only.

3.3.2 Dynamic Energy Management Using Lyapunov Optimization

In order to better control users' energy storage devices, we define a shifted energy level $X_i(t)$ for user i's energy storage device in time slot t as follows:

$$X_i(t) = B_i(t) - V\beta^{max} - D_i^{max} \tag{3.23}$$

[2]Note that we use $\bar{x}$ to denote the long-term time averaged expected value of a stochastic process $x(t)$ in this study.

where β^{max} is the maximum first-order derivative of $U(t)$ with respect to $N(t)$, and V is a positive constant to be defined later. We also denote by β^{min} the minimum first-order derivative of $U(t)$ with respect to $N(t)$.

Thus, according to (3.2), $X_i(t)$ is updated by the following queueing rule:

$$X_i(t+1) = X_i(t) + C_i(t) - D_i(t). \tag{3.24}$$

Consequently, we can define a Lyapunov function [36] as

$$L(\mathbf{X}(t)) = \frac{1}{2}\sum_{i\in\mathcal{I}}(X_i(t))^2. \tag{3.25}$$

where $\mathbf{X}(t) = \{X_1(t), \ldots, X_n(t)\}$. This function represents a scalar measure of stored energy in the system. $L(\mathbf{X}(t))$ being small implies that all stored energy levels are low, while $L(\mathbf{X}(t))$ being large implies that at least one stored energy level is high. Besides, the one-slot conditional Lyapunov drift can be defined as

$$\Delta(\mathbf{X}(t)) = \mathbb{E}\{L(\mathbf{X}(t+1)) - L(\mathbf{X}(t))|\mathbf{X}(t)\} \tag{3.26}$$

Since our objective is to minimize the long-term time averaged expected total cost of the utility company, instead of taking a control action to minimize $\Delta(\mathbf{X}(t))$, we minimize the following drift-plus-penalty function:

$$\Delta(\mathbf{X}(t)) + V\mathbb{E}\{U(t)|\mathbf{X}(t)\}. \tag{3.27}$$

We can have the following lemma.

3.3.2.1 Lemma 1

Given $\Delta(\mathbf{X}(t))$ *defined in (3.26), we have*

$$\begin{aligned}&\Delta(t) + V\mathbb{E}\{U(t)|\mathbf{X}(t)\}\\ \le\, &A + V\mathbb{E}\{U(t)|\mathbf{X}(t)\} + \sum_{i\in\mathcal{I}} \Bigg(X_i(t)\mathbb{E}\{C_i(t) - D_i(t)|\mathbf{X}(t)\}\end{aligned} \tag{3.28}$$

where A *is a constant, i.e.,*

$$A = \sum_{i\in\mathcal{I}} \Bigg(\frac{\max[(C_i^{max})^2, (D_i^{max})^2]}{2} \Bigg). \Bigg(\tag{3.29}$$

Proof: Squaring both sides of (3.24), we get

$$\begin{aligned}&\frac{X_i^2(t+1) - X_i^2(t)}{2}\\ =\ &\frac{(C_i(t) - D_i(t))^2}{2} + X_i(t)(C_i(t) - D_i(t))\\ \le\ &\frac{\max[(C_i^{max})^2, (D_i^{max})^2]}{2} + X_i(t)(C_i(t) - D_i(t)).\end{aligned} \tag{3.30}$$

Thus, we can obtain that

$$\begin{aligned}&\Delta(t) + V\mathbb{E}\{U(t)|\mathbf{X}(t)\}\\ \le\ &V\mathbb{E}\{U(t)|\mathbf{X}(t)\} + \mathbb{E}\Bigg\{ \sum_{i\in\mathcal{I}} \Bigg(\Bigg(\frac{\max[(C_i^{max})^2, (D_i^{max})^2]}{2}\\ &+ X_i(t)(C_i(t) - D_i(t)) \Bigg) |\mathbf{X}(t) \Bigg\}, \Bigg(\end{aligned} \tag{3.31}$$

and (3.28) directly follows. ■

Our objective is to minimize the right-hand side of (3.28) in each time slot t given the current stored energy levels $\mathbf{X}(t)$ and system state $\mathbf{S}(t)$. Since A is a constant, we aim to minimize $VU(t) + \sum_{i \in \mathcal{I}} X_i(t)(C_i(t) - D_i(t))$. Moreover, recall that in $\mathbf{P1}$, constraints (3.2), (3.10)-(3.12) couple the energy levels of users' energy storage devices among all the time slots. We can break this coupling by leaving (3.2), (3.10) out, and relaxing (3.11), (3.12) into two constraints as follows:

$$0 \leq \quad C_i(t) \quad \leq C_i^{max} \tag{3.32}$$

$$0 \leq \quad D_i(t) \quad \leq D_i^{max}. \tag{3.33}$$

Therefore, we can formulate a relaxed optimization problem called $\mathbf{P2}$ in the following:

$$\textbf{P2:} \quad \textbf{Minimize} \quad VU(t) + \sum_{i \in \mathcal{I}} X_i(t)(C_i(t) - D_i(t)) \tag{3.34}$$

$$\textbf{s.t.} \quad (3.1), (3.3), (3.4), (3.9), (3.13) - (3.20), (3.32), (3.33)$$

Our dynamic energy management is carried out as follows. The utility company solves the problem $\mathbf{P2}$ in each time slot t given $\mathbf{X}(t)$ and $\mathbf{S}(t)$ collected from the users. It then sends the obtained control decisions to the users, who follow the instructions and update their stored energy levels $\mathbf{X}(t)$ according to (3.24) and (3.2). We denote the corresponding long-term time averaged expected total cost, i.e., $\overline{U}$, by $\mathbf{P2}^*$.

3.3.2.2 Theorem 1

Define the maximum value of V as

$$V^{max} = \min_{i \in \mathcal{I}} \frac{B_i^{max} - C_i^{max} - D_i^{max}}{\beta^{max} - \beta^{min}}. \tag{3.35}$$

For $0 \leq B_i(0) \leq B_i^{max}$ for all $i \in \mathrm{I}$ and any $0 \leq V \leq V^{max}$, our dynamic energy management scheme has the following properties:

1. *An arbitrary user i's stored energy level $B_i(t)$ satisfies the constraint (3.10), i.e., $0 \leq B_i(t) \leq B_i^{max}$, for all $t \geq 0$, and is strongly stable.*
2. *The obtained control decisions are feasible solutions to* $\mathbf{P1}$.
3. $$\mathbf{P2}^* - A/V \leq \mathbf{P1}^* \leq \mathbf{P2}^*. \tag{3.36}$$

Proof:

a) We prove 1. by induction. Particularly, assume that for an arbitrary user i, (3.10) holds in time slot t. Then, we consider the following cases to prove that (3.10) also holds in time slot $t+1$.

First, $0 \leq B_i(t) < D_i^{max}$. Recall that $C_i(t) = c_i^g(t) + c_i^r(t)$. In this case, the partial derivative of the objective function of **P2**, denoted by $P2(t)$, with respect to $c_i^g(t)$, is

$$\begin{aligned}\frac{\partial P2(t)}{\partial c_i^g(t)} &= V\frac{\partial U(t)}{\partial c_i^g(t)} + X_i(t) \\ &\leq V\beta^{max} + B_i(t) - V\beta^{max} - D_i^{max} \\ &< 0. \end{aligned} \tag{3.37}$$

Similarly, we can have

$$\frac{\partial P2(t)}{\partial c_i^r(t)} = X_i(t) < -V\beta^{max} < 0. \tag{3.38}$$

Thus, by solving $\mathbf{P2}$, i.e., minimizing $P2(t)$, our energy management scheme leads to control decisions that satisfy $C_i(t) = c_i^r(t) + c_i^g(t) = C_i^{max}$. Due to constraint (3.9), we have $D_i(t) = 0$. Therefore, according to (3.2), we get $B_i(t+1) = B_i(t) + C_i^{max}$ and hence

$$0 \leq B_i(t+1) \leq D_i^{max} + C_i^{max} \leq B_i^{max} \tag{3.39}$$

due to constraint (3.13).

Second, $D_i^{max} \leq B_i(t) \leq V(\beta^{max} - \beta^{min}) + D_i^{max}$. Since

$$V \leq V^{max} \leq \frac{B_i^{max} - C_i^{max} - D_i^{max}}{\beta^{max} - \beta^{min}}, \tag{3.40}$$

we have $B_i(t) \leq B_i^{max} - C_i^{max}$. Thus, according to (3.2), we can obtain

$$B_i(t+1) \leq B_i^{max} - C_i^{max} + C_i(t) - D_i(t) \leq B_i^{max} \tag{3.41}$$

and

$$B_i(t+1) \geq D_i^{max} + C_i(t) - D_i(t) \geq 0. \tag{3.42}$$

Third, $V(\beta^{max} - \beta^{min}) + D^{max} < B_i(t) \leq B_i^{max}$. Note that $V \leq \frac{B_i^{max} - C_i^{max} - D_i^{max}}{\beta^{max} - \beta^{min}}$, and hence $V(\beta^{max} - \beta^{min}) + D^{max} \leq B_i^{max} - C_i^{max} < B_i^{max}$. The partial derivative of the objective function of **P2** with respect to $d_i^g(t)$ is

$$\begin{aligned} \frac{\partial P2(t)}{\partial d_i^g(t)} &= -V\frac{\partial U(t)}{\partial d_i^g(t)} - X_i(t) \\ &\leq -V\beta^{min} - B_i(t) + V\beta^{max} + D_i^{max} \\ &< 0. \end{aligned} \tag{3.43}$$

Similarly, we can also get that $\partial P2(t)/\partial d_i^l(t) < 0$. Thus, our energy management scheme minimizing $P2(t)$ results in control decisions that satisfy $D_i(t) = d_i^g(t) + d_i^l(t) = D_i^{max}$. Due to constraint (3.9), we have $C_i(t) = 0$. Thus, according to (3.2), we get $B_i(t+1) = B_i(t) - D_i^{max}$ and hence

$$0 \leq B_i(t+1) \leq B_i^{max} - D_i^{max} \leq B_i^{max}. \tag{3.44}$$

Therefore, we can see that (3.10) holds for all $t \geq 0$.

b) We have known from 1) that constraint (3.10) holds. Besides, according to (3.2), we have $C_i(t) = B_i(t+1) - B_i(t) + D_i(t) \leq B_i^{max} - B_i(t) + D_i(t)$. Due to constraint (3.9), we have that $D_i(t) = 0$ when $C_i(t) > 0$. Thus, we get $C_i(t) \leq B_i^{max} - B_i(t)$. Furthermore, we have $B_i(t+1) = B_i(t) + C_i(t) - D_i(t) \geq 0$, which leads to $D_i(t) \leq B_i(t) + C_i(t)$. Similarly, since $C_i(t) = 0$ when $D_i(t) > 0$, we get $D_i(t) \leq B_i(t)$. Therefore, both (3.11) and (3.12) hold as well. In addition, our dynamic energy management scheme updates the stored energy levels $\mathbf{X}(\mathbf{t})$ according to (3.24), which means (3.24) holds too. As a result, the control decisions obtained by our dynamic energy management scheme satisfy all the constraints of $\mathbf{P1}$, and hence are feasible solutions to $\mathbf{P1}$.

c) Denote by $\widehat{C}_i(t)$, $\widehat{D_i}(t)$, and $\widehat{U}(t)$ the results obtained by our dynamic energy management scheme in time slot t, i.e., based on the optimal solution to $\mathbf{P2}$. We also denote by $C_i^*(t)$, $D_i^*(t)$, and $U^*(t)$ the results that we get for time slot t based on the optimal solution to $\mathbf{P1}$. Thus, from Lemma 1 in Section 3.3.2.1, we can have

$$\begin{aligned}
&\Delta(t) + V\mathbb{E}\{\widehat{U}(t)|\mathbf{X}(t)\} \\
\leq& A + V\mathbb{E}\{\widehat{U}(t)|\mathbf{X}(t)\} + \sum_{i\in\mathcal{I}} X_i(t)\mathbb{E}\{\widehat{C}_i(t) - \widehat{D_i}(t)|\mathbf{X}(t)\} \\
\leq& A + V\mathbb{E}\{U^*(t)|\mathbf{X}(t)\} + \sum_{i\in\mathcal{I}} X_i(t)\mathbb{E}\{C_i^*(t) - D_i^*(t)|\mathbf{X}(t)\} \\
=& A + V\mathbb{E}\{U^*(t)\} + \sum_{i\in\mathcal{I}} X_i(t)\mathbb{E}\{C_i^*(t) - D_i^*(t)\}
\end{aligned} \tag{3.45}$$

Note that the last step is due to the fact that the optimal solutions to $\mathbf{P1}$ are obtained independent of the current stored energy levels.

Besides, since the system state $\mathbf{S}(t)$ is i.i.d., it follows that $C_i^*(t)$ and $D_i^*(t)$ are also i.i.d. stochastic processes. Recall the strong law of large numbers: If $\{a(t)\}_{t=0}^{\infty}$ are i.i.d. random variables, we have $\Pr(\frac{1}{T}\lim_{T\to\infty}\sum_{t=0}^{T-1} a(t) = \mathbb{E}\{a(t)\}) = 1$ almost surely. Consequently, we get

$$\begin{aligned}
&\mathbb{E}\{L(\mathbf{X}(t+1)) - L(\mathbf{X}(t))|\mathbf{X}(t)\} + V\mathbb{E}\{\widehat{U}(t)|\mathbf{X}(\mathbf{t})\} \\
\leq &A + V\mathbb{E}\{U^*(t)\} + \sum_{i\in\mathcal{I}} X_i(t) \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\Big(\big(C_i^*(t) - D_i^*(t)\big) \qquad (3.46)
\end{aligned}$$

Taking expectation of the above inequality, we get

$$\begin{aligned}
&\mathbb{E}\{L(\mathbf{X}(t+1))\} - \mathbb{E}\{L(\mathbf{X}(t))\} + V\mathbb{E}\{\widehat{U}(t)\} \\
\leq &A + V\mathbb{E}\{U^*(t)\} \\
&+\sum_{i\in\mathcal{I}}\mathbb{E}\{X_i(t)\}\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\Big(\mathbb{E}\{C_i^*(t) - D_i^*(t)\} \\
= &A + V\mathbb{E}\{U^*(t)\} + \sum_{i\in\mathcal{I}}\Big(\mathbb{E}\{X_i(t)\}(\overline{C_i^*} - \overline{D_i^*}). \qquad (3.47)
\end{aligned}$$

In addition, summing (3.2) over all the time slots $t \in \{0, 1, 2, ..., T-1\}$ and taking expectation on both sides, we have

$$\mathbb{E}\{B_i(T)\} - \mathbb{E}\{B_i(0)\} = \sum_{t=0}^{T-1}\Big(\mathbb{E}\{C_i^*(t) - D_i^*(t)\} \qquad (3.48)$$

Dividing the above equation by T and taking limits as $T \to \infty$, we get $\overline{C_i^*} - \overline{D_i^*} = 0$. Therefore, we can obtain

$$\begin{aligned}
&\mathbb{E}\{L(\mathbf{X}(t+1))\} - \mathbb{E}\{L(\mathbf{X}(t))\} + V\mathbb{E}\{\widehat{U}(t)\} \\
\leq \;\; &A + V\mathbb{E}\{U^*(t)\}. \qquad (3.49)
\end{aligned}$$

Summing the above inequality over all the time slots $t \in \{0, 1, 2, ..., T-1\}$, we get

$$\begin{aligned}
\sum_{t=0}^{T-1} V\mathbb{E}\{\widehat{U}(t)\} \;\; \leq \;\; &AT + V\sum_{t=0}^{T-1}\Big(\mathbb{E}\{U^*(t)\} - \mathbb{E}\{L(\mathbf{X}(T))\} \\
&+\mathbb{E}\{L(\mathbf{X}(0))\} \qquad (3.50)
\end{aligned}$$

Since $0 \leq B_i(t) \leq B_i^{max}$ for all $t \geq 0$, $X_i(t)$ is finite in all time slots as well. Then, dividing both sides of the above equality by VT and taking limits as $T \to \infty$, we can obtain

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{\widehat{U}(t)\} \leq \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{U^*(t)\} + \frac{A}{V}, \tag{3.51}$$

which means $\mathbf{P1}^* \geq \mathbf{P2}^* - A/V$.

Besides, as shown in b), the optimal solutions to $\mathbf{P2}$ are also a feasible solution to $\mathbf{P1}$. Thus, the value of the objective function of $\mathbf{P1}$ calculated based on the optimal solution to $\mathbf{P2}$, i.e., $\mathbf{P2}^*$, is an upper bound on $\mathbf{P1}^*$, i.e, $\mathbf{P1}^* \leq \mathbf{P2}^*$.

We have now finished the proof. ∎

3.4 Dynamic Energy Management with Mixed Load Demands

In this section, we extend the basic system model described in Section 3.2 to the case that users have both delay intolerant (DI) and delay tolerant (DT) load demands. In particular, the same as before, DI load demands need to be satisfied in the same time slots when they are requested without any delay. In contrast, DT load demands just need to be served before some user-defined deadlines. Examples for DT load demands are washer/dryer machines, dishwashers, etc.

3.4.1 Mixed Load Demand Model

Consider that an arbitrary user i has both DI and DT load demands. DI load demands are modeled in the same way as described in Section 3.2.2. We denote user i's DT load demand in time slot t by $T_i(t)$. We also assume that $\{T_i(t)\}_{t=0}^{\infty}$ is an i.i.d. non-negative stochastic process, and $0 \leq T_i(t) \leq T_i^{max}$ for all $t \geq 0$. Besides, we assume $T_i^{max} \leq G_i^{max}$.

It means that the DT load demand that a user can have in one slot is no larger than the maximum amount of energy it can draw from the power grid, which is reasonable.

User i's DT demand is placed in a local queue $Q_i(t)$, which is updated as follows:

$$Q_i(t+1) = \max[Q_i(t) - y_i(t), 0] + T_i(t) \tag{3.52}$$

where $y_i(t) = d_i^q(t) + g_i^q(t) + r_i^q(t)$ is the amount of service received by the queue. Particularly, $d_i^q(t)$, $g_i^q(t)$, and $r_i^q(t)$ are the energy drawn from user i's energy storage device, the power grid, and user i's renewable energy resource in time slot t to support user i's DT demand, respectively.

Due to the introduction of DT load demands, constraint (3.1) changes into:

$$e_i(t) = r_i^l(t) + r_i^q(t) + r_i^g(t) + c_i^r(t). \tag{3.53}$$

$C_i(t)$ remains the same, while $D_i(t)$ changes from (3.4) into:

$$D_i(t) = d_i^g(t) + d_i^l(t) + d_i^q(t). \tag{3.54}$$

$M_i(t)$ remains the same, while $G_i(t)$ changes from (3.17) into:

$$0 \le G_i(t) = g_i^l(t) + g_i^q(t) + c_i^g(t) \le G_i^{max}, \tag{3.55}$$

and (3.16) changes into

$$\mathbf{1}_{g_i^l(t)+c_i^g(t)+g_i^q(t)>0} + \mathbf{1}_{d_i^g(t)+r_i^g(t)>0} \le 1. \tag{3.56}$$

Besides, the amount of energy supplied by the utility company in time slot t changes from (3.14) into

$$P(t) = \sum_{i\in\mathcal{I}} \big(l_i(t) + g_i^q(t) + c_i^g(t) - r_i^l(t) - r_i^g(t) - d_i^g(t) - d_i^l(t)\big). \tag{3.57}$$

3.4.2 Problem Formulation with Mixed Load Demand Model

Let $\mathbf{H}(t) = \{H_1(t), H_2(t), \ldots, H_n(t)\}$ be the vector of decision variables in the system, where $H_i(t) = \{g_i^l(t), g_i^q(t), d_i^g(t), d_i^l(t), d_i^q(t), c_i^r(t), c_i^g(t), r_i^g(t), r_i^l(t), r_i^q(t)\}$. We also denote the system state by a vector of random variables, i.e., $\mathbf{S}(t) = \{S_1(t), S_2(t), \ldots, S_n(t), R(t)\}$ where $S_i(t) = \{l_i(t), T_i(t), e_i(t)\}$. Thus, the dynamic energy management problem with mixed load demand model, which we call **P3**, can be formulated as follows:

$$\textbf{P3:} \quad \textbf{Minimize} \quad \overline{U} \tag{3.58}$$

s.t. DT loads are served before user-defined deadlines

Constraints: $(3.2), (3.3), (3.9) - (3.13), (3.15), (3.18) - (3.20),$

$(3.53) - (3.57), \ \forall t \geq 0$

We denote the optimal result, i.e., the minimum of the objective function, of $\mathbf{P3}$ by $\mathbf{P3}^*$. We notice that $\mathbf{P3}$ is also a time-coupling optimization problem, which is prohibitively difficult to solve as explained in Section 3.3.1. Similarly, in what follows we reformulate this problem based on Lyapunov optimization theory such that it can be solved based on current system state only.

3.4.3 Delay Aware Virtual Queue

In order to characterize the delay in serving users' DT load demand, we define a delay-aware virtual queue $Z_i(t)$ for each user i, whose queueing function is as follows:

$$Z_i(t+1) = \max[Z_i(t) - y_i(t), \ 0] + \epsilon_i \mathbf{1}_{Q_i(t)>0}. \tag{3.59}$$

In particular, $Z_i(t)$ has the same serving rate as $Q_i(t)$, but a different arrival rate. ϵ_i is a constant related to user-defined service deadline, which will be specified in Lemma 2 in Section 3.4.3.1. We also assume that $\epsilon_i \leq G_i^{max}$, i.e., the arriving rate is no larger than the maximum amount of energy user i can draw from the power grid. We have the following lemma.

3.4.3.1 Lemma 2

Assume that the queues $\mathbf{Q}(t)$ and $\mathbf{Z}(t)$ are controlled in such a way that $Q_i(t) < Q_i^{max}$ and $Z_i(t) < Z_i^{max}$ for all $t \geq 0$ and $i \in \mathcal{I}$, where Z_i^{max} and Q_i^{max} are deterministic positive constants. Then, an arbitrary user i's DT load demand $T_i(t)$ can be served within a maximum delay of

$$\mu_i^{max} = \lceil \frac{Q_i^{max} + Z_i^{max}}{\epsilon_i} \rceil. \tag{3.60}$$

Proof: In what follows, we prove (3.60) by contradiction.

Assume that the delay in serving an arbitrary user i's DT demand is larger than μ_i^{max}. Suppose $T_i(t) > 0$ in time slot t. Thus, we have $Q(t+1) > 0$ according to (3.52), and $Q(\tau) > 0$ for $t+1 \leq \tau \leq t + \mu_i^{max}$. Referring to (3.59), we get

$$Z_i(\tau + 1) \geq Z_i(\tau) - y_i(\tau) + \epsilon_i \tag{3.61}$$

for $t+1 \leq \tau \leq t + \mu_i^{max}$. Summing over the time slots from $t+1$ to $t + \mu_i^{max}$ yields

$$Z_i(t + \mu_i^{max} + 1) - Z_i(t+1) \geq \mu_i^{max}\epsilon_i - \sum_{\tau=t+1}^{t+\mu_i^{max}} \Big(y_i(\tau) \tag{3.62}$$

Since $Z(t + \mu_i^{max} + 1) \leq Z_{max}$ and $Z(t + 1) \geq 0$, we can get

$$\sum_{t+1}^{t+\mu_i^{max}} \Bigg(y_i(t) \geq \mu_i^{max}\, \epsilon_i - Z_i^{max} \tag{3.63}$$

Consider that the DT loads are served in a first in first out (FIFO) manner. Since $Q_i(t + 1) < Q_i^{max}$ and user i's DT demand $T_i(t)$ has not been served by $t + \mu_i^{max}$, we have $\sum_{\tau=t+1}^{t+\mu_i^{max}} y_i(t) < Q_i^{max}$. Thus, from (3.63) we can get

$$Q_i^{max} + Z_i^{max} > \mu_i^{max}\epsilon_i. \tag{3.64}$$

Since $\mu_i^{max} = \lceil \frac{Q_i^{max}+Z_i^{max}}{\epsilon_i} \rceil$, we have $Q_i^{max} + Z_i^{max} > Q_i^{max} + Z_i^{max}$, which is impossible. Thus, the assumption that the delay in serving an arbitrary user i's DT demand is larger than μ_i^{max} is invalid, and Lemma 2 in Section 3.4.3.1 follows. ■

According to Lemma 2 in Section 3.4.3.1, each user i can set ϵ_i based on Q_i^{max} and Z_i^{max} to make sure that its DT load demand can be satisfied by a certain deadline. We will describe Q_i^{max} and Z_i^{max} in detail later. We are now ready to present our Lyapunov optimization based energy management scheme.

3.4.4 Dynamic Energy Management based on Lyapunov Optimization

Notice that the queues that are maintained in the system can be denoted by a vector $\mathbf{\Theta}(t) = \{\mathbf{X}(t), \mathbf{Q}(t), \mathbf{Z}(t)\}$. Thus, we can define a Lyapunov function as

$$L(\mathbf{\Theta}(t)) = \frac{1}{2}\sum_{i\in\mathcal{I}} \Big(\Bigg((X_i(t))^2 + (Q_i(t))^2 + (Z_i(t))^2 \Big), \Bigg(\tag{3.65}$$

and the one-slot conditional Lyapunov drift is

$$\Delta(\mathbf{\Theta}(t)) = \mathbb{E}\{L(\mathbf{\Theta}(t + 1)) - L(\mathbf{\Theta}(t))|\mathbf{\Theta}(t)\}. \tag{3.66}$$

Recall that $y_i(t) = d_i^q(t) + g_i^q(t) + r_i^q(t)$. Since $0 \leq d_i^q(t) \leq D_i^{max}$, $0 \leq g_i^q(t) \leq G_i^{max}$, and $0 \leq r_i^q(t) \leq e_i(t) \leq e_i^{max}$, we have $0 \leq y_i(t) \leq D_i^{max} + G_i^{max} + e_i^{max}$. We denote the upper bound on $y_i(t)$ as y_i^{max}. Then, we can have the following lemma regarding the drift-plus-penalty function.

3.4.4.1 Lemma 3

Given $\Delta(\boldsymbol{\Theta}(t))$ defined in (3.66), we have

$$
\begin{aligned}
\Delta(\boldsymbol{\Theta}(t)) + & V\mathbb{E}\{U(t)|\boldsymbol{\Theta}(\mathbf{t})\} \\
& \leq K + V\mathbb{E}\{U(t)|\boldsymbol{\Theta}(\mathbf{t})\} \\
& + \sum_{i \in \mathcal{I}} \Big(X_i(t)\mathbb{E}\{C_i(t) - D_i(t)|\boldsymbol{\Theta}(\mathbf{t})\} \\
& + \sum_{i \in \mathcal{I}} \Big(Q_i(t)\mathbb{E}\{T_i(t) - y_i(t)|\boldsymbol{\Theta}(\mathbf{t})\} \\
& + \sum_{i \in \mathcal{I}} \Big(Z_i(t)\mathbb{E}\{\epsilon_i - y_i(t)|\boldsymbol{\Theta}(\mathbf{t})\}
\end{aligned}
\tag{3.67}
$$

where K is a constant, i.e.,

$$
\begin{aligned}
K = \sum_{i \in \mathcal{I}} \Big(& \frac{\max[(C_i^{max})^2, (D_i^{max})^2]}{2} + \frac{(T_i^{max})^2 + (y_i^{max})^2}{2} \\
& + \frac{\epsilon_i^2 + (y_i^{max})^2}{2} \Big). \Big(
\end{aligned}
\tag{3.68}
$$

Proof: We have obtained in Lemma 1 in Section 3.3.2.1 that

$$
\begin{aligned}
& \frac{X_i^2(t+1) - X_i^2(t)}{2} \\
\leq & \frac{\max[(C_i^{max})^2, (D_i^{max})^2]}{2} + X_i(t)(C_i(t) - D_i(t)).
\end{aligned}
\tag{3.69}
$$

Besides, note that $\forall x, y, z$ with $x \geq 0$, $0 \leq y \leq y_{max}$, $0 \leq z \leq z_{max}$, we have

$$\begin{aligned}(\max\{x-y,0\}+z)^2 &\leq x^2+y^2+z^2+2x(z-y)\\ &\leq x^2+y_{max}^2+z_{max}^2+2x(z-y).\end{aligned} \tag{3.70}$$

Thus, squaring both sides of (3.52), we get

$$\begin{aligned}&\frac{Q_i(t+1)^2-Q_i(t)^2}{2}\\ \leq&\frac{(T_i^{max})^2+(y_i^{max})^2}{2}+Q_i(t)\left(T_i(t)-y_i(t)\right)\end{aligned} \tag{3.71}$$

Similarly, squaring both sides of (3.59), we have

$$\frac{Z_i(t+1)^2-Z_i(t)^2}{2} \leq \frac{\epsilon_i^2+(y_i^{max})^2}{2}+Z_i(t)(\epsilon_i-y_i(t)). \tag{3.72}$$

Therefore, summing (3.69), (3.71), and (3.72) over all $i \in \mathcal{I}$, taking expectations conditioned on $\mathbf{\Theta}(t)$, and adding the cost function $V\mathbb{E}\{U(t)|\mathbf{\Theta}(t)\}$, we arrive at Lemma 3 in Section 3.4.4.1. ■

Similar to that in Section 3.3.2, we aim to minimize the right-hand side of (3.67) in each time slots t based on current system state. Note that in (3.67) $T_i(t)$ is a constant given the current system state, and ϵ_i is a constant, too. Thus, removing the constants and relaxing the constraints (3.2), (3.10)-(3.12), we formulate a new problem $\mathbf{P4}$ as follows:

$$\textbf{P4:}\quad \textbf{Minimize}\quad \begin{aligned}&VU(t)+\sum_{i\in\mathcal{I}}\Big(X_i(t)(C_i(t)-D_i(t))\\ &\qquad -(Q_i(t)+Z_i(t))y_i(t)\Big)\Big(\end{aligned} \tag{3.73}$$

$$\begin{aligned}\textbf{s.t.}\quad &\text{Constraints: } (3.3),(3.9),(3.13),(3.15),(3.18)-(3.20),\\ &(3.32),(3.33),(3.53)-(3.57)\end{aligned}$$

Our dynamic energy management scheme works as follows. The utility company solves the problem $\mathbf{P4}$ in each time slot t given $\mathbf{\Theta}(t)$ and $\mathbf{S}(t)$ collected from the users. It then sends the obtained control decisions to the users, who follow the instructions and update their queues $\mathbf{X}(\mathbf{t})$, $\mathbf{Q}(t)$, and $\mathbf{Z}(t)$ in the system according to (3.24) and (3.2), (3.52), and (3.59), respectively. We denote the corresponding long-term time averaged expected total cost, i.e., $\overline{U}$, by $\mathbf{P4}^*$.

3.4.4.2 Theorem 2

Define the maximum value of V as

$$V^{max} = \min_{i \in \mathcal{I}} \frac{B_i^{max} - C_i^{max} - D_i^{max} - N_i}{\beta^{max}}. \tag{3.74}$$

where $N_i = \frac{4B_i^{max} - 4C_i^{max} - 2D_i^{max} + 3T_i^{max} + 3\epsilon_i}{7}$. Assume $B_i^{max} \gg C_i^{max} + D_i^{max} + T_i^{max} + \epsilon_i$. Suppose all DT load demand queues and virtual queues start with zero backlogs, i.e., $Q_i(0) = Z_i(0) = 0$ for all $i \in \mathcal{I}$, and all energy storage devices start with feasible energy levels, i.e., $0 \le B_i(0) \le B_i^{max}$ for all $i \in \mathcal{I}$. Then, for any $0 \le V \le V^{max}$, our dynamic energy management scheme has the following properties:

1. *For an arbitrary user i, its queues $Q_i(t)$ and $Z_i(t)$ are deterministically upper bounded by constants Q_i^{max} and Z_i^{max}, respectively, for all $t \ge 0$ where*

$$Q_i^{max} = \frac{2V\beta^{max} + D_i^{max}}{3} + T_i^{max} \tag{3.75}$$

$$Z_i^{max} = \frac{2V\beta^{max} + D_i^{max}}{3} + \epsilon_i \tag{3.76}$$

2. *For an arbitrary user i, its stored energy level $B_i(t)$ satisfies (3.10), i.e., $0 \le B_i(t) \le B_i^{max}$ for all $t \ge 0$.*

3. *For an arbitrary user i, its DT load demand can be served with a maximum delay of*

$$\mu_i^{max} = \lceil \frac{4V\beta^{max} + 2D_i^{max} + 3T_i^{max} + 3\epsilon_i}{3\epsilon_i} \rceil. \tag{3.77}$$

4. *The obtained control solutions are feasible solutions to* $\mathbf{P3}$.

5.
$$\mathbf{P4}^* - K/V \le \mathbf{P3}^* \le \mathbf{P4}^*. \tag{3.78}$$

Proof:

a) We first prove (3.75) by induction. Obviously, (3.75) holds for $t = 0$. Assume that (3.75) holds in time slot t. In the following, we show that (3.75) also holds in time slot $t + 1$.

First, $0 \le Q_i(t) \le \frac{2V\beta^{max}+D_i^{max}}{3}$. Since $T_i(t) \le T_i^{max}$, then according to (3.52), we have

$$Q_i(t+1) \le \max[Q_i(t) - y_i(t) + T_i^{max}, T_i^{max}]. \tag{3.79}$$

Thus, we get $Q_i(t+1) \le Q_i(t) + T_i^{max} \le \frac{2V\beta^{max}+D_i^{max}}{3} + T_i^{max}$.

Second, $\frac{2V\beta^{max}+D_i^{max}}{3} < Q_i(t) \le \frac{2V\beta^{max}+D_i^{max}}{3} + T_i^{max}$. In this case, the partial derivative of the objective function of $\mathbf{P4}$, denoted by $P4(t)$, with respect to $g_i^q(t)$, is

$$\begin{aligned}\frac{\partial P4(t)}{\partial g_i^q(t)} &= V\frac{\partial U(t)}{\partial g_i^q(t)} - (Q_i(t) + Z_i(t)) \\ &\le V\beta^{max} - (Q_i(t) + Z_i(t)).\end{aligned} \tag{3.80}$$

Similarly, we can have

$$\frac{\partial P4(t)}{\partial r_i^q(t)} = -(Q_i(t) + Z_i(t)), \tag{3.81}$$
$$\frac{\partial P4(t)}{\partial d_i^q(t)} = -(X_i(t) + Q_i(t) + Z_i(t)). \tag{3.82}$$

Since $X_i(t) \ge -V\beta^{max} - D_i^{max}$ and $Q_i(t) > \frac{2V\beta^{max}+D_i^{max}}{3}$, we get

$$\begin{aligned}&\frac{\partial P4(t)}{\partial g_i^q(t)} + \frac{\partial P4(t)}{\partial r_i^q(t)} + \frac{\partial P4(t)}{\partial d_i^q(t)} \\ \le\ & V\beta^{max} - X_i(t) - 3Q_i(t) - 3Z_i(t) < 0.\end{aligned} \tag{3.83}$$

Thus, our dynamic energy scheme that minimizes $P4(t)$ will choose $y_i(t)$ to be its maximum value. Since $y_i(t) = d_i^q(t) + g_i^q(t) + r_i^q(t)$ where $d_i^q(t) \leq D_i^{max}$, $g_i^q(t) \leq G_i^{max}$, and $r_i^q(t) \leq e_i(t)$, the maximum of $y_i(t)$, denoted by $y_i^{max}(t)$, is $y_i^{max}(t) = D_i^{max} + G_i^{max} + e_i(t)$.

- If $Q_i(t) \geq y_i^{max}(t)$, we have $Q_i(t+1) = Q_i(t) - y_i^{max}(t) + T_i(t)$. Since $T_i(t) \leq T_i^{max} \leq G_i^{max}$, we get $T_i(t) \leq y_i^{max}(t)$ and hence $Q_i(t+1) \leq Q_i(t) \leq \frac{2V\beta^{max}+D_i^{max}}{3} + T_i^{max}$.
- If $Q_i(t) < y_i^{max}(t)$, we have $Q_i(t+1) = T_i(t) \leq T_i^{max} \leq \frac{2V\beta^{max}+D_i^{max}}{3} + T_i^{max}$.

As a result, (3.75) holds for all $t \geq 0$.

Next, we prove (3.76) by induction. Note that (3.76) holds for $t = 0$. Assume that (3.76) holds in time slot t. In what follows, we show that (3.76) also holds in time slot $t + 1$.

First, $0 \leq Z_i(t) \leq \frac{2V\beta^{max}+D_i^{max}}{3}$. According to (3.59), we have $Z_i(t+1) \leq \max[Z_i(t) - y_i(t) + \epsilon_i, \epsilon_i]$. Thus, we get $Z_i(t+1) \leq Z_i(t) + \epsilon_i \leq \frac{2V\beta^{max}+D_i^{max}}{3} + \epsilon_i$.

Second, $\frac{2V\beta^{max}+D_i^{max}}{3} < Z_i(t) \leq \frac{2V\beta^{max}+D_i^{max}}{3} + \epsilon_i$. From (3.80)-(3.82), we can have

$$\begin{aligned}
& \frac{\partial P4(t)}{\partial g_i^q(t)} + \frac{\partial P4(t)}{\partial r_i^q(t)} + \frac{\partial P4(t)}{\partial d_i^q(t)} \\
\leq\ & V\beta^{max} - X_i(t) - 3Q_i(t) - 3Z_i(t) \\
\leq\ & 2V\beta^{max} + D_i^{max} - 3Z_i(t) \\
<\ & 0
\end{aligned} \tag{3.84}$$

due to $X_i(t) \geq -V\beta^{max} - D_i^{max}$ and $Q_i(t) \geq 0$. Thus, our dynamic energy scheme minimizing $P4(t)$ will choose $y_i(t) = y_i^{max}(t) = D_i^{max} + G_i^{max} + e_i(t)$ as shown above.

- If $Z_i(t) \geq y_i^{max}(t)$, we have $Z_i(t+1) \leq Z_i(t) - y_i^{max}(t) + \epsilon_i$. Since $\epsilon_i \leq G_i^{max}$, we get $T_i(t+1) \leq Z_i(t) \leq \frac{2V\beta^{max}+D_i^{max}}{3} + \epsilon_i$.

- If $Q_i(t) < y_i^{max}(t)$, we have $Z_i(t+1) = \epsilon_i \leq \frac{2V\beta^{max}+D_i^{max}}{3} + \epsilon_i$.

Therefore, (3.76) holds for all $t \geq 0$.

b) We prove 2. by induction. Assume that for an arbitrary user i, (3.10) holds in time slot t. Then, we consider the following cases to prove that (3.10) also holds in time slot $t+1$.

First, $0 \leq B_i(t) < D_i^{max}$. This case is identical to the first case of Theorem 1a in Section 3.3.2.2. Thus, our energy management scheme takes control decisions c_i^r and c_i^g, such that $C_i(t) = c_i^r(t) + c_i^g(t) = C_i^{max}$. Due to constraint (3.9), we also have $D_i(t) = 0$. Thus, according to (3.2), we get $B_i(t+1) = B_i(t) + C_i^{max}$ and

$$0 \leq B_i(t+1) \leq D_i^{max} + C_i^{max} \leq B_i^{max} \tag{3.85}$$

due to constraint (3.13).

Second, $D_i^{max} \leq B_i(t) \leq V\beta^{max} + D_i^{max} + Q_i^{max} + Z_i^{max}$. Since

$$V \leq V^{max} \leq \frac{B_i^{max} - C_i^{max} - D_i^{max} - N_i}{\beta^{max}}, \tag{3.86}$$

according to (3.75) and (3.76), we have

$$\begin{aligned}
&B_i(t)\\
\leq &B_i^{max} - C_i^{max} + \frac{4V\beta^{max} + 2D_i^{max} + 3T_i^{max} + 3\epsilon_i}{3} - N_i\\
\leq &B_i^{max} - C_i^{max} +\\
&\frac{4B_i^{max} - 4C_i^{max} - 2D_i^{max} + 3T_i^{max} + 3\epsilon_i - 7N_i}{3}\\
= &B_i^{max} - C_i^{max}.
\end{aligned} \tag{3.87}$$

Thus, according to (3.2), we can obtain

$$B_i(t+1) \leq B_i^{max} - C_i^{max} + C_i(t) - D_i(t) \leq B_i^{max} \tag{3.88}$$

and

$$B_i(t+1) \geq D_i^{max} + C_i(t) - D_i(t) \geq 0. \tag{3.89}$$

Third, $V\beta^{max} + D_i^{max} + Q_i^{max} + Z_i^{max} < B_i(t) \leq B_i^{max}$. Note that we have shown above that $V\beta^{max} + D_i^{max} + Q_i^{max} + Z_i^{max} \leq B_i^{max} - C_i^{max} < B_i^{max}$. The partial derivative of the objective function of **P4** with respect to $d_i^g(t)$ is

$$\begin{aligned} \frac{\partial P4(t)}{\partial d_i^g(t)} &= -V\frac{\partial U(t)}{\partial d_i^g(t)} - X_i(t) \\ &\leq -V\beta^{min} - B_i(t) + V\beta^{max} + D_i^{max} \\ &\leq -V\beta^{min} - Q_i^{max} - Z_i^{max} \\ &< 0. \end{aligned} \tag{3.90}$$

Similarly, we can also get that $\partial P4(t)/\partial d_i^l(t) < 0$. The partial derivative of the objective function of **P4** with respect to $d_i^q(t)$ is

$$\begin{aligned} \frac{\partial P4(t)}{\partial d_i^q(t)} &= -(X_i(t) + Q_i(t) + Z_i(t)) \\ &= -B_i(t) + V\beta^{max} + D_i^{max} - Q_i(t) - Z_i(t) \\ &\leq -V\beta^{max} - D_i^{max} - Q_i^{max} - Z_i^{max} \\ &\quad +V\beta^{max} + D_i^{max} - Q_i(t) - Z_i(t) \\ &< 0. \end{aligned} \tag{3.91}$$

After minimizing $P4(t)$, our energy management scheme results in control decisions that satisfy $D_i(t) = d_i^g(t) + d_i^l(t) + d_i^q(t) = D_i^{max}$. Due to constraint (3.9), we have $C_i(t) = 0$. Thus, according to (3.2), we get $B_i(t+1) = B_i(t) - D_i^{max}$ and hence

$$0 \leq B_i(t+1) \leq B_i^{max} - D_i^{max} \leq B_i^{max}. \tag{3.92}$$

Therefore, we can see that (3.10) holds for all $t \geq 0$.

c) The result (3.77) directly follows Lemma 2 in Section 3.4.3.1.

d) Part a) has shown that constraint (3.10) holds. The same as Theorem 1b in Section 3.3.2.2, we can show that (3.32) and (3.33) hold. Part b) of this theorem has shown that DT load demands can be served before user-defined deadlines. Thus, the control decisions obtained by our dynamic energy management scheme satisfy all the constraints of $\mathbf{P3}$, and hence are feasible solutions to $\mathbf{P3}$.

e) Denote by $\widehat{C}_i(t)$, $\widehat{D_i}(t)$, $\widehat{y}_i(t)$, and $\widehat{U}(t)$ the results obtained by our dynamic energy management scheme in time slot t, i.e., based on the optimal solution to $\mathbf{P4}$. We also denote by $C_i^*(t)$, $D_i^*(t)$, $y_i^*(t)$, and $U^*(t)$ the results that we get for time slot t based on the optimal solution to $\mathbf{P3}$. Thus, from Lemma 3 in Section 3.4.4.1, we can have

$$\begin{aligned}
&\Delta(t) + V\mathbb{E}\{\widehat{U}(t)|\boldsymbol{\Theta}(t)\} \\
&\qquad \le K + V\mathbb{E}\{\widehat{U}(t)|\boldsymbol{\Theta}(t)\} \\
&\qquad\qquad + \sum_{i\in\mathcal{I}} X_i(t)\mathbb{E}\{\widehat{C}_i(t) - \widehat{D}_i(t)|\boldsymbol{\Theta}(t)\} \\
&\qquad\qquad + \sum_{i\in\mathcal{I}} Q_i(t)\mathbb{E}\{T_i(t) - \widehat{y}_i(t)|\boldsymbol{\Theta}(t)\} \\
&\qquad\qquad + \sum_{i\in\mathcal{I}} Z_i(t)\mathbb{E}\{\epsilon_i - \widehat{y}_i(t)|\boldsymbol{\Theta}(t)\} \\
&\qquad \le K + V\mathbb{E}\{U^*(t)\} \\
&\qquad\qquad + \sum_{i\in\mathcal{I}} X_i(t)\mathbb{E}\{C_i^*(t) - D_i^*(t)\} \\
&\qquad\qquad + \sum_{i\in\mathcal{I}} Q_i(t)\mathbb{E}\{T_i(t) - y_i^*(t)\} \\
&\qquad\qquad + \sum_{i\in\mathcal{I}} Z_i(t)\mathbb{E}\{\epsilon_i - y_i^*(t)\}
\end{aligned} \tag{3.93}$$

Note that the second step is based on the fact that the optimal solutions to $\mathbf{P3}$ are obtained independent of the queue state $\boldsymbol{\Theta}(t)$.

Besides, since the system state $\mathbf{S}(t)$ is i.i.d., $C_i^*(t)$, $D_i^*(t)$, and $y_i^*(t)$ are also i.i.d. stochastic processes. Similar to the proof of Theorem 1c in Section 3.3.2.2, applying the strong law of large numbers and taking expectation of both sides, we get

$$\begin{aligned}
&\mathbb{E}\{L(\mathbf{\Theta}(t+1))\} - \mathbb{E}\{L(\mathbf{\Theta}(t))\} + V\mathbb{E}\{\widehat{U}(t)\} \\
\le& K + V\mathbb{E}\{U^*(t)\} \\
&+ \sum_{i\in\mathcal{I}} \left(\left(\mathbb{E}\{X_i(t)\} \cdot \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \left(\mathbb{E}\{C_i^*(t) - D_i^*(t)\}\right)\right.\right. && (3.94) \\
&+ \sum_{i\in\mathcal{I}} \left(\left(\mathbb{E}\{Q_i(t)\} \cdot \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \left(\mathbb{E}\{T_i(t) - y_i^*(t)\}\right)\right.\right. && (3.95) \\
&+ \sum_{i\in\mathcal{I}} \left(\left(\mathbb{E}\{Z_i(t)\} \cdot \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \left(\mathbb{E}\{\epsilon_i - y_i^*(t)\}\right)\right.\right. && (3.96)
\end{aligned}$$

We have shown by (3.48) that $\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{C_i^*(t)\} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{D_i^*(t)\}$. Thus, the component (3.94) is equal to 0. Since $Q_i(t) \le Q_i^{max} < \infty$ for all $t \ge 0$ where Q_i^{max} is a constant defined in (3.52), we have $\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{Q_i(t)\} \le Q_i^{max}$, i.e., queue $Q_i(t)$ is strongly stable [36]. Since $y_i(t) - T_i(t) \le y_i(t) \le y_i^{max}$, we know that queue $Q_i(t)$ is also rate stable (Theorem 2.8, i.e., Strong Stability Theorem, in [36]), i.e., $\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{T_i(t)\} \le \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{y_i^*(t)\}$, which means the component (3.95) is no larger than 0. Similarly, since $y_i(t) - \epsilon_i \mathbf{1}_{Q_i(t)>0} \le y_i^{max}$, queue $Z_i(t)$ is rate stable and $\limsup_{T\to\infty} frac1T \sum_{t=0}^{T-1} \mathbb{E}\{\epsilon_i \mathbf{1}_{Q_i(t)>0}\} \le \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{y_i^*(t)\}$. Thus, we have $\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{\epsilon_i - y_i^*(t)\} \le 0$, i.e., the component (3.96) is no larger than 0. Therefore, we have

$$\begin{aligned}
&\mathbb{E}\{L(\mathbf{\Theta}(t+1))\} - \mathbb{E}\{L(\mathbf{\Theta}(t))\} + V\mathbb{E}\{\widehat{U}(t)\} \\
\le\ & K + V\mathbb{E}\{U^*(t)\} && (3.97)
\end{aligned}$$

Similar to the proof of Theorem 1c in Section 3.3.2.2, summing the above inequality over all the time slots $t \in \{0, 1, 2, ..., T-1\}$, dividing both sides by VT, and taking limits as $T \to \infty$, we can get $\mathbf{P3}^* \geq \mathbf{P4}^* - K/V$.

Besides, as shown in d), the optimal solution to $\mathbf{P4}$ is also a feasible solution to $\mathbf{P3}$. Thus, the value of the objective function of $\mathbf{P3}$ calculated based on the optimal solution to $\mathbf{P4}$, i.e., $\mathbf{P4}^*$, is an upper bound on $\mathbf{P3}^*$, i.e, $\mathbf{P3}^* \leq \mathbf{P4}^*$.

We have now completed the proof.

■

3.5 Simulation Results

In this section, we evaluate the performance of our dynamic energy management scheme using practical renewable energy generation data. We study two cases: when users have DI load demands only and when users have both DI and DT load demands. In each case, we first obtain the lower and upper bounds on the optimal result. Then, we calculate our total energy generation cost and compare it with that of a simple energy management strategy. We implement our proposed dynamic energy management schemes on a general purpose PC with 64-bit Windows 7, 25GB RAM, and a 2.26GHz CPU. Using CPLEX, we solve optimization problems **P2** and **P4** for the two cases, respectively.

Some simulation settings are as follows. We consider 10 users using energy for a period of 10 days with 5-minute long time slots, i.e., 3000 time slots in total. Users' renewable energy generation capabilities are set based on the global horizontal irradiance data for Las Vegas area available at the Measurement and Instrumentation Data Center [32]. In particular, we assume the energy conversion efficiency is 15% and the maximum output is $200W$. Besides, the maximum charging and discharging limits on each user's energy storage device in a time slot, i.e., C_i^{max} and D_i^{max}, are both set to 1.5kWh. The maximum amount of energy that each user can draw from the power grid in a time slot, i.e., G_i^{max}, is set to the maximum load request plus the maximum charging limit in a time slot. The maximum amount of energy that each user can sell to the grid, i.e., M_i^{max}, is set to be the same as G_i^{max}. In addition, we ignore the utility company's renewable energy resource and focus on the management of users' energy resources in this simulation. So the utility company's energy generation cost function is defined as $U(t) = aP^2(t) + bP(t) + c$, where $a = 0.75, b = 0.1$ and $c = 0$.

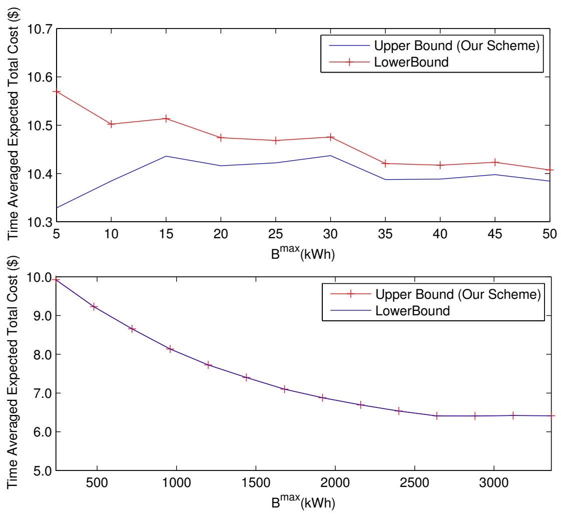
In the case that users have DI load demands only, we consider that each user's DI load demands are i.i.d. uniform random variables over the interval [1,7]kWh. Fig. 3.1(a) shows the upper and lower bounds on the optimal result. Note that the upper bound is the time averaged expected total cost during the whole simulation period, obtained by our dynamic energy management scheme. The lower bound is the upper bound minus A/V as shown in Theorem 1 in Section 3.3.2.2. In our simulations, we set $V = V^{max}$. Recall that according to Theorem 1 in Section 3.3.2.2, A is independent of B^{max} while V^{max} increases as B^{max} increases. Thus, the performance bounds get tighter as B^{max} increases as we can see in Fig. 3.1(a). In addition, we compare in Fig. 3.1(b) the total energy generation cost of our dynamic energy management scheme since $t = 0$ with that of a simple energy management strategy. In particular, the simple strategy satisfies users' DI load demands in the same time slot when they are requested. It does not consider users selling energy to the grid or using energy storage devices. We can observe noticeable savings using our scheme, which keep increasing as time goes by.

In the case that each user has both a DI and a DT load demands, we consider that each user's both load demands are i.i.d. uniform random variables over the interval [1,3.5]kWh. We set all DT load demand deadlines to 168 hours (7 days), i.e., $\mu_i^{max} = 2016$, and set ϵ_i according to (3.77) for each energy storage device size. We show the upper and lower bounds on the optimal result in Fig. 3.2(a), and find that the bounds get tighter as B^{max} increases. We also compare the total energy generation cost of our dynamic energy management scheme since $t = 0$ with that of a simple energy management strategy in Fig. 3.2(b). In particular, the simple strategy satisfies users' DI and DT load demands in the same time slot when they are requested. It does not consider users selling energy to the grid or using energy storage devices. We can observe noticeable savings using our scheme as well. Fig. 3.2(c) shows the time that it takes DT load demands to be satisfied when each user has an energy storage device with capability of $B^{max} = 240$kWh. We observe that all DT loads can be served within 15 hours, much earlier than the user-defined deadline. In Fig. 3.2(d), we present the energy level of a user's energy storage device which always remains within its physical limits as described in Theorem 2 in Section 3.4.4.2.

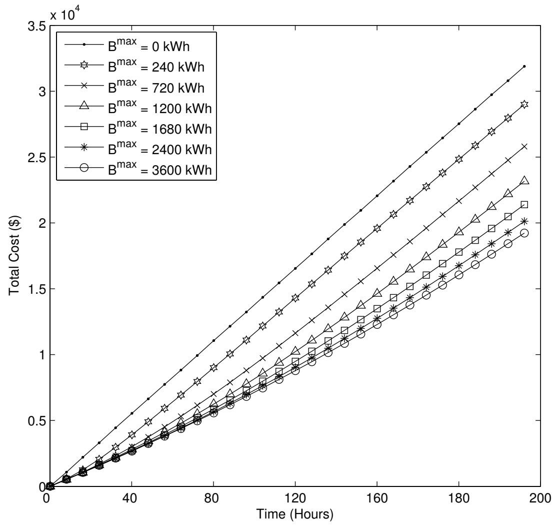
Moreover, although in our dynamic energy management schemes, the optimization problems $\mathbf{P2}$ and $\mathbf{P4}$ need to be solved once every time slot, we find that on average they can be solved in about 0.3 seconds on the PC we use for our simulations. The computation time is very low and can be even lower on more powerful computers.

3.6 Conclusions

In this chapter, we have explored dynamic energy management in the Smart Grid, considering unpredictable load demands, and distributed uncertain renewable energy resources and energy storage devices. We have studied two kinds of user load demands: DI demands only, and both DI and DT demands. In particular, with the objective of minimizing the long-term time averaged expected total cost of supporting all users' load demands, we formulate an optimization problem, which is a time-coupling problem and prohibitively expensive to solve. Then, employing Lyapunov optimization theory, we reformulate the problem and develop a dynamic energy management scheme which can dynamically solve the problem in each time slot. The developed scheme result in both a lower and an upper bound on the optimal result of the original optimization problem. Furthermore, in the case of both DI and DT load demands, we show that DT load demands are guaranteed be served within user-defined deadlines. Extensive simulation results are presented to validate the efficiency of the proposed scheme.



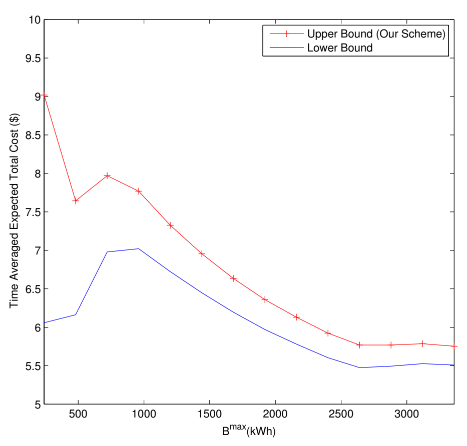
(a) Bounds on optimal time averaged expected total cost.

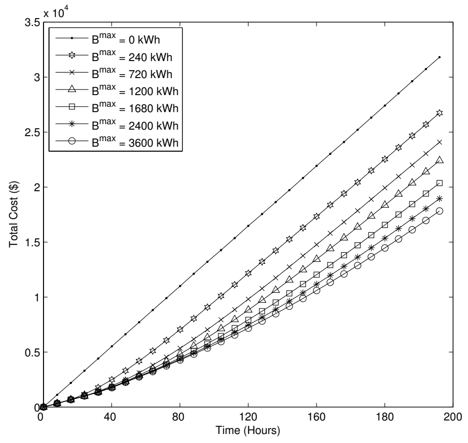


(b) Total generation cost.

Figure 3.1

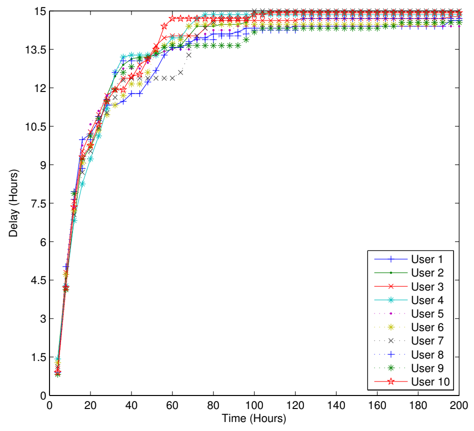
The case of DI load demands.

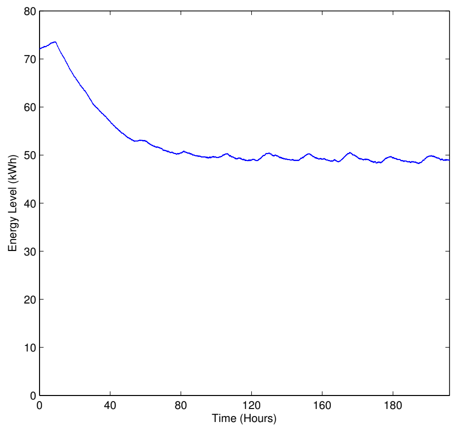




(a) Bounds on optimal time averaged expected total cost.

(b) Total generation cost.





(c) Service delays of DT load demands when $B^{max} = 240$kWh.

(d) Energy storage device behavior when $B^{max} = 240$kWh

Figure 3.2

The case of both DI and DT load demands.

CHAPTER 4

PRIVACY-PRESERVING ENERGY THEFT DETECTION IN SMART GRIDS

4.1 Introduction

Energy theft has been a notorious problem in traditional power systems. The utility companies (UCs) in the U.S. lose approximately six billion dollars every year due to this problem [31]. Recently, the smart grid has been proposed as a new type of electrical grid to modernize current power grids to efficiently deliver reliable, economic, and sustainable electricity services. One of the most salient features of smart grids is the replacement of conventional analog mechanical meters by digital meters, usually called "smart meters". In addition to recording users' energy usage, due to their communication capability, smart meters can provide a two-way communication path between UCs and power users, which can facilitate efficient power system control and monitoring. However, compared to mechanical meters which can only be physically tampered, smart meters are vulnerable to more types of attacks (e.g., network attack), which may make energy theft easier to commit and hence an even more serious problem in smart grids.

Some research has been conducted to detect energy theft in traditional power grids. Nizar et al. [38] employ a data mining technique known as Extreme Learning Machine (ELM) to classify users' electricity consumption patterns or load-profiles. By comparing the results to a database of users' load profiles, the proposed algorithm yields a list of users who could be stealing energy, which we call “energy thieves”. Nagi et al. [34] propose a similar approach but choose to use genetic algorithms and Support Vector Machine (SVM) instead of ELM. Depuru et al. [14] develop another data mining based scheme utilizing SVM as well. Unfortunately, these techniques cannot sort out the energy thieves with absolute certainty. In contrast, Bandim et al. [5] propose a central observer to measure the total energy consumption of a small number of users, and are able to identify all the energy thieves by comparing the total energy consumption with the reported energy consumption from the users.

Notice that in all the above works, the UC has to know some of users’ private information, e.g., users’ load profiles or meter readings at certain times, in order to find the energy thieves. However, the disclosure of such information would violate users’ privacy and raise concerns about privacy, safety, etc. In particular, users’ private information may be sold to interested third-parties. Insurance companies may buy load-profiles from the UC to make premium adjustments on the users’ policies. For example, they could find electricity consumption patterns that increase the risk of fire in a property and increase insurance premiums accordingly. Marketing companies may also be interested in this data to identify potential customers. Moreover, criminals may utilize such private information to commit crimes. For instance, robbers may analyze the energy consumption pattern of potential victims to deduce their daily behavior. They can even know if a robbery alarm has been set at their target location [43]. Many researchers, such as Quinn [42], have realized how high resolution electricity usage information can be used to reconstruct many intimate details of a consumer's daily life and invade his/her privacy, and thus call for state legislators and public utility commissions to address this new privacy threat [35].

Unfortunately, there is currently a lack of research on privacy-preserving energy theft detection in smart grids. Li et al. [29] design a privacy-preserving aggregation protocol to collect the total energy consumption of a group of users at a distribution station in smart grids, which shares a similar idea to those works like [21] on privacy-preserving data aggregation in wireless sensor networks. However, such algorithms cannot be used to detect energy theft in smart grids. To the best of our knowledge, we are the first to investigate the energy theft detection problem considering users' privacy issues.

In particular, intuitively and as in previous works, we need to know about a user's electric power consumption in order to tell whether he/she is committing fraud or not, which, however, results in the reveal of the user's privacy. Therefore, energy theft detection and users' privacy seem to be two conflicting problems. How to detect energy theft while preserving users' privacy is a challenging problem. In this paper, utilizing peer-to-peer (P2P) computing [46], we propose three distributed algorithms to solve a linear system of equations (LSE) for the users' "*honesty coefficients*". If a user's honesty coefficient is equal to 1, this user is honest. Otherwise, if the honesty coefficient is larger than 1, then this user has reported less consumed energy and hence is committing fraud. The users' privacy can be preserved because they do not need to disclose any of their energy consumption data to others.

More specifically, we propose to take advantage of distributed LU and QR decompositions to solve our LSE. Although some distributed algorithms for LU or QR decomposition [39] have been proposed in the literature, e.g., [1,18,30,57], they cannot preserve each node's private information. In this paper, we first develop a distributed privacy-preserving energy theft detection algorithm leveraging LU decomposition, called LUD. We find that LUD can successfully identify all the energy thieves in a small size network but may be unstable in large networks[1]. Then, we design another algorithm based on LU decomposition with partial pivoting, called LUDP, which can find all the energy thieves even in large-size networks. We also propose a third algorithm by QR decomposition, called QRD, which also works well in large-size networks. Moreover, the LUD, LUDP, and QRD algorithms are proposed in the case that users commit energy theft at a constant rate, i.e., with constant honesty coefficients. We further propose adaptive LUD/LUDP/QRD algorithms to account for the scenarios where the users have variable honesty coefficients.

In addition, after presenting the proposed algorithms, we analyze the computational and communication complexities of the two stable algorithms, i.e., LUDP and QRD. We find that LUDP algorithm has a computational complexity of $\Theta(2n^3/3)$ and a communication complexity of $\Theta(2n^3/3)$, and the QRD algorithm has a computational complexity of $\Theta(2n^3)$ and a communication complexity of $\Theta(5n^3/6)$. In other words, the QRD algorithm has higher computational complexity and higher communication complexity compared with the LUDP algorithm.

The rest of this paper is organized as follows. Section 4.2 introduces network model. Section 4.3 presents the linear system of equations for energy theft detection. Section 4.4 details the proposed distributed algorithms for solving the LSE. Computational and

[1]This is due to the rounding errors in LU decomposition.

communication complexity analysis is provided in Section 4.5. Simulation results are shown in Section 4.6. Finally, we conclude this paper in Section 4.7.

4.2 Network Model

In this section, we first present the network architecture considered in this paper, and then briefly introduce the possible attacks on smart meters (SMs) by energy thieves, and possible implementations of the proposed algorithms on the SMs.

4.2.1 Network Architecture

In the smart grid, communications and electricity networks overlay each other. Utility companies (UCs) deploy control centers (CCs) to monitor their distribution stations (DSs) and distribution networks, and deploy SMs at users's premises to measure their individual energy consumption. Since a CC is usually physically far away from users, a communication entity that can facilitate the communication between users and the CC is necessary. To this end, an access point, called "*the collector*", is placed at each of the serviced areas. One SM is installed at each collector to measure the total energy consumed by the serviced area.

A typical network architecture is depicted in Figure 4.1. In a serviced area, the users' SMs together with the collector form a Field Area Network (FAN). The communications among SMs and between SMs and the collector are carried out wirelessly due to SMs' communication capability, while the communications among the CC, the DS, and the collector are conducted via wired medium.

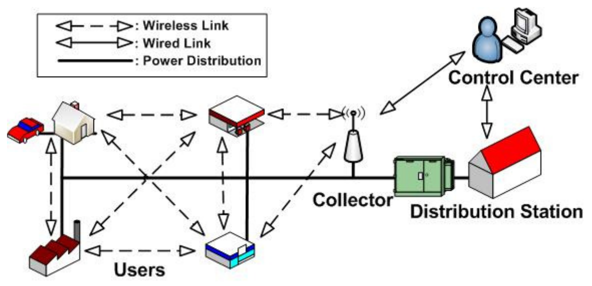


Figure 4.1

A typical architecture of Field Area Network (FAN).

4.2.2 Attacks on Smart Meters

Smart meters can provide the users with a plethora of unique features. For example, users can be provided with real time electricity pricing and thus determine when to turn on/off some of their electrical devices. Smart meters can also send incentive-based load reduction signals to users so that they can be compensated for their efforts to save energy. However, compared to mechanical meters which can only be physically tampered, smart meters are vulnerable to more types of attacks, which may make energy theft easier to commit and hence an even more serious problem in smart grids.

4.2.2.1 Physical Attack

Conventional mechanical meters and SMs are both vulnerable to this type of attack. It refers to the scenarios where illegal users physically modify their meters to record wrong values that will lower their electric bills. Physical attack to electricity meters includes meter reversing, tampering with strong magnets, pressure coil damaging, supply voltage regulation, and even disconnecting the meters. The readers are referred to [13] for a more extensive description on physical attacks.

One way to detect physical attacks is to visually check the meter for any broken seals or other signs of damage. However, this detection method is both resource and time consuming because employees from the UC have to visit the users' premises to verify the meters' integrity. Moreover, signs of damage may not be obvious and seals may be replaced.

4.2.2.2 Network Attack

An illegal user can operate a malicious node to perform network attack. For example, an illegal user may impersonate his/her own SM and make it record lower power consumption. Network attack may be easier to launch and more difficult to detect.

In addition to attacking the smart meter, a user may also get some energy that is not being measured, e.g., through a conductor that bypasses the meter. In this case, the smart meter does not correctly measure the energy consumption of the user and hence can also be considered as being attacked. The proposed algorithms can address all these problems.

4.2.3 Possible Implementation of the Proposed Algorithms

The proposed algorithms can be implemented in the firmware of the smart meters. Many mechanisms have been proposed to protect the firmware of embedded systems, such as passwords, centralized intrusion detection, local intrusion detection, and intrusion self-reporting. For example, LeMay et al. [27] develop a remote attestation mechanism that allows centralized intrusion detection of all SMs in a neighborhood. If any intrusion to the firmware is detected by the UC, the suspect SMs cannot be trusted and must be inspected.

Consequently, SMs can be trusted in correctly executing the proposed privacy preserving energy theft detection algorithms[2].

4.3 A Linear System of Equations for Energy Theft Detection

In this section, we present a mathematical model for energy theft detection. As mentioned before, we assume that an SM is installed at the collector such that the collector can know the total power consumption of the users in the service area. We also assume that the UC installs an SM at each of the users' premises, which is capable of recording energy consumption at any time instant.

Consider a FAN of n users. We define a *sampling period* denoted by SP. Then, after every sampling period, all the $n + 1$ SMs will record their energy consumption in the past sampling period. We denote such energy consumption recorded by user j ($1 \leq j \leq n$) and by the collector at time t_i, by $p_{t_i,j}$ and $\overline{\mathcal{P}}_{t_i}$, respectively. We further define an *honesty coefficient*, denoted by k_j where $k_j > 0$, for each user j. Thus, $k_j \cdot p_{t_i,j}$ gives the real energy consumption of user j from time instant $t_i - SP$ to time instant t_i. Since the sum of all the recorded energy consumption at time t_i must be equal to the total energy consumption of the neighborhood measured at the collector at time t_i, we have

$$k_1 p_{t_i,1} + k_2 p_{t_i,2} + ... + k_n p_{t_i,n} = \overline{\mathcal{P}}_{t_i} \tag{4.1}$$

Our objective is to find all the k_j's. Obviously, 1) if $k_j = 1$, then user j is honest and did not steal energy; 2) if $k_j > 1$, then user j recorded less energy than what he/she

[2]Note that although we assume a secure firmware for SMs, the upper layer software for SMs can still be compromised.

consumes and hence is an energy thief; and 3) if $0 < k_j < 1$, then user j recorded more than what he/she consumes, which means that his/her smart meter may be malfunctioning.

In particular, with n linear equations, we can have a linear system of equations (LSE) as follows:

$$\begin{aligned} k_1 p_{t_1,1} + k_2 p_{t_1,2} + ... + k_n p_{t_1,n} &= \overline{\mathcal{P}}_{t_1} \\ &\vdots \\ k_1 p_{t_n,1} + k_2 p_{t_n,2} + ... + k_n p_{t_n,n} &= \overline{\mathcal{P}}_{t_n} \end{aligned} \tag{4.2}$$

which can also be formulated in matrix form:

$$\boldsymbol{P}\boldsymbol{k} = \overline{\boldsymbol{\mathcal{P}}}. \tag{4.3}$$

The jth column of $\boldsymbol{P}$ represents the data recorded and stored by user j or SM_j, while the ith row of $\boldsymbol{P}$ represents the data recorded by all the users at t_i. The collector can choose n time instances when $\overline{\mathcal{P}}_{t_i}$'s all have different values. In this case, it is highly likely that the LSE is independent and the rows of $\boldsymbol{P}$ are independent as well, especially when n is large[3]. Thus, the above LSE only has a single unique solution, i.e., the feasible solution $k_j = \overline{p}_{t_i,j}/p_{t_i,j}$ where $\overline{p}_{t_i,j}$ is the real energy consumption of user j from time instant $t_i - SP$ to time instant t_i.

Note that our model does not take into account power dissipation, or technical losses (TLs), in the power system, which are mainly caused by the intrinsic inefficiencies in transformers and low voltage power lines. However, TLs can be calculated without using con-

[3]Besides, note that usually there are always some appliances running at users' premises, such as refrigerators and air conditioners, whose working powers are in practice dynamic with some fluctuations instead of constants.

sumers' power measurements. For example, Oliveira et al. [40] describe how to calculate TLs using measurements at the distribution station and the knowledge of the distribution network which does not need to compromise users' privacy. Thus, once the technical losses are calculated by the collector, the collector can adjust the model by subtracting the TLs from vector $\overline{\boldsymbol{\mathcal{P}}}$.

Besides, note that finding the honesty coefficient vector, $\boldsymbol{k}$, is delay tolerant. In other words, $\boldsymbol{k}$, is not required to be found and transmitted to the collector in real time. This gives priority to other real time traffic in the FAN, such as electricity pricing, incentive-based load reduction signals, and emergency load reduction signals.

4.4 Finding Honesty Coefficients by P2P Computing

In what follows, based on P2P computing (or distributed/collaborative computing), we propose three algorithms that can solve the linear system of equations in (4.3) while preserving the users' privacy. The challenge is that each smart meter SM_j needs to find its own honesty coefficient k_j without knowing any of the other smart meters' recorded energy consumption data $p_{t_i,l}$'s, where $1 \leq i \leq n$ and $j \neq l$.

Specifically, we first develop an LU decomposition based approach, called LUD, to detect the energy thieves while preserving users' privacy. We notice that in its original form, LUD may not be numerically stable in large size networks. The reason is that the inaccuracies involved when using finite resolution numbers may lead to solutions with significant errors when n is large. Therefore, after proposing the LUD algorithm, we design another algorithm to achieve numerical stability by exchanging rows of the matrix $\boldsymbol{P}$ during LU decomposition, i.e., LUD with partial pivoting (LUDP). After that, we also propose a QR Decomposition based algorithm, called QRD, to perform stable P2P computing in large-scale networks to find the energy thieves.

In the following, we detail these three algorithms, respectively, when the honesty coefficient vector, $\boldsymbol{k}$, is a constant, and then discuss the cases where $\boldsymbol{k}$ changes with time.

4.4.1 The LUD Algorithm

We first describe the LUD algorithm as follows, which is based on distributed LU decomposition. The LU decomposition is to factorize the power consumption data matrix $\boldsymbol{P}$ into two triangular matrices: a lower triangular matrix $\boldsymbol{L}$ and an upper triangular matrix $\boldsymbol{U}$, i.e., $\boldsymbol{P} = \boldsymbol{L}\boldsymbol{U}$.

The elements of upper triangular matrix $\boldsymbol{U}$ can be calculated as follows:

$$u_{i,j} = 0, \quad i > j \tag{4.4}$$

$$u_{1,j} = p_{t_{1,j}}, \quad j = 1, 2, ..., n \tag{4.5}$$

$$u_{r,j} = p_{t_{r,j}} - \sum_{k=1}^{r-1} l_{r,k} u_{k,j}, \qquad r = 2, ..., n, j = r, ..., n \tag{4.6}$$

where $p_{t_{i,j}}$ is the ith element of column j in matrix $\boldsymbol{P}$. Besides, the elements of lower triangular matrix $\boldsymbol{L}$ can be obtained by

$$l_{i,j} = 0, \quad i < j \tag{4.7}$$

$$l_{i,1} = \frac{p_{t_{i,1}}}{p_{t_{1,1}}}, \quad i = 1, 2, ..., n \tag{4.8}$$

$$l_{i,q} = \frac{p_{t_{i,q}} - \sum_{k=1}^{q-1} l_{i,k} u_{k,q}}{u_{q,q}}, \quad q = 2, ..., n, i = q, .., n \tag{4.9}$$

Note that the diagonal elements of $\boldsymbol{L}$ are equal to 1. This guarantees that the decomposition of $\boldsymbol{P}$ is unique.

After matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ are collaboratively computed, the following system can be solved:

$$\boldsymbol{L}\boldsymbol{y} = \overline{\boldsymbol{\mathcal{P}}}, \tag{4.10}$$

$$\boldsymbol{U}\boldsymbol{k} = \boldsymbol{y}. \tag{4.11}$$

In particular, to solve for $\boldsymbol{y}$, each SM_{j-1} will calculate y_j as follows, i.e., $y_1 = \overline{\mathcal{P}}_{t_1}$, and

$$y_j = \overline{\mathcal{P}}_{t_j} - \sum_{q=1}^{j-1} l_{j,q} y_q, \quad j = 2, ..., n. \tag{4.12}$$

The required values for this computation are the elements of $\boldsymbol{y}$ with index less than j and row j of $\boldsymbol{L}$. Finally, Each SM_j solves for k_j using backward substitution, i.e., $k_n = \frac{y_n}{u_{n,n}}$, and

$$k_j = \frac{y_j - \sum_{p=j+1}^{n} u_{j,p} k_p}{u_{j,j}} \quad j = 1, ..., n-1. \tag{4.13}$$

Therefore, our LUD algorithm is composed of two parts: Distributed LU Decomposition and Backward Substitution, which are detailed by Procedure 1 in Fig. 4.2 and Procedure 2 in Fig. 4.3, respectively. Besides, before the algorithm can begin, the collector must number all the SMs from 1 to n, and the index number of any SM is only known to the SM itself and the collector. The collector also transmits $\overline{\mathcal{P}}_{t_{j+1}}$ to each SM_j to allow the SMs to collaboratively compute $\boldsymbol{L}$, $\boldsymbol{U}$, and $\boldsymbol{y}$. We denote the smart meter at the collector as SM_0. All SMs start running Procedure 1 in Fig. 4.2 when the collector requests them by sending a control message.

Specifically, SM_0 first calculates $y_1 = \overline{\mathcal{P}}_{t_1}$, and then transmits it to SM_1. SM_0 does not need to compute any element of $\boldsymbol{L}$ or $\boldsymbol{U}$. After SM_1 receives y_1, it computes column 1 of $\boldsymbol{U}$, column 1 of $\boldsymbol{L}$, and y_2. Then, SM_1 transmits column 1 of $\boldsymbol{L}$, y_1, and y_2 to SM_2. SM_j ($1 < j < n$), receives y_1 through y_j and columns 1 through $j-1$ of $\boldsymbol{L}$ from SM_{j-1}, based on which it calculates column j of $\boldsymbol{U}$, column j of $\boldsymbol{L}$, and y_{j+1}. After that, SM_j transmits columns 1 through j of $\boldsymbol{L}$ and y_1 through y_{j+1} to SM_{j+1}. Finally, SM_n receives y_n and columns 1 through $n-1$ from SM_{n-1}, calculates column n of $\boldsymbol{U}$ and column n of $\boldsymbol{L}$, and notifies the collector that the Back Substitution procedure, i.e., Procedure 2 in Fig. 4.3, can start. Notice that each SM_j ($j > 0$) is responsible for computing column j of $\boldsymbol{U}$, column j of $\boldsymbol{L}$, and y_{j+1} ($1 \leq j < n$).

After Procedure 1 in Fig. 4.2 ends, SM_j ($1 \leq j \leq n$) has obtained the jth column of $\boldsymbol{U}$ and y_j. The collector then instructs all the smart meters to run Procedure 2 in Fig. 4.3 to solve for their own honesty coefficients according to (4.12), starting from SM_n. In particular, SM_n transmits $u_{n-1,n}k_n$, which is needed by SM_{n-1} to solve for k_{n-1}, along with $u_{n-2,n}k_n$, ..., $u_{1,n}k_n$, needed by SM_{n-2}, ..., SM_1, respectively, to SM_{n-1}. Similarly, SM_j ($1 < j < n$) receives $\sum_{q=j+1}^{n} u_{j,q}k_q$ from SM_{j+1} and solves for k_j, and then transmits $\sum_{q=j}^{n} u_{j-1,q}k_q$ along with $u_{1,j}k_j$, ..., $u_{j-2,j}k_j$, $u_{1,j+1}k_{j+1}$, ..., $u_{j-2,j+1}k_{j+1}$, ..., $u_{1,n}k_n ... u_{j-2,n}k_n$ to SM_{j-1}. Finally, SM_1 receives $\sum_{q=2}^{n} u_{1,q}k_q$ from SM_2 and solves for k_1. Moreover, after obtaining its honesty coefficient, each smart meter SM_j encrypts k_j using the collector's public key, resulting in $E(k_j)$, and then transmits $E(k_j)$ to the collector. When all the $E(k_j)$'s have been reported to the collector, the LSE can be successfully solved, and hence the collector can decrypt all the elements of $\boldsymbol{k}$ and identify all the fraudulent users.

Notice that in LUD, the collector does not know $\boldsymbol{L}$ or $\boldsymbol{U}$, and hence cannot recover $\boldsymbol{P}$. Moreover, from equation (4.13), we can observe that SM_j will need all the elements of $\boldsymbol{U}$ in row j and $k_n, k_{n-1}, ..., k_{j+1}$ in order to compute k_j. If such data are transmitted to SM_j separately, an eavesdropper would be able to figure out all the elements of $\boldsymbol{U}$, except those on the diagonal, by eavesdropping on all the transmissions in the network. Since an eavesdropper is able to obtain $\boldsymbol{L}$ by eavesdropping, too, it can figure out some elements of $\boldsymbol{P}$ (e.g., all the elements above the diagonal of $\boldsymbol{P}$). To prevent this from happening, as mentioned above and shown by Procedure 2 in Fig. 4.3, we transmit the multiplication of an element of $\boldsymbol{U}$ and the corresponding honesty coefficient instead. Notice that in this case an eavesdropper may be able to guess the power consumption of certain honest users (i.e., those whose honesty coefficients are equal to 1) at certain times by assuming $\boldsymbol{k} = \boldsymbol{1}$. However, even so since the eavesdropper does not know the mapping between smart meter indexes and users (only the collector knows), it cannot really know any user's power consumption data. Besides, the eavesdropper does not know which users are honest anyway. In addition, to defend against the case that the collector has the capability of eavesdropping on all the transmissions in the network, we can just enable each neighboring two smart meters, i.e., SM_j and SM_{j+1} where $1 \leq j \leq n-1$, to establish a symmetric security key on their own to encrypt the data transmitted in Procedure 2 in Fig. 4.3 (line 7 and line 8). In so doing, no user's private data will be revealed to or recovered by anyone else.

4.4.2 The LUDP Algorithm

As mentioned before, LUD may not be numerically stable when n is large. Here, we propose another algorithm, i.e., LUD with partial pivoting (LUDP), to address the stability problem. Partial pivoting is to interchange rows of the matrix $\boldsymbol{P}$ in order to place the element that has the greatest absolute value in each column in the diagonal position of the matrix. Thus, LUDP decomposition has the following form, $\boldsymbol{EP} = \boldsymbol{LU}$, where $\boldsymbol{E}$ is the permutation matrix.

The LUDP algorithm consists of three procedures: LU Decomposition with Partial Pivoting, Forward Substitution, and Backward Substitution. Procedure 3 in Fig. 4.4 shows how LU decomposition with partial pivoting works. Specifically, we first let $\boldsymbol{U} = \boldsymbol{P}$. Then, SM_1 finds the maximum element in column 1 of $\boldsymbol{U}$, lets the pivot index of column 1 be the row this element is in, interchanges this element with the element in row 1 if it is not, and updates the first column of $\boldsymbol{U}$. SM_1 also computes the first column of $\boldsymbol{L}$, and transmits it together with the pivot index of column 1 to SM_2. Note that in LUDP, we compute $\boldsymbol{U}$ and $\boldsymbol{L}$ in a different way from that in LUD, as shown in Line 8 and Line 17 of Procedure 3, respectively, which now allows partial pivoting [39]. After receiving the data from SM_1, SM_2 repeats SM_1's row interchange, i.e., interchanging the element in column 2, SM_1's pivot index row of $\boldsymbol{U}$ with the element in column 2, row 1 of $\boldsymbol{U}$. Then, SM_2 performs its own row interchange, which is to interchange the maximum element in column 2 of $\boldsymbol{U}$ with the second element in column 2 of $\boldsymbol{U}$, lets the pivot index of column 2 be the row this maximum element was in, and updates the second column of $\boldsymbol{U}$. After that,

SM_2 computes the second column of $\boldsymbol{L}$, and transmits the first two columns of $\boldsymbol{L}$ along with the pivot index of SM_1 and of SM_2 to SM_3. Finally, SM_n receives columns 1 to $n-1$ of $\boldsymbol{L}$ and all the previous nodes' pivot indexes from SM_{n-1}, repeats all the previous row interchanges, performs its own row interchange, and calculates column n of $\boldsymbol{U}$. Note that $l_{j,j} = 1$ for $1 \leq j \leq n$.

Moreover, due to (4.3), we need make the same row interchanges for $\overline{\mathcal{P}}$ as those for $\boldsymbol{P}$. Thus, we let SM_n send all the n pivot row indexes to the collector SM_0, which then performs the same row interchanges for $\overline{\mathcal{P}}$. Now we can solve for $\boldsymbol{y}$ and $\boldsymbol{k}$ according to (4.10) and (4.11), respectively. In particular, since $\boldsymbol{y}$ is computed according to (4.12), $\boldsymbol{y}$ can only be computed after Procedure 3 in Fig. 4.4 is finished, which is different from that in LUD where $\boldsymbol{y}$ can be computed at the same time as $\boldsymbol{L}$ and $\boldsymbol{U}$. Therefore, we propose the Forward Substitution procedure as shown in Procedure 4 in Fig. 4.5, to enable the smart meters to solve for $\boldsymbol{y}$ in a distributed way. Forward Substitution calculates y_j according to (4.12), and works similar to Backward Substitution except that it starts from SM_1. At last, after $\boldsymbol{y}$ is available, Back Substitution as described by Procedure 2 in Fig. 4.3 can be used to solve for $\boldsymbol{k}$.

Furthermore, the same as that in LUD, we can enable each neighboring two smart meters, i.e., SM_j and SM_{j+1} where $1 \leq j \leq n-1$, to establish a symmetric security key on their own and encrypt the data transmitted in Procedure 2 in Fig. 4.3 (line 7 and line 8) to protect users' privacy.

Compared to the LUD algorithm, LUDP takes more time to complete. This is because in LUDP the forward substitution to calculate $\boldsymbol{y}$ can only be carried out after $\boldsymbol{L}$ and $\boldsymbol{U}$ have been obtained, while in LUD, $\boldsymbol{y}$, $\boldsymbol{L}$, and $\boldsymbol{U}$ can be obtained at the same time. On the other hand, LUDP is numerically stable while LUD is not.

4.4.3 The QRD Algorithm

Here, we present another privacy-preserving energy theft detection algorithm, called QRD. In particular, by QR decomposition, matrix $\boldsymbol{P}$ can be decomposed into an orthogonal matrix $\boldsymbol{Q}$ and an upper triangular matrix $\boldsymbol{R}$, i.e., $\boldsymbol{P} = \boldsymbol{QR}$, where $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^{T}$. Thus, we have $\boldsymbol{Pk} = \boldsymbol{QRk} = \overline{\boldsymbol{\mathcal{P}}}$, which yields a new system

$$\boldsymbol{Rk} = \boldsymbol{Q}^{T}\overline{\boldsymbol{\mathcal{P}}}. \tag{4.14}$$

The basic idea is to utilize distributed QR decomposition to enable each smart meter to compute its own honesty coefficient without using other smart meters' energy consumption data.

We first present how to determine $\boldsymbol{Q}$ and $\boldsymbol{R}$ in the following. In particular, $\boldsymbol{Q}^{T}$ is formed as the product of $\frac{n(n-1)}{2}$ plane rotation matrices as follows:

$$\boldsymbol{Q}^{T} = \boldsymbol{G}_{n,n-1}(\boldsymbol{G}_{n-1,n-2}\boldsymbol{G}_{n,n-2}) \cdots (\boldsymbol{G}_{2,1} \cdots \boldsymbol{G}_{n,1}). \tag{4.15}$$

Let $\hat{\boldsymbol{P}}_{1,0} = \boldsymbol{P}$ and

$$\hat{\boldsymbol{P}}_{i,j} = (\boldsymbol{G}_{i,j} \cdots \boldsymbol{G}_{n,j}) \cdots (\boldsymbol{G}_{2,1} \cdots \boldsymbol{G}_{n,1})\hat{\boldsymbol{P}}_{1,0}. \tag{4.16}$$

Then, $\hat{\boldsymbol{P}}_{i,j} = \boldsymbol{G}_{i,j}\hat{\boldsymbol{P}}_{i+1,j}$ when $i < n$ and $\hat{\boldsymbol{P}}_{i,j} = \boldsymbol{G}_{i,j}\hat{\boldsymbol{P}}_{j,j-1}$ when $i = n$. Besides, when $\boldsymbol{G}_{i,j}$ multiplies $\hat{\boldsymbol{P}}_{i+1,j}$ when $i < n$ (or $\hat{\boldsymbol{P}}_{j,j-1}$ when $i = n$) from the left, it zeros element

$\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}(i,j)$ when $i < n$ (or $\hat{\boldsymbol{P}}_{\boldsymbol{j,j-1}}(i,j)$ when $i = n$), modifies rows i and $i-1$ of $\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}$ (or $\hat{\boldsymbol{P}}_{\boldsymbol{j,j-1}}$), and preserves previously introduced zeros[4]. Finally, $\boldsymbol{Q}^{\boldsymbol{T}}\boldsymbol{P}$ reduces $\boldsymbol{P}$ into an upper triangular matrix $\boldsymbol{R}$, i.e., $\hat{\boldsymbol{P}}_{\boldsymbol{n,n-1}} = \boldsymbol{R}$.

The two most common methods to find plane rotation matrices ($\boldsymbol{G}_{\boldsymbol{i,j}}$'s) are Householder Rotations and Givens Rotations (GR). In this paper we adopt the GR approach. The non-zero elements of $\boldsymbol{G}_{\boldsymbol{i+1,j}}$ are

$$g_{qq} = 1, q \neq i, j \tag{4.17}$$

$$g_{i,i} = c_{i,j},\ g_{i+1,i+1} = c_{i,j},\ g_{i,i+1} = s_{i,j},\ g_{i+1,i} = -s_{i,j} \tag{4.18}$$

where $c_{i,j}$ and $s_{i,j}$ are calculated as

$$p^{'}_{i,j} = (p^2_{i,j} + p^2_{i+1,j})^{1/2},\ c_{i,j} = \frac{p_{i,j}}{p^{'}_{i,j}},\ s_{i,j} = \frac{p_{i+1,j}}{p^{'}_{i,j}}. \tag{4.19}$$

Note that for simplicity we use $p_{i,j}$ to denote the element in ith row and jth column in the previously rotated matrix, i.e., $\hat{\boldsymbol{P}}_{\boldsymbol{i+2,j}}$ when $i < n-1$ and $\hat{\boldsymbol{P}}_{\boldsymbol{j,j-1}}$ when $i = n-1$.

Besides, the elements of the matrix after rotation, i.e., $\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}$, are

$$\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}(i,r) = c_{i,j}p_{i,r} + s_{i,j}p_{i+1,r} \quad \text{for } r \geq j \tag{4.20}$$

$$\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}(i+1,j) = 0 \tag{4.21}$$

$$\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}(i+1,r) = -s_{i,j}p_{i,r} + c_{i,j}p_{i+1,r} \quad \text{for } r > j \tag{4.22}$$

We denote by $\mathbb{G}_{i,j}$ the set that contains $c_{i,j}$ and $s_{i,j}$, i.e., $\mathbb{G}_{i,j} = \{c_{i,j}, s_{i,j}\}$. From equation (4.19), we can observe that the values needed by SM_j to compute $\mathbb{G}_{i,j}$ reside in column j. This allows SM_j to find all its rotation matrices, i.e., $\boldsymbol{G}_{\boldsymbol{j+1,j}}$,..., $\boldsymbol{G}_{\boldsymbol{n,j}}$, only using its

[4]$\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}(i,j)$ denotes the element of matrix $\hat{\boldsymbol{P}}_{\boldsymbol{i+1,j}}$ in row i, column j.

locally stored and calculated data. Moreover, (4.20) shows that SM_r, when $r > j$, needs the set $\mathbb{G}_{i,j}$ from SM_j to update its own data, i.e., column r of the rotated matrix $\boldsymbol{\hat{P}_{i+1,j}}$. In addition, notice that each column j $(1 \leq j \leq n)$ has $n-j$ elements that need to be converted to zero in order to finally find $\boldsymbol{R}$ in (4.14). The set that contains all the $\mathbb{G}_{i,j}$'s which need to be calculated by SM_j, denoted by $\mathbb{B}_j$, is thus $\mathbb{B}_j = \{\mathbb{G}_{n-1,j}, \mathbb{G}_{n-2,j}, ..., \mathbb{G}_{j,j}\}$.

The QRD algorithm is composed of two procedures: Distributed QR Decomposition and Backward Substitution. Distributed QR Decomposition works as follows. SM_1 first generates $\boldsymbol{G_{n,1}} \cdots \boldsymbol{G_{2,1}}$ to zero $n-1$ elements in the first column of $\boldsymbol{P}$ and hence obtain the first column of $\boldsymbol{R}$. After that, it transmits $\mathbb{B}_1$ to SM_2. SM_2 uses $\mathbb{B}_1$ to update its energy consumption data, i.e,. the second column of $\boldsymbol{P}$, and generates $\boldsymbol{G_{n,2}} \cdots \boldsymbol{G_{3,2}}$ to find the second column of $\boldsymbol{R}$. After that, SM_2 transmits $\mathbb{B}_1$ and $\mathbb{B}_2$ to SM_3, and so on and so forth. Finally, SM_n receives $\mathbb{B}_1$, $\mathbb{B}_2$, ..., and $\mathbb{B}_{n-1}$ from SM_{n-1}, updates its own energy consumption data, finds the nth column of $\boldsymbol{R}$. SM_n then transmits $\mathbb{B}_1$, $\mathbb{B}_2$, ..., and $\mathbb{B}_n$ to the collector. Therefore, at the end of this procedure each smart meter SM_j can obtain the jth column of $\boldsymbol{R}$, and the collector can compute $\boldsymbol{Q^T}$ and hence $\boldsymbol{Q^T\overline{\mathcal{P}}}$ using (4.15) and (4.17). The procedure is explained in details in Procedure 5 in Fig. 4.6.

After Procedure 5 in Fig. 4.6, the collector will instruct the smart meters to run Backward Substitution to compute their honesty coefficients in a distributed way. In particular, according to (4.14), at SM_j $(1 \leq j \leq n)$ we have

$$r_{j,j}k_j + r_{j,j+1}k_{j+1} + ... + r_{j,n}k_n = \boldsymbol{Q^T\overline{\mathcal{P}}}(j) \tag{4.23}$$

where $r_{i,j}$ is the element in the ith row and the jth column of $\boldsymbol{R}$, and $\boldsymbol{Q^T}\overline{\boldsymbol{\mathcal{P}}}(j)$ is the jth element of $\boldsymbol{Q^T}\overline{\boldsymbol{\mathcal{P}}}$. So, once the collector receives all the sets $\mathbb{B}_j$'s from SM_n, it will compute $\boldsymbol{Q^T}\overline{\boldsymbol{\mathcal{P}}}$ and distribute the jth element to SM_j. SM_n can then obtain its honesty coefficient k_n. After that, SM_n transmits $r_{n-1,n}k_n$, which is needed by SM_{n-1} to solve for k_{n-1}, along with $r_{n-2,n}k_n$, ..., $r_{1,n}k_n$, needed by SM_{n-2}, ..., SM_1, respectively, to SM_{n-1}. Similarly, SM_j receives $\sum_{q=j+1}^{n} r_{j,q}k_q$ from SM_{j+1} and solves for k_j, and then transmits $\sum_{q=j}^{n} r_{j-1,q}k_q$ along with $r_{1,j}k_j$, ..., $r_{j-2,j}k_j$, $r_{1,j+1}k_{j+1}$, ..., $r_{j-2,j+1}k_{j+1}$, ..., $r_{1,n}k_n ... r_{j-2,n}k_n$ to SM_{j-1}. Finally, SM_1 receives $\sum_{q=2}^{n} r_{1,q}k_q$ from SM_2 and solves for k_1. Moreover, after obtaining its honesty coefficient, each smart meter SM_j encrypts k_j using the collector's public key, resulting in $E(k_j)$, and then transmits $E(k_j)$ to the collector. When all the $E(k_j)$'s have been reported to the the collector, the LSE can be successfully solved, and hence the collector can decrypt all the elements of $\boldsymbol{k}$ and identify all the fraudulent users. This procedure is detailed in Procedure 6 in Fig. 4.7.

Notice that in QRD, although the collector can recover $\boldsymbol{Q}$ by knowing the rotation matrices $\boldsymbol{G_{i,j}}$'s, it does not know $\boldsymbol{R}$ and hence cannot recover $\boldsymbol{P}$. Moreover, similar to that in LUD and LUDP, we can enable each neighboring two smart meters, i.e., SM_j and SM_{j+1} where $1 \leq j \leq n-1$, to establish a symmetric security key on their own and encrypt the data transmitted in Procedure 6 in Fig. 4.7 (line 7 and line 8) to protect users' privacy, if the collector can eavesdrop on all the transmissions in the network.

4.4.4 Variable Honesty Coefficients

In the above LUD, LUDP, and QRD algorithms, we have only considered that the honesty coefficient vector $\boldsymbol{k}$ is a constant[5]. However, when an illegal user commits energy theft, it is possible that the rate at which he/she steals energy is variable. In other words, an illegal user can alter the smart meter in such a way that it steals energy at different rates at different times. Unfortunately, if $\boldsymbol{k}$ changes in an LSE, the proposed algorithms may not work well. Next, we design adaptive algorithms to address this problem.

We notice that when $\boldsymbol{k}$ changes in an LSE, the LUD, LUDP, and QRD algorithms will result in an honesty coefficient vector, many of whose elements are not equal to 1. Thus, when the collector gets the honesty coefficient vector $\boldsymbol{k}$ and counts the number of elements that are not equal to 1, it can infer by statistics whether it is possible to have this many energy thieves in the network. If it is unlikely for this event to happen, the collector can reduce the sampling period SP and run the algorithms again, until the possibility of that event is high and $\boldsymbol{k}$ does not change any more.

We give a mathematical model as follows. Assume there are n users in a serviced area and each of them commits energy theft independently with the same probability p. Let X denote the total number of energy thieves in the neighborhood. Then, X is a random variable, which has a Binomial distribution. Thus, when the collector runs LUD/LUDP/QRD and obtains the honesty coefficient vector $\boldsymbol{k}$, it can find the number of elements that are not

[5]Note that we can enable the SMs to report to the collector if they are disconnected from the loads.

equal to 1, which we denote by Y. Then, the collector can calculate the probability that this event happens as follows:

$$P(X = Y) = \binom{n}{Y} p^Y (1-p)^{n-Y}. \tag{4.24}$$

In addition, in the case that each user j commits energy theft independently with a different probability p_j, X is also a random variable, but its expectation becomes $\mathbb{E}[X] = \sum_{j=1}^{n} p_j$. Recall the Chernoff bounds [9]:

- For any $\delta > 0$,

$$P(X > (1+\delta)\mathbb{E}[X]) < e^{-\mathbb{E}[X]f(\delta)} \tag{4.25}$$

where $f(\delta) = (1+\delta)\log(1+\delta) - \delta$.

- For any $0 < \delta < 1$,

$$P(X < (1-\delta)\mathbb{E}[X]) < e^{-\frac{1}{2}\delta^2\mathbb{E}[X]}. \tag{4.26}$$

Then, the collector can infer whether the obtained $\boldsymbol{k}$ is true or not by calculating

$$P(X \geq Y) < e^{-\mathbb{E}[X]f(\delta)} \text{ with } \delta = Y/\mathbb{E}[X] - 1 \tag{4.27}$$

when $Y > \mathbb{E}[X]$, and

$$P(X \leq Y) < e^{-\frac{1}{2}\delta^2\mathbb{E}[X]} \text{ with } \delta = 1 - Y/\mathbb{E}[X] \tag{4.28}$$

when $Y < E[X]$. Besides, when $Y = E[X]$, we set $P(X = Y) = 1$[6].

Thus, if the estimated probability P is lower than a threshold th, we reduce the sampling period SP by g ($g > 0$), which is a step variable, and run the LUD/LUDP/QRD

[6]As shown in Procedure 7 in Fig. 4.8, by setting $P(X = Y) = 1$ in this case, we will run the LUD/LUDP/QRD algorithm again with a reduced sampling period. If the obtained $\boldsymbol{k}$ does not change any more, we consider it is the true honesty coefficient vector we are looking for.

algorithm again to obtain another $\boldsymbol{k}$. This process repeats until P is no less than the threshold and the obtained $\boldsymbol{k}$ is the same as the previous one. By then we consider the $\boldsymbol{k}$ is true, i.e., the real honesty coefficient vector in the network.

We finally present the adaptive LUD/LUDP/QRD algorithm in Procedure 7 in Fig. 4.8, which can detect illegal users with variable honesty coefficients.

4.5 Computational and Communication Complexity Analysis

In this section we analyze the computational and communication complexities of LUDP and QRD, the two stable algorithms. We define the computational complexity as the number of elementary arithmetic operations (additions, subtractions, multiplications, divisions, and square roots), plus the number of comparisons and row exchanges needed to find vector $\boldsymbol{k}$. We define the communication complexity as the total traffic demand in the network, i.e., the total number of quantities that need to be transmitted in the network.

4.5.1 The LUDP Algorithm

4.5.1.1 Computational Complexity

To determine LUDP's computational cost, we need look into the operations in Procedure 3 in Fig. 4.4, Procedure 4 in Fig. 4.5, and Procedure 2 in Fig. 4.3 as follows.

In Procedure 3 in Fig. 4.4, lines 6, 8, 14, 15 and 17 conduct computations. Specifically, line 6 performs one row exchange and repeats $(j-1)$ times at SM_j, where $2 \leq j \leq n$ and n is the number of users in the network. Line 15 also performs one row exchange and repeats $(n-1)$ times. Thus, the total number of row exchanges in Procedure 3 is $\sum_{j=2}^{n}(j-1)+(n-1)=\frac{n^2+n-2}{2}$.

Line 8 performs two elementary operations and lies inside a nested "for" loop. To find the total number of times that line 8 is executed, we first consider the nested "for" loops only, then consider the number of times the procedure is executed. In particular, the inner "for" loop iterates $(n-f)$ times and the outer "for" loop iterates $(j-1)$ times. Therefore, we have that line 8 executes $\sum_{f=1}^{j-1}(n-f) = n(j-1) - \frac{j(j-1)}{2}$ times for each $2 \leq j \leq n$. Then, the total number of elementary operations contributed by this line is $2\sum_{j=2}^{n}\left(n(j-1) - \frac{j(j-1)}{2}\right) = \frac{2n^3-3n^2+n}{3}$.

Line 14 contributes one search for the highest absolute value among $(n-j+1)$ elements in column j, where $1 \leq j \leq n-1$. In the worst case scenario, each search needs $(n-j)$ comparisons to determine the pivot row. Thus, the total number of elementary operations by line 14 is $\sum_{j=1}^{n-1}(n-j) = \frac{n^2-n}{2}$.

In addition, line 17 performs one elementary operation and repeats $(n-j)$ times for $1 \leq j \leq n-1$. Therefore, the total number of elementary operations performed by line 17 is $\sum_{j=1}^{n-1}(n-j) = \frac{n^2-n}{2}$.

In Procedure 4 in Fig. 4.5, line 6 computes $(j-1)$ multiplications and as many additions or subtractions, which are the computations in lines 8 and 9 carried out at the previous node. Line 11 also conducts $j(n-j-1)$ multiplications for $1 \leq j \leq n-1$. Therefore, the total number of elementary arithmetic operations in the Forward Substitution procedure is given by $2\sum_{j=1}^{n}(j-1) = n^2 - n$.

The computational complexity of Procedure 2 in Fig. 4.3, i.e., Backward Substitution, is similar to that of Procedure 4 in Fig. 4.5, i.e., Forward Substitution, with the exception of n additional divisions. So Procedure 2's computational complexity is

$$2\sum_{j=1}^{n}(j-1)+n=n^2. \tag{4.29}$$

As a result, adding the above computational complexity results together, we can find that the total computational complexity of LUDP, denoted by PC_{LUDP}, is

$$PC_{LUDP}=\frac{4n^3+15n^2-7n-6}{6}. \tag{4.30}$$

4.5.1.2 Communication Complexity

The total communication complexity of LUDP is also determined by the traffic demand of Procedure 3 in Fig. 4.4, Procedure 4 in Fig. 4.5, and Procedure 2 in Fig. 4.3.

In Procedure 3 in Fig. 4.4, only lines 20 and 23 account for communications. According to line 20, SM_j transmits to SM_{j+1} the first j columns of $\boldsymbol{L}$ ($n-f+1$ elements in column f) and j pivot row indexes. Besides, in line 23, SM_n transmits all the n pivot row indices to SM_0, i.e., n quantities. Thus, the traffic demand of Procedure 3 is $\sum_{j=1}^{n-1}\left(\sum_{f=1}^{j}(n-f+1)+j\right)+n=\frac{4n^3+6n^2-2n}{12}$.

In Procedure 4 in Fig. 4.5, lines 3, 7, 10, 11, and 14 carry out transmissions. Lines 3 and 14 are executed only once and transmit one quantity each, while lines 7 and 10 repeat $(n-1)$ times, each of which transmits one quantity. Line 11 transmits $j(n-j-1)$ quantities for $1\le j\le n-1$. Consequently, the traffic demand of Procedure 4 is $\sum_{j=1}^{n-1}j(n-j-1)+2(n-1)+2=\frac{n^3-3n^2+14n}{6}$.

Similarly, in Procedure 2 in Fig. 4.3, lines 3, 7, and 8 carry out transmissions. Particularly, lines 3 and 7 transmit one quantity each and repeat n and $n-1$ times, respectively. Line 8 transmits $(j-2)(n-j+1)$ quantities for $2 \leq j \leq n$. Therefore, the traffic demand of Procedure 2 is

$$\sum_{j=2}^{n}(j-2)(n-j+1)+2n-1=\frac{n^3-3n^2+14n-6}{6}. \tag{4.31}$$

Thus, from the above results, we can find that the total communication complexity of LUDP, denoted by MC_{LUDP}, is

$$MC_{LUDP}=\frac{8n^3-6n^2+54n-12}{12}. \tag{4.32}$$

4.5.2 The QRD Algorithm

4.5.2.1 Computational Complexity

We need examine Procedure 5 in Fig. 4.6 and Procedure 6 in Fig. 4.7 to analyze QRD's computational complexity.

In Procedure 5 in Fig. 4.6, line 4 performs six elementary operations $(n-f)$ times for $1 \leq f \leq j-1$, where $2 \leq j \leq n$. Thus, the total number of elementary operations by line 4 is $6\sum_{j=2}^{n}\sum_{f=1}^{j}(n-f)=2n^3-8n+6$.

Besides, line 14 carries out six elementary operations and repeats $(n-j)$ times for $1 \leq j \leq n-1$. Therefore, the total number of elementary operations by line 14 is $6\sum_{j=1}^{n-1}(n-j)=3n^2-3n$.

Moreover, the computational complexity of Procedure 6 in Fig. 4.7 is the same as that of Procedure 2 in Fig. 4.7 shown in (4.29). As a result, from the above results and (4.29), we can have that the computational complexity of QRD, denoted by PC_{QRD}, is

$$PC_{QRD} = 2n^3 + 4n^2 - 11n + 6. \tag{4.33}$$

4.5.2.2 QRD Communication Complexity

The total communication complexity of QRD is also determined by the traffic demand of Procedure 5 in Fig. 4.6 and Procedure 6 in Fig. 4.7.

In Procedure 5 in Fig. 4.6 , lines 19 and 21 carry out transmissions of $\mathbb{B}_1 ... \mathbb{B}_j$ for $1 \leq j \leq n$. Since $\mathbb{B}_k$ $(1 \leq k \leq j)$ contains $n - k$ $\mathbb{G}_{p,q}$ sets, each of which contains two quantities, the traffic demand of Procedure 5 is $2 \sum_{j=1}^{n} \sum_{k=1}^{j} (n-k) = \frac{2n^3 - 2n}{3}$.

Moreover, the traffic demand of Procedure 6 in Fig. 4.7 is the same as that of Procedure 2 in Fig. 4.7 shown in (4.31). Consequently, the communication complexity of QRD, denoted by MC_{QRD}, is

$$MC_{QRD} = \frac{5n^3 - 3n^2 + 10n - 6}{6}. \tag{4.34}$$

4.6 Simulation Results

Here, we perform two series of simulations to evaluate the performance of our privacy-preserving energy theft detection algorithms LUD, LUDP, and QRD. In the first part, we assume that illegal users steal energy at a constant rate and thus have constant honesty coefficients. In the second part, we consider that illegal users have variable honesty coefficients. We conduct simulations in Matlab R2010a. The simulation results in the above two cases are presented in Section 4.6.1 and Section 4.6.2, respectively.

Besides, we generate users' power consumption data, $\boldsymbol{P}$, based on a set of data from [7] and [51]. These two studies conduct experiments in which both commercial and residential users are metered every hour and every half-hour, respectively. With these measurements, both studies provide typical daily user load profiles for different days of the week and different seasons of the year.

4.6.1 Constant Honesty Coefficients

We first perform simulations when illegal users steal energy at a constant rate. In other words, each illegal SM chooses a rate to steal energy and never changes this rate or stops stealing, thus having a constant honesty coefficient.

We evaluate the performance of LUD, LUDP, and QRD, when every user commits energy theft with a probability of 0.3 and there are totally 15, 30, and 50 users, respectively. Each energy thief chooses a honesty coefficient uniformly and randomly in [1.1, 10]. As shown in Fig. 4.9, the LUD algorithm can work well when there are 15 and 30 users in total. In particular, in Fig. 4.9(a) we can see that 6 users have an honesty coefficient larger than 1. It means that these 6 SMs only record a fraction of their consumed energy. Using these results, the collector can easily identify the energy thieves and how much less they have paid in their monthly bills. We can also observe that the legal users have an honesty coefficient equal to 1 and can be easily identified as well. Similar results are shown in Fig. 4.9(b) when there are 30 users. Besides, we can also find that LUDP and QRD can obtain the same results as LUD in these two cases. Moreover, the results of LUD, LUDP, and QRD when the number of users is 50 are presented in Fig. 4.10. In this case, the LUD algorithm is not stable. It finds 49 illegal users while in practice there are only 17 energy thefts. In contrast, the LUDP and QRD algorithms can successfully identify all the 17 illegal users.

4.6.2 Variable Honesty Coefficients

We then conduct simulations when illegal users steal energy at variable rates. We consider that each energy thief chooses a new honesty coefficient uniformly and randomly in [1.1, 10] each time after a certain period. we first consider that all the users commit energy theft with the same probability $p = 0.3$, and then consider that each user commits energy theft with a probability independently and randomly chosen between 0.3 and 0.7.

In particular, when all the users have the same cheating probability $p = 0.3$, we find that the adaptive LUD algorithm is not stable when there are more than 25 users in the network. The results are omitted due to space limitation. Besides, we show the results of the adaptive LUDP/QRD algorithm in Fig. 4.11, when the number of users is equal to 100, 200, and 300, respectively. We can see that all the energy thieves can be found. Moreover, when each user commits energy theft with a probability independently and randomly chosen between 0.3 and 0.7, we show the results of the adaptive LUDP/QRD algorithm in Fig. 4.12, in the cases that the number of users is equal to 100, 200, and 300, respectively. We can find that in these cases, the adaptive LUDP/QRD algorithm can also successfully and efficiently identify those fraudulent users.

4.7 Conclusion

In this paper, we have presented three P2P computing algorithms, i.e., LUD, LUDP, and QRD, which can identify the users who are committing energy theft in smart grids while preserve all users' privacy. The three algorithms are distributed algorithms and are based on LU or QR decomposition. We can observe that no private data from any user needs to be transmitted to other users or to the collector, which cannot be recovered either, thus preserving users' privacy. We have also analyzed the computational and communication complexities of the proposed algorithms, and find that QRD has higher computational complexity and higher communication complexity compared to LUDP. Moreover, extensive simulations have been conducted. The simulation results show that fraudulent users can be detected both when they commit energy theft at a constant rate, i.e., with constant honesty coefficients, and when they steal energy at variable rates, i.e., with variable honesty coefficients.

```
Input: $j \to SM_j$ , $\overline{\mathcal{P}}_{t_{j+1}} \to SM_j$
 1: if $j = 0$ or $SM_j$ receives packets from $SM_{j-1}$ then
 2:     if $j = 0$ then
 3:         Compute $y_1$ using (4.12)
 4:         Transmit $y_1$ only to $SM_1$
 5:     end if
 6:     if $1 \leq j \leq n-1$ then
 7:         for $q = 1$ to $j$ do
 8:             Compute $u_{q,j}$ using (4.4)
 9:         end for
10:         for $q = j+1$ to $n$ do
11:             Compute $l_{q,j}$ using equation (4.7)
12:         end for
13:         Compute $y_{j+1}$ using (4.12)
14:         Transmit columns $1, 2, ..., j$ of $\boldsymbol{L}$ and all known elements of $y_1, ..., y_{j+1}$
            only to $SM_{j+1}$
15:     end if
16:     if $j = n$ then
17:         Notify the collector that $\boldsymbol{L}$, $\boldsymbol{U}$, and $\boldsymbol{y}$ are available
18:     end if
19: end if
```

Figure 4.2

Procedure 1: Distributed LU Decomposition

```
1: if $j = n$ or $SM_j$ receives packet from $SM_{j+1}$ then
2:     Compute $k_j$ as described in (4.13) using $s_{j+1}$ if necessary
3:     Compute $E(k_j)$ and transmit $E(k_j)$ to the collector
4:     Compute $u_{q,j}k_j$ for $q = j-1, j-2, ..., 1$
5:     if $j \neq 1$ then
6:         Compute $s_j = \sum_{q=j}^{n} u_{j-1,q}k_q$
7:         Transmit $s_j$ to $SM_{j-1}$
8:         Transmit $u_{1,j}k_j$, ..., $u_{j-2,j}k_j$, $u_{1,j+1}k_{j+1}$, ..., $u_{j-2,j+1}k_{j+1}$, ..., $u_{1,n}k_n$, ...,
           $u_{j-2,n}k_n$ to $SM_{j-1}$
9:     end if
10: end if
```

Figure 4.3

Procedure 2: Backward Substitution

```
Input: $j \rightarrow SM_j$
 1: $\boldsymbol{U} = \boldsymbol{P}$
 2: if received packet from $SM_{j-1}$ or $j = 1$ then
 3:     if $j \neq 1$ then
 4:         Receive columns $1, 2, ..., j-1$ of $\boldsymbol{L}$ and pivot indexes
 5:         for $f = 1$ to $j-1$ do
 6:             Update column $j$ of $\boldsymbol{U}$ by interchanging the $j$th element of row $f$
                with the $j$the element of the pivot row of $SM_f$
 7:             for $r = f+1$ to $n$ do
 8:                 $u_{r,j} = u_{r,j} - l_{r,f}u_{f,j}$
 9:             end for
10:         end for
11:     end if
12:     Compute $l_{j,j} = 1$
13:     if $j \neq n$ then
14:         Determine pivot rows in column $j$ of $\boldsymbol{U}$
15:         Interchange the $j$th element of row $j$ with the $j$th element of the pivot
            row in $\boldsymbol{U}$ and $\boldsymbol{L}$
16:         for $r = j+1$ to $n$ do
17:             Compute $l_{r,j} = \frac{u_{r,j}}{u_{j,j}}$
18:             Compute $u_{r,j} = 0$
19:         end for
20:         Transmit columns $1, 2..., j$ of $\boldsymbol{L}$ and pivot row indexes to $SM_{j+1}$.
21:     else
22:         Notify the collector that $\boldsymbol{L}$ and $\boldsymbol{U}$ are available
23:         Transmit all the $n$ pivot row indexes to $SM_0$
24:     end if
25: end if
```

Figure 4.4

Procedure 3: LU Decomposition with Partial Pivoting

```
**Input:** $j \rightarrow SM_j$
1: if $j = 0$ or $SM_j$ receives packets from $SM_{j-1}$ then
2:     if $j = 0$ then
3:         Compute $y_1 = \overline{\mathcal{P}}_{t_1}$ and transmit $y_1$ to $SM_1$
4:     end if
5:     if $1 \leq j \leq n-1$ then
6:         Compute $y_{j+1}$ as described in (4.12) using $s_{j-1}$ if necessary
7:         Transmit $y_{j+1}$ to $SM_{j+1}$
8:         Compute $l_{q,j}y_j$ for $q = j+1, j+2, ..., n$
9:         Compute $s_j = \sum_{q=1}^{j} l_{j+1,q}y_q$
10:        Transmit $s_j$ to $SM_{j+1}$
11:        Transmit $l_{j+2,1}y_1$, ..., $l_{n,1}y_1$, ..., $l_{j+2,j}y_j$, ..., $l_{n,j}y_j$ to $SM_{j+1}$
12:    end if
13:    if $j = n$ then
14:        Notify the collector that $\boldsymbol{y}$ is available
15:    end if
16: end if
```

Figure 4.5

Procedure 4: Forward Substitution

```
Input: $j \rightarrow SM_j$
1:  if $j > 1$ and $SM_j$ receives $\mathbb{B}_1, \mathbb{B}_2 ... \mathbb{B}_{j-1}$ from $SM_{j-1}$ then
2:      for $f = 1$ to $j - 1$ do
3:          for $r = n - 1$ to $f$ do
4:              Update elements in column $j$ of $\boldsymbol{P}$ using $\mathbb{G}_{r,f}$ and $\mathbb{G}_{r+1,f}$ as described in (4.20).
5:          end for
6:      end for
7:      for $q = n$ to $j + 1$ do
8:          Compute $c_{q,j}$ and $s_{q,j}$ using (4.19) and store them
9:          Zero element $p_{q,j}$ using (4.20)
10:     end for
11: end if
12: if $j = 1$ then
13:     for $q = n$ to $j + 1$ do
14:         Compute $c_{q,j}$ and $s_{q,j}$ using (4.19) and store them
15:         Zero element $p_{q,j}$ using (4.20)
16:     end for
17: end if
18: if $j \neq n$ then
19:     Transmit $\mathbb{B}_1 ... \mathbb{B}_j$ to $SM_{j+1}$
20: else
21:     Transmit $\mathbb{B}_1 ... \mathbb{B}_j$ to the collector
22: end if
```

Figure 4.6

Procedure 5: Distributed QR Decomposition

```
1:  **if** $j = n$ or $SM_j$ receives packet from $SM_{j+1}$ **then**
2:      Compute $k_j$ as described in (4.23) using $s_{j+1}$ if necessary
3:      Compute $E(k_j)$ and transmit $E(k_j)$ to the collector
4:      Compute $r_{q,j}k_j$ for $q = j-1, j-2, ..., 1$
5:      **if** $j \neq 1$ **then**
6:          Compute $s_j = \sum_{q=j}^{n} r_{j-1,q}k_q$
7:          Transmit $s_j$ to $SM_{j-1}$
8:          Transmit $r_{1,j}k_j$, ..., $r_{j-2,j}k_j$, $r_{1,j+1}k_{j+1}$, ..., $r_{j-2,j+1}k_{j+1}$, ..., $r_{1,n}k_n$, ...,
            $r_{j-2,n}k_n$ to $SM_{j-1}$
9:      **end if**
10: **end if**
```

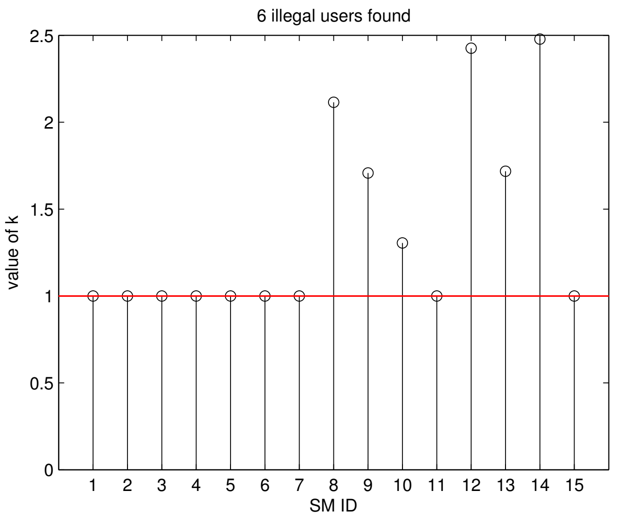
Figure 4.7

Procedure 6: Backward Substitution

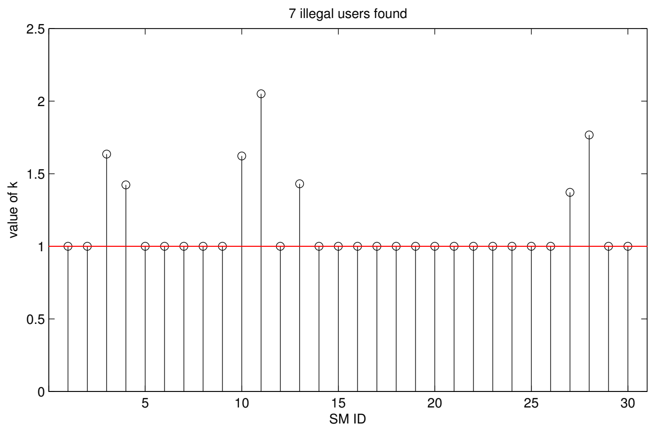
```
1: repeat
2:     The collector instructs all SMs to take $n$ samples with a initial sampling period $SP$
3:     Run the LUD/LUDP/QRD algorithm
4:     if The collector receives all elements in $\boldsymbol{k}$ then
5:         $Y =$ the number of elements in $\boldsymbol{k}$ unequal to 1, i.e., the number of illegal SMs
6:     end if
7:     The collector calculates the probability that there are $Y$ illegal users according to (4.24) or (4.27) or (4.28), denoted by $P$.
8:     if $P < th$ (a threshold) or $P = 1$ then
9:         $SP = SP - g$ ($g > 0$ is a step variable)
10:    end if
11: until $P \geq th$ and $\boldsymbol{k}$ does not change any more
```

Figure 4.8

Procedure 7: Adaptive LUD/LUDP/QRD Algorithm



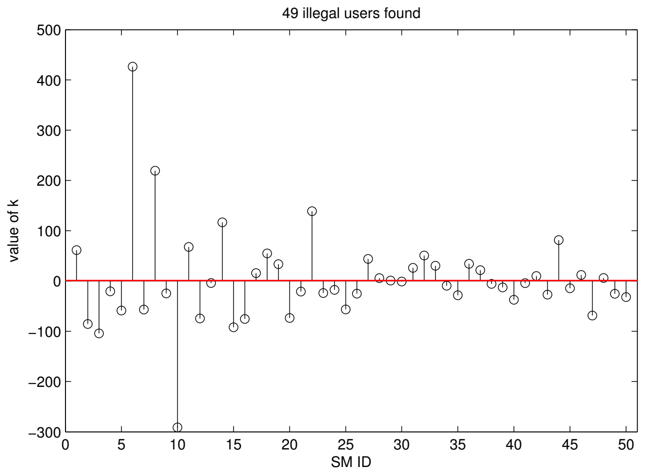
(a) 15 users



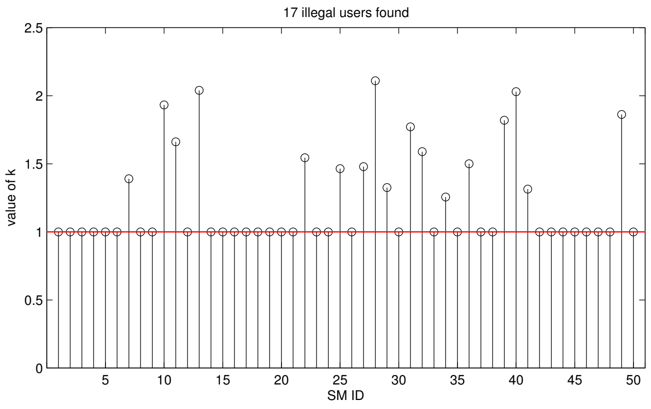
(b) 30 users

Figure 4.9

Elements of ***k*** obtained by the LUD algorithm.



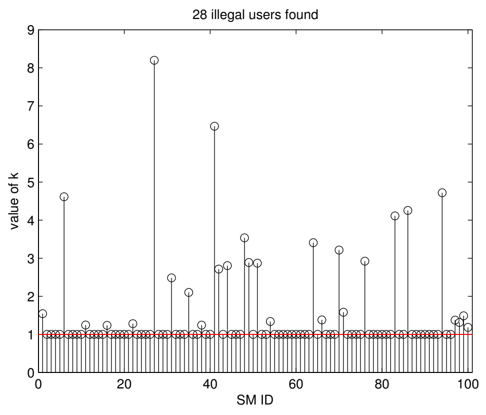
(a) LUD



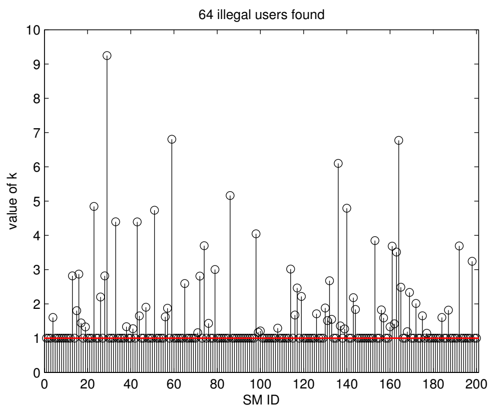
(b) LUDP and QRD

Figure 4.10

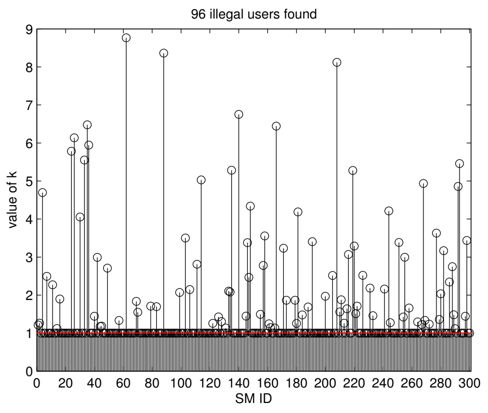
Elements of $\boldsymbol{k}$ obtained by the LUD, LUDP, and QRD algorithms in a network of 50 users.



(a) 100 users



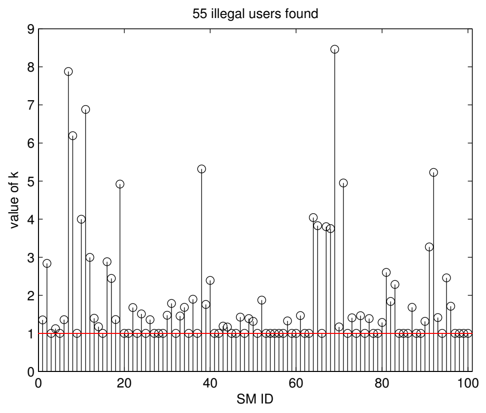
(b) 200 users



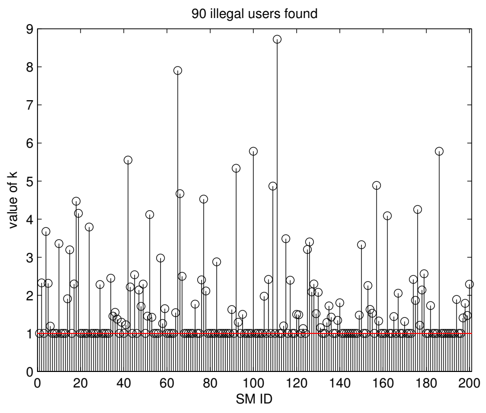
(c) 300 users

Figure 4.11

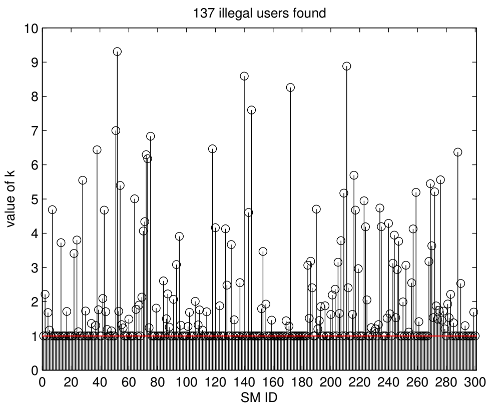
Elements of $\boldsymbol{k}$ obtained by the LUDP and QRD algorithms – constant cheating probability.



(a) 100 users



(b) 200 users



(c) 300 users

Figure 4.12

Elements of $\boldsymbol{k}$ obtained by the LUDP and QRD algorithms – variable cheating probability.

CHAPTER 5

CONCLUSIONS

This dissertation studies optimal energy management, and privacy-preserving energy theft detection in smart grids. In Chapter 2, we consider a third-party managing the energy consumption of a group of smart grid users, and formulate the load scheduling problem as a constrained multi-objective optimization problem. The first objective is to minimize the total energy consumption cost, while the second is to maximize its utility measured by a certain utility function. To solve the problem, we develop an evolutionary algorithm (EA), called LSEA, to retrieve a set of Pareto-optimal solutions and show the trade-offs between the energy consumption cost and the utility. To improve the algorithm efficiency, we design ϵ-LSEA, an ϵ-Pareto evolutionary algorithm that finds ϵ-Pareto fronts of the objective space. Extensive simulations were conducted to evaluate the performance of the proposed algorithms. We can observe that ϵ-LSEA is more efficient compared to LSEA.

In Chapter 3, we study the optimal energy management problem taking into consideration unpredictable load demands and distributed energy resources. Both delay intolerant (DI) and delay tolerant (DT) load demands were studied. We first formulated an optimization problem, which turned out to be a time-coupling problem and prohibitively expensive to solve. Then, we reformulated the problem using Lyapunov optimization theory and developed a dynamic energy management scheme that can dynamically solve the problem in each time slot based only on the current system state. Through mathematical analysis, we were able to obtain both a lower and an upper bound on the optimal result of the original optimization problem. Furthermore, in the case of both DI and DT load demands, we showed that DT load demands are guaranteed to be served within user-defined deadlines. Extensive simulations were conducted to validate the efficiency of the developed schemes.

In Chapter 4, we have presented three P2P computing algorithms, i.e., LUD, LUDP, and QRD, which can identify the users who are stealing energy in smart grids while preserving all users' privacy. The three algorithms are distributed algorithms and are based on LU or QR decomposition. We can observe that no private data from any user needs to be transmitted to other users or to the collector, which cannot be recovered either, thus preserving users' privacy. We have also analyzed the computational and communication complexities of the proposed algorithms, and find that QRD has higher computational complexity and higher communication complexity compared to LUDP. Moreover, extensive simulations have been conducted. The simulation results show that fraudulent users can be detected both when they commit energy theft at a constant rate, i.e., with constant honesty coefficients, and when they steal energy at variable rates, i.e., with variable honesty coefficients.

To conclude, we describe our plans for future work. In Chapters 2 and 3 , we observe that the energy consumption schedules found by our proposed algorithms may not be compatible with the underlying physical characteristics of the power system, e.g., a generator may be too slow to adjust its output to satisfy the schedule, or a power line may be overloaded during certain time slots. Hence, to have a holistic modeling of the energy management problem, we can consider the physical characteristics of the power system. Moreover, analyzing the energy theft detection algorithms presented in Chapter 4, we observe that the utility company needs an approximation of the network's thermal losses to adjust the linear system of equations, which may sometimes be infeasible in practice. Future work for energy theft detection includes taking into account the physical characteristics of the power system, i.e., voltages, currents, and impedances, to enable real-time calculation of thermal losses. Moreover, by considering stronger energy thieves, i.e., thieves that can completely compromise their smart meters, we can design more robust energy theft detection algorithms.

REFERENCES

[1] S. Abdelhak, A. Abdelgawad, S. Ghosh, and M. Bayoumi, "A complete scheme for distributed LU decomposition on wireless sensor networks," *IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, Seattle, Washington, USA, August 2010.

[2] T. Agarwal and S. Cui, "Noncooperative Games for Autonomous Consumer Load Balancing over Smart Grid," *Game Theory for Networks*, V. Krishnamurthy, Q. Zhao, M. Huang, and Y. Wen, eds., vol. 105 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Springer Berlin Heidelberg, 2012, pp. 163–175.

[3] R. B. Agrawal and K. Deb, "Simulated Binary Crossover for Continuous Search Space,", Convenor Technical Reports, Indian Institute of Technology, 1994.

[4] M. Albadi and E. El-Saadany, "Demand Response in Electricity Markets: An Overview," *Proceedings of the 2007 IEEE Power Engineering Society General Meeting*, Tampa, Fl, USA, June 2007.

[5] C. Bandim, J. Alves, A. Pinto, F. Souza, M. Loureiro, C. Magalhges, and F. Galvez-Durand, "Identification of energy theft and tampered meters using a central observer meter: a mathematical approach," *Proceedings of the Transmission and Distribution Conference and Exposition*, Dallas, Texas, USA, September 2003.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control, 2nd ed*, vol. 1 and 2., Athena Scientific, Belmont, MA, USA, 2007.

[7] *California Commercial End-Use Survey*, The California Energy Commission, 2006.

[8] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for solving Multi-Objective Problems*, Springer, 2007.

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms (2nd ed.)*, MIT Press, 2001.

[10] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization," *Parallel Problem Solving from Nature Conference*, Paris, France, September 2000, pp. 839–848.

[11] K. Deb, M. Mohan, and S. Mishra, "Evaluating the -Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions," *Evolutionary Computation*, vol. 13, no. 4, Winter 2005, pp. 501–525.

[12] K. Deb, Pratap, A., Agarwal, S., and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, April 2002, pp. 182–197.

[13] S. Depuru, L. Wang, and V. Devabhaktuni, "A conceptual design using harmonics to reduce pilfering of electricity," *Proceedings of the Power and Energy Society General Meeting*, Minneapolis, Minnesota, USA, July 2010.

[14] S. Depuru, L. Wang, and V. Devabhaktuni, "Support vector machine based data classification for detection of electricity theft," *Proceedings of the Power Systems Conference and Exposition (PSCE)*, Phoenix, Arizona, USA, March 2011.

[15] P. Du and N. Lu, "Appliance Commitment for Household Load Scheduling," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, 2012, pp. 411–4119.

[16] *Pot growers stealing $100M worth of power: B.C. Hydro*, Edmonton Journal, 2010.

[17] N. Gatsis and G. Giannakis, "Residential Load Control: Distributed Scheduling and Convergence With Lost AMI Messages," *IEEE Transactions on Smart Grid*, vol. 3, no. 2, 2012, pp. 770–786.

[18] G. A. Geistt and C. H. Romine, "LU Factorization Algorithms on Distributed-Memory Multiprocessor Architectures," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 4, July 1988, pp. 639–649.

[19] H. Goudarzi, S. Hatami, and M. Pedram, "Demand-side load scheduling incentivized by dynamic energy prices," *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Brussels, Beligium, October 2011.

[20] Y. Guo, M. Pan, and Y. Fang, "Optimal Power Management of Residential Customers in the Smart Grid," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 9, September 2012, pp. 1593–1606.

[21] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. Abdelzaher, "PDA: Privacy-Preserving Data Aggregation in Wireless Sensor Networks," *IEEE International Conference on Computer Communications (INFOCOM)*, Anchorage, Alaska, USA, May 2007.

[22] J. Horn, Nafpliotis, N., and D. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," *First IEEE Conference on Evolutionary Computation*, Orlando, FL, USA, June 1994, pp. 82–87.

[23] C. Joe-Wong, S. Sen, S. Ha, and M. Chiang, "Optimized Day-Ahead Pricing for Smart Grids with Device-Specific Scheduling Flexibility," *Selected Areas in Communications, IEEE Journal on*, vol. 30, no. 6, July 2012, pp. 1075–1085.

[24] T. Kim and H. Poor, "Scheduling Power Consumption With Price Uncertainty," *IEEE Transactions on Smart Grid*, vol. 2, no. 3, 2011, pp. 519–527.

[25] J. Knowles and D. Corne, "The Pareto Archived Evolution Strategy : A New Baseline Algorithm for Pareto Multiobjective Optimisation," *Congress on Evolutionary Computation (CEC)*, Washington, D.C., USA, July 1999.

[26] M. Laumanns, L. Thiele, E. Zitzler, and K. Deb, "Archiving with Guaranteed Convergence and Diversity in Multi-Objective Optimization," *Proceedings of the Genetic and Evolutionary Computation Conference*, New York, NY, USA, July 2002.

[27] M. LeMay and C. A. Gunter, "Cumulative attestation kernels for embedded systems," *Proceedings of the 14th European conference on Research in computer security (ESORICS)*, Saint Malo, France, September 2009.

[28] D. Li, S. Jayaweera, and A. Naseri, "Auctioning game based Demand Response scheduling in smart grid," *Online Conference on Green Communications (GreenCom)*, Online, September 2011.

[29] F. Li, B. Luo, and P. Liu, "Secure Information Aggregation for Smart Grids Using Homomorphic Encryption," *Proceedings of the First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Gaitherburg, Maryland, USA, October 2010.

[30] Q. Liang and L. Wang, "Redundancy reduction in wireless sensor networks using SVD-QR," *IEEE Military Communications Conference (MILCOM)*, Atlantic City, New Jersey, USA, October 2005.

[31] P. McDaniel and S. McLaughlin, "Security and Privacy Challenges in the Smart Grid," *IEEE Security Privacy*, vol. 7, no. 3, May-June 2009, pp. 75–77.

[32] *Global horizontal irradiance data*, Measurement and Instrumentation Data Center, 2012.

[33] A. Mohsenian-Rad, V. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous Demand-Side Management Based on Game-Theoretic Energy Consumption Scheduling for the Future Smart Grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 3, December 2010, pp. 320–331.

[34] J. Nagi, K. Yap, S. Tiong, S. Ahmed, and A. Mohammad, "Detection of abnormalities and electricity theft using genetic Support Vector Machines," *Proceedings of the IEEE Region 10 Conference*, Hyderabad, India, November 2008.

[35] *Privacy and the Smart Grid*, National Institute of Standards and Technology (NIST), 2010.

[36] M. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*, Morgan & Claypool, 2010.

[37] M. Neely, A. Tehrani, and A. Dimakis, "Efficient Algorithms for Renewable Energy Allocation to Delay Tolerant Consumers," *Proceedings of the First Internationl Conference on Smart Grid Communications (SmartGridComm)*, October 2010, pp. 549–554.

[38] A. Nizar, Z. Dong, and Y. Wang, "Power Utility Nontechnical Loss Analysis With Extreme Learning Machine Method," *IEEE Transactions on Power Systems*, vol. 23, no. 3, August 2008, pp. 946–955.

[39] B. Noble, *Applied Linear Algebra*, Prentice Hall, 1969.

[40] M. Oliveira and A. Padilha-Feltrin, "A Top-Down Approach for Distribution Loss Evaluation," *IEEE Transactions on Power Delivery*, vol. 24, no. 4, October 2009, pp. 2117–2124.

[41] A. Papavasiliou and S. Oren, "Supplying renewable energy to deferrable loads: Algorithms and economic analysis," *IEEE Power and Energy Society General Meeting*, Minneapolis, MN, July 2010.

[42] E. L. Quinn, "Privacy and the New Energy Infrastructure," *Social Science Research Network*, 2009, pp. 1995–2008.

[43] A. Ruzzelli, C. Nicolas, A. Schoofs, and G. O'Hare, "Real-Time Recognition and Profiling of Appliances through a Single Electricity Sensor," *Proceedings of the 7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, Boston, Massachusetts, USA, June 2010.

[44] S. Salinas, M. Li, and P. Li, "Multi-Objective Optimal Energy Consumption Scheduling in Smart Grids," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, March 2013, pp. 341–348.

[45] P. Samadi, R. Schober, and V. Wong, "Optimal energy consumption scheduling using mechanism design for the future smart grid," *Proceedings of IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Brussels, Belgium, October 2011.

[46] X. Shen, H. Yu, J. Buford, and M. Akon, *Handbook of Peer-to-Peer Networking*, Springer Verlag, 2010.

[47] M. Shinwari, A. Youssef, and W. Hamouda, "A Water-Filling Based Scheduling Algorithm for the Smart Grid," *IEEE Transactions on Smart Grid*, vol. 3, no. 2, June 2012, pp. 710–719.

[48] J. Smith, *Smart Meters Take Bite Out of Electricity Theft*, National Geographic, 2011.

[49] G. Strbac, "Demand side management: Benefits and challenges," *Energy Policy*, vol. 36, no. 12, November 2008, pp. 4419–4426.

[50] G. Strbac, "Demand side management: Benefits and challenges," *Energy Policy*, vol. 36, no. 12, November 2008, pp. 4419–4426.

[51] *Electricity user load profiles by profile class*, UK Energy Research Center, June 1997.

[52] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, San Jose, CA, USA, June 2011.

[53] *Data Access and Privacy Issues Related too Smart Grid Technologies*, U.S. Department of Energy, 2010.

[54] *Annual Energy Outlook*, U.S. Energy Information Administration, 2012.

[55] *Annual Energy Review*, U.S. Energy Information Administration, 2012.

[56] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective Optimization with Messy Genetic Algorithms," *Symposium on Applied Computing*, Villa Olmo, Italy, March 2000.

[57] A. K. Zille Huma Kamal, Ajay Gupta Leszek Lilien, "Classification Using Efficient LU Decomposition in Sensornets," *Proceedings of WSN*, Banff, Alberta, Canada, July 2006.

[58] E. Zitzler, M. Laumanns, and S. Bleuler, "A tutorial on Evolutionary Multiobjective Optimization," *Metaheuristics for Multiobjective Optimisation*, vol. 535, 2003, pp. 3–38.

[59] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm,", TIK-Report Nr. 103, 2001.