energies

MDPI

Article

Efficient Energy Consumption Scheduling: Towards Effective Load Leveling

Yuan Hong [1,*], Shengbin Wang [2] and Ziyue Huang [3]

[1] Department of Information Technology Management, University at Albany, SUNY, 1400 Washington Ave., Albany, NY 12222, USA
[2] Department of Marketing Transportation & Supply Chain, North Carolina A&T State University, 1601 E. Market St., Greensboro, NC 27411, USA; swang@ncat.edu
[3] Department of Information & Supply Chain Management, University of North Carolina at Greensboro, 1400 Spring Garden St., Greensboro, NC 27412, USA; z_huang4@uncg.edu
* Correspondence: hong@albany.edu

Academic Editors: Pierluigi Siano and Miadreza Shafie-khah
Received: 7 November 2016; Accepted: 12 January 2017; Published: 17 January 2017

Abstract: Different agents in the smart grid infrastructure (e.g., households, buildings, communities) consume energy with their own appliances, which may have adjustable usage schedules over a day, a month, a season or even a year. One of the major objectives of the smart grid is to flatten the demand load of numerous agents (viz. consumers), such that the peak load can be avoided and power supply can feed the demand load at anytime on the grid. To this end, we propose two Energy Consumption Scheduling (ECS) problems for the appliances held by different agents at the demand side to effectively facilitate load leveling. Specifically, we mathematically model the ECS problems as Mixed-Integer Programming (MIP) problems using the data collected from different agents (e.g., their appliances' energy consumption in every time slot and the total number of required in-use time slots, specific preferences of the in-use time slots for their appliances). Furthermore, we propose a novel algorithm to efficiently and effectively solve the ECS problems with large-scale inputs (which are NP-hard). The experimental results demonstrate that our approach is significantly more efficient than standard benchmarks, such as CPLEX, while guaranteeing near-optimal outputs.

Keywords: smart grid; scheduling; load leveling; demand response; demand side management

1. Introduction

The smart grid integrates the communication network into the existing power grid and provides operational intelligence via analyzing data collected from different agents on the grid [1], including power suppliers (e.g., utilities) and energy consumers (e.g., households, factories, universities and hospitals). For instance, smart meters are installed at the power consumer end to monitor the energy usage in a real-time fashion; the meter readings are continuously transmitted to the electric utility with a time interval as frequent as 15 min [1]. Analyzing such fine-grained meter readings (viz. the energy demand) functions for many applications for electric utilities and consumers on the grid, such as load forecasting [2], billing [3], regional statistics [4] and energy theft detection [5].

The time series power usage of different consumers directly generates the demand load (also time series) of the power supply. In reality, the demand load of both residential and commercial buildings highly fluctuates at different times [6], e.g., peak vs. off-peak times. Such fluctuation would result in many issues on the power grid. For instance, it makes it difficult for the utilities to always balance their power supply and demand load within a tight margin, then the energy transmission and production might not be optimal. Furthermore, power blackouts may occur if the power supply cannot feed the demand load over time. At the same time, the power quality (e.g., volts) of individual consumers might

be affected at peak times. Therefore, the smart grid has begun to develop techniques that can flatten the demand load of different buildings, communities or geographical areas. Specifically, many utilities try to incentivize a flattened demand load of the consumers by adopting dynamic pricing plans for the energy consumption times (e.g., time-of-use plan http://www.pge.com/: lower price at off-peak, higher at on-peak and highest at critical peak). Furthermore, the ABB Group (Automation and Power Technologies) (http://new.abb.com/substations/energy-storage-applications/load-leveling) provides a power storage-based solution (e.g., an energy bank or battery) to store the excessive energy during periods of light load and deliver it at peak times. More recently, a series of intelligent load management techniques [7–10] was proposed for smart homes to automatically implement the dynamic schedules for appliances, e.g., turning off unnecessary lights, changing the time for washing clothes, such that the peak electricity demand can be flattened to some extent.

However, most of the existing load leveling techniques have some drawbacks or limitations. First, the dynamic pricing plans highly rely on the behavioral response from each consumer to flatten the demand load. In the case that the consumers do not care about the high price of energy consumption at peak times, the plan would not be effective for load leveling. Since the response is somewhat random from the consumers, it is also challenging to quantify the effectiveness of load leveling. Second, the energy storage-based techniques require additional devices or facilities to implement the scheme and extra maintenance cost. Finally, the intelligent load management techniques [7–10] only relatively flatten the peak demand towards an optimal goal; however, the proposed solutions neither quantitatively measure the optimum nor converge towards the optimal objectives in their approaches. Meanwhile, their implementation is limited for only one household (a single smart home), rather than multiple consumers.

To address the above concerns, we propose a novel agent-based approach to flatten the demand load by optimally scheduling the usage times for appliances held by a single or multiple agent(s) (*agent refers to an energy consumer in this context*), motivated as follows. Agents (e.g., smart homes [7,10]) may have many appliances with adjustable usage schedules. For instance:

1. When to use some appliances or machines is not strictly tied to fixed time slots everyday. For instance, the air conditioner can be programmed to run at different times; washing clothes can be postponed to a certain time [10]; in a factory, machines may have adjustable schedules to manufacture and assemble parts in the morning or afternoon of a day or different days.
2. A rechargeable battery is attached with an increasing number of appliances such as electric vehicles, laptops, cordless vacuum cleaners, cell phones, tablets, etc. The batteries can be charged at any time, whereas the appliances can be used at other different times. In these cases, the battery charging time will be recorded as power consumption time by the meters.

More specifically, we focus on a single or multiple agents' appliances, each of which has a set of possible in-use time slots, and optimize their schedules to align the time series power usage in the specified time slots (involved in scheduling) to a flattened or fixed amount. For instance, Figure 1 presents a real-world household's time series power consumption over 3 h [11] (which fluctuates with several peak loads). We formulate a mathematical model to produce an optimal energy consumption schedule for all of the appliances (note that some appliances have adjustable usage schedules, while some other appliances may have fixed usage schedules; we consider both in our model, as discussed in Section 7.1), which lays the time series aggregated energy consumption (of all of the appliances) close to an ideal fixed amount, i.e., the horizontal line in Figure 1. Notice that, although the optimal solution of the scheduling problem may not be able to get the exact horizontal line, the overall deviation is minimized as a small number close to zero, as shown in the experiments.

Given p agents (each agent holds some appliances), the scheduling problem can be applied to: (1) each agent's appliances; or (2) all of the agents' appliances together. For Case (1), each agent locally formulates and solves the scheduling problem where $p = 1$. Then, each agent's demand load at different times can be flattened with its own scheduling solution, and the overall demand load of

all p agents (e.g., households in a community) can be automatically flattened to a stable aggregated amount. For Case (2), the overall demand load can be directly flattened using the scheduling solution (jointly derived from all p agents). Thus, while demanding a fixed amount of load from the grid at different times, our scheduling-based load leveling technique can greatly improve the reliability of power supply from the electric utility.

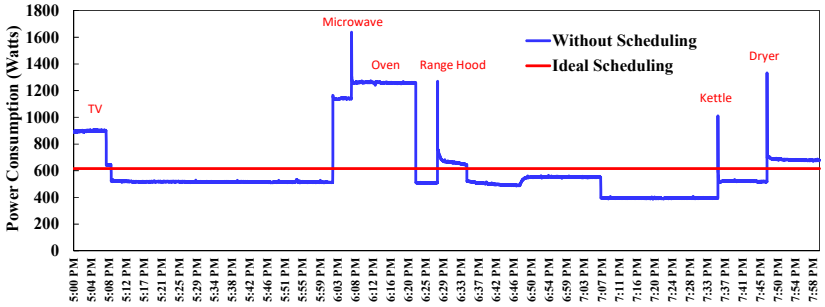

Figure 1. Energy consumption without scheduling vs. ideal scheduling.

In this paper, we define such a scheduling problem for load leveling as the Energy Consumption Scheduling (ECS) problem. Notice that the ECS problem can be integrated into both the Demand Response (DR) programs [9,12] and the Demand Side Management (DSM) programs [13–17] in shifting load to off-peak times so as to benefit both electricity consumers and utilities. More specifically, if the automated control systems are in place, our ECS problem can be implemented as a DR program that encourages energy consumers to make short-term reductions in energy demand and consume the electricity at the off-peak times. Similarly, our ECS problem can be also implemented as a DSM program to pursue energy efficiency from a long-term point of view with possible facility upgrade, such as building automation upgrades.

The ECS problem will be formulated among one or multiple agents (e.g., households), any of which (or a trusted-third party) can be the problem solver and owns the scheduling facility to derive the optimal schedule with inputs from all of the agents. Specifically, all of the agents are expected to send the information of their appliances (e.g., consumption rate) and the possible time slots of their appliances to the problem solver, which then formulates and solves the optimization problem to obtain the optimal consumption times of all of each agent's involved appliances. After solving the problem, the problem solver will distribute the optimal solution to the corresponding agents for running their appliances per their optimal schedules. In the real-world Internet of Things (IoT), formulating and solving the ECS problem, as well as the communication among agents can be implemented in the current smart grid infrastructure [1], which enables communication among entities on the existing power grid (viz. agents with computation capacity, such as smart homes [7,10]). The optimal solution can be implemented in two different ways: manual and automatic. For the former one, each agent will receive a message of the optimal running times of its appliances. Then, each agent can manually turn on their appliances per the scheduled times. For the latter one, each agent's share of the optimal solution (the scheduled times of its appliances) can be automatically implemented in the existing DR or DSM devices (e.g., a load control switch). In addition, the next generation smart grid would enable households or buildings (i.e., a smart home [7,10]) to automatically switch their appliances on and off at pre-scheduled time slots; the optimal solutions of our ECS problem can be directly implemented in such a smart home environment. Thus, our scheduling problem can be smoothly integrated into the IoT to function as load leveling at the demand side.

Note that the ECS problem is a fundamental energy consumption scheduling problem, which outputs a specific time to turn on each appliance. Then, the output schedules can be implemented in different power grid infrastructures (e.g., Asia, Europe and North America), as long as different agents can communicate with each other. Although the layouts and configurations in the load connection and

power distribution of such systems are quite different, each agent (e.g., a household) in such systems can manually or automatically turn on its appliances according to the received scheduling time in the optimal solution. In summary, the main contributions of this paper are given as follows:

- We propose a novel Energy Consumption Scheduling (ECS) problem for a single or multiple (energy demand) agents to flatten their demand load.
- We extend the ECS problem to a more generalized form (the GECS problem) by enabling each agent to specify a range of usage time for each of their appliances. This can complete the scheduling as the appliances' required usage times are unknown at the scheduling stage.
- Both ECS and GECS problems are mathematically modeled as Mixed-Integer Programming (MIP) problems. We develop a novel effective algorithm (temporal decomposition) for efficiently solving them. Note that the algorithm can return a near-optimal solution for the ECS/GECS problem with ∼1,000,000 variables in reasonable time.
- The experimental results demonstrate that our algorithm is significantly more efficient than the standard benchmarks (e.g., IBM ILOG CPLEX 12.2), while ensuring near-optimal solutions.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature. Sections 3 and 4 present the Energy Consumption Scheduling (ECS) problem and the Generalized Energy Consumption Scheduling (GECS) problem, respectively. Then, Section 5 illustrates our novel algorithm for solving the ECS and GECS problems. Section 6 demonstrates the experimental results, and Section 7 gives some discussions. Finally, Section 8 concludes the paper and discusses the future work.

2. Related Work

The smart grid overlays the power distribution network with a communication network [1], collects massive sensor data and develops automation technologies to improve the grid performance [9]. As a critical component in smart grid infrastructure, demand response management [18] aims at optimizing the power consumption at the demand side: electricity consumers. Specifically, Demand-Side Management (DSM) enables operational intelligence at the single home level and utilizes the home area network to interact with the power grid. For instance, the power usage of each appliance can be monitored in the smart home [7,10]. An increasing number of appliances, such as lights, HVACs (Heating, Ventilating and Air Conditioning) and refrigerators, can be programed to optimize the energy consumption, cut down the utility bill (e.g., automatically shutting down heating as temperature is high) and support the Internet of Things (IoT).

The proposed agent-based energy consumption scheduling in this paper pursues load leveling by further optimizing the in-use schedule of the appliances. Besides our agent-based model, some other approaches were adopted in the industry and academia, including the time-of-use energy pricing plan, storing electricity at light load and delivering it at peak times (e.g., ABB Group), and home automation (e.g., a least slack first policy/algorithm proposed by Barker et al. [10] in the SmartCap application). In addition, load leveling problems are also solved in some specific environments, such as reducing T & Dline losses [19], fault-tolerant distributed computing systems [20] and system-wide demand response management [21]. As discussed in Section 1, our agent-based energy consumption scheduling is different from the prior work, primarily in two ways. First, our scheduling solution could converge towards an optimal or near-optimal load leveling from the global point of view (multi-agents) while being subjected to the constraints derived from the in-use schedule of the appliances. Second, with the participation of multiple agents, the demand load can be flattened to a stable amount with reduced deviation of the original routine schedule of the appliances (as discussed in Section 7).

Furthermore, scheduling problems have been studied for different applications in the smart grid infrastructure. For example, Lin and Tsai [22] proposed a home energy management system facilitated by non-intrusive load monitoring techniques to save on electricity bills via scheduling. Lu et al. [23] proposed a multi-objective energy consumption scheduling to minimize the total energy consumption

cost and maximize the social utility. Paterakis et al. [24] mathematically formulated the problem of distribution network reconfiguration to determine the optimal radial configuration by minimizing the active power losses and a set of commonly-used reliability indices w.r.t. the number of customers. Chetto [25] studied the scheduling for real-time jobs, which are executed on a uniprocessor system supplied by a renewable energy source. Wang et al. [26] studied the energy-aware data allocation and task scheduling problem on multiprocessor system for real-time applications. Lin et al. [27] studied the problem of scheduling co-design for reliability and energy by minimizing total energy while guaranteeing reliability constraints. Ahmed et al. [28] proposed a hybrid Lightning Search Algorithm (LSA)-based Artificial Neural Network (ANN) to predict the optimal on/off status for residential appliances, which can provide inputs (e.g., estimated some candidate running time slots for appliances) to function as our ECS problems.

Finally, since the current centralized model of production and transmission is incredibly inefficient and places the grid under great pressure [29,30], novel multi-agent systems [31] have significantly advanced the development of the smart grid recently, especially demand response [21,28,32]. For example, Cha et al. [32] proposed a multi-agent system to perform scheduling for maximum benefit in response to the electricity prices. Agrawal et al. [33] studied the decentralized power supply restoration problem in the case that line failure occurs. The proposed multi-agent scheme can optimally cover different sub-regions and in turn lead to the agent-based decentralized control [33]. Cerquides [29] presented a multi-agent framework for microgrids on the power grid to trade their local electricity on the energy market. A network of households with solar panels or other distributed energy resources are able to sell the excess electricity to other households, which demand extra energy from the main grid. To deal with uncertainty in the microgrids' energy generation, Strawser et al. [34] developed a multi-agent power market for selling electricity, which can price reliability.

3. Energy Consumption Scheduling

We first study the ECS problem where each appliance's overall energy consuming ("in-use") time is predetermined, e.g., the washer should be continuously running for 2 hours (specified by the agent for scheduling). Some frequently-used notations are given in Table 1.

Table 1. Frequently-used notations.

p	number of agents
n	number of equally-divided time slots for scheduling
m_i	agent i's number of appliances
$i \in [1, p]$	agent index
$j \in [1, m_i]$	agent i's appliance index
$k \in [1, n]$	time slot index
e_{ij}	the energy consumption of agent i's j-th appliance in any time slot (if in-use)
c_{ij}	agent i's j-th appliance's number of in-use time slots
Δ_i	agent i's overall consumption
x_{ijk}	agent i's j-th appliance is "on" or "off" in time slot k

3.1. Objective Function

As shown in Table 1, we denote the number of time slots as n, and agent i's j-th appliance's number of in-use time slots as c_{ij} where $c_{ij} \leq n$, $i \in [1, p]$ and $j \in [1, m_i]$. Letting e_{ij} be agent i's j-th appliance's consumed energy in a single time slot, $\forall i \in [1, n]$, all of agent i's appliances' overall consumption in all of the time slots is a constant: $\Delta_i = \sum_{j=1}^{m_i}(e_{ij}c_{ij})$, where c_{ij} out of n time slots are "on" for agent i's j-th appliance.

Then, we can aggregate all of the p agents' overall consumption in n time slots as $\Delta = \sum_{i=1}^{p} \Delta_i$. In addition, all agents' overall average energy consumption within any single time slot (averaging by time) is $\frac{\Delta}{n}$. Recall that the goal of our model is to allocate all of the agents' appliances' energy

consuming time slots, such that their total power consumption within every time slot is equal to or close to $\frac{\Delta}{n}$ at all times. In other words, the sum of deviations between all of the agents' overall energy consumption within each time slot and $\frac{\Delta}{n}$ should be minimized. In addition, we define binary variables $\forall i \in [1,p]$, $\forall j \in [1,m_i]$, $\forall k \in [1,n]$, $x_{ijk} \in \{0,1\}$ as whether agent i's j-th appliance is "on" or "off" in time slot k. If $x_{ijk} = 1$, agent i's j-th appliance is "on" in time slot k; otherwise, it is "off". Thus, we have the objective function:

$$\min: \quad \sum_{k=1}^{n} |\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - \frac{\Delta}{n}| \tag{1}$$

Note that the length of time slots can be short to make every appliance be completely on or off in every time slot, then the overall power consumption would be relatively stable in every time slot. If the above objective value equals zero, the demand profile will be exactly a horizontal line (as shown in Figure 1). In most cases, the optimal objective value cannot reach zero since the variables are discrete. In those cases, we will pursue a minimum deviation close to zero, such that the demand profile would be flattened towards the horizontal line $\frac{\Delta}{n}$.

3.2. Constraints

3.2.1. Fixed Number of In-Use Time Slots

Essentially every agent's each appliance ($\forall i \in [1,p]$ and $\forall j \in [1,m_i]$) has an equality constraint: the corresponding total number of in-use time slots ("on") in the schedule is specified as c_{ij}:

$$s.t. \sum_{k=1}^{n} x_{ijk} = c_{ij} \tag{2}$$

Here, we have $\sum_{i=1}^{p} m_i$ equality constraints in total.

3.2.2. Running Appliances in Continuous Time Slots

In the real world, many appliances might be on in continuous time slots, especially in the case where the time slot is very short (e.g., 15 min or less). Then, meeting the following constraints is equivalent to making all of the appliances running in continuous time slots (details are given in Appendix A).

$$\forall i \in [1,p], \forall j \in [1,m_i]:$$

$$s.t. \begin{cases} x_{ij1}c_{ij} \le \sum_{k=1}^{c_{ij}} x_{ijk} \\ x_{ij2}(2c_{ij}-1) \le \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} \\ x_{ij3}(3c_{ij}-3) \le \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} + \sum_{k=3}^{c_{ij}+2} x_{ijk} \\ x_{ij4}(4c_{ij}-6) \le \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} + \sum_{k=3}^{c_{ij}+2} x_{ijk} \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ x_{ij(n-2)}(3c_{ij}-3) \le \sum_{k=n-c_{ij}-1}^{n-2} x_{ijk} + \sum_{k=n-c_{ij}}^{n-1} x_{ijk} + \sum_{k=n-c_{ij}+1}^{n} x_{ijk} \\ x_{ij(n-1)}(2c_{ij}-1) \le \sum_{k=n-c_{ij}}^{n-1} x_{ijk} + \sum_{k=n-c_{ij}+1}^{n} x_{ijk} \\ x_{ijn}c_{ij} \le \sum_{k=n-c_{ij}+1}^{n} x_{ijk} \end{cases} \tag{3}$$

3.2.3. Agents' Preferences of the In-Use Time Slots

If any agent has preferences for the usage time of their appliances, e.g., an agent intends to use the washer for one hour in the morning instead of the whole day (scheduling is done for all of the appliances over one day), then some local constraints can be derived for such partially adjustable schedule. For instance, the ECS problem schedules n time slots, and agent i's j-th appliance has a

partially adjustable schedule (running c_{ij} continuous time slots between time slot n_1 and time slot n_2 where $0 < n_1 < n_2 < n$ and $c_{ij} \leq n_2 - n_1$. A set of constraints can be given by letting $\forall k \in [1, n_1 - 1]$ and $\forall k \in [n_2 + 1, n]$, $x_{ijk} = 0$. Similarly, $x_{ijk} = 1$ can be also specified by agent i based on its preferences.

Indeed, such constraints could reduce the complexity of the ECS problem. We can denote this kind of constraint as "$x_{ijk} = 0$ or 1 if specified by agent i" in the mathematical models.

3.3. Problem Formulation

As a result, after combining the objective function (Equation (1)) and constraints (Equality Constraints (2) and Inequality Constraints (3)), we can mathematically formulate our ECS problem as below:

$$\min: \quad \sum_{k=1}^{n} |\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - \frac{\Delta}{n}|$$

$$\forall i \in [1,p], \forall j \in [1,m_i]:$$

$$s.t. \begin{cases} \sum_{k=1}^{n} x_{ijk} = c_{ij} \\ x_{ij1}c_{ij} \leq \sum_{k=1}^{c_{ij}} x_{ijk} \\ x_{ij2}(2c_{ij}-1) \leq \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} \\ x_{ij3}(3c_{ij}-3) \leq \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} + \sum_{k=3}^{c_{ij}+2} x_{ijk} \\ x_{ij4}(4c_{ij}-6) \leq \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} + \sum_{k=3}^{c_{ij}+2} x_{ijk} \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ x_{ij(n-2)}(3c_{ij}-3) \leq \sum_{k=n-c_{ij}-1}^{n-2} x_{ijk} + \sum_{k=n-c_{ij}}^{n-1} x_{ijk} + \sum_{k=n-c_{ij}+1}^{n} x_{ijk} \\ x_{ij(n-1)}(2c_{ij}-1) \leq \sum_{k=n-c_{ij}}^{n-1} x_{ijk} + \sum_{k=n-c_{ij}+1}^{n} x_{ijk} \\ x_{ijn}c_{ij} \leq \sum_{k=n-c_{ij}+1}^{n} x_{ijk} \\ \forall k \in [1,n], x_{ijk} \in \{0,1\} (x_{ijk} = 0 \text{ or } 1 \text{ if specified by agent } i) \end{cases} \tag{4}$$

where constant $\Delta = \sum_{i=1}^{p}\sum_{j=1}^{m_i}(c_{ij}e_{ij})$. Letting $\forall k \in [1,n], y_k = |\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - \frac{\Delta}{n}|$ (overall deviation in time slot k), the problem can be transformed into the following Mixed-Integer Programming (MIP) problem (details are given in Appendix B):

$$\min: \sum_{k=1}^{n} y_k$$

$$\begin{cases} \forall i \in [1,p], \forall j \in [1,m_i], \sum_{k=1}^{n} x_{ijk} = c_{ij} \\ \forall k \in [1,n], \sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - \frac{\Delta}{n} \leq y_k \\ \forall k \in [1,n], -\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) + \frac{\Delta}{n} \leq y_k \\ \forall k \in [1,n], y_k \geq 0 \\ \forall i \in [1,p], \forall j \in [1,m_i]: \\ s.t. \begin{cases} x_{ij1}c_{ij} \leq \sum_{k=1}^{c_{ij}} x_{ijk} \\ x_{ij2}(2c_{ij}-1) \leq \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} \\ x_{ij3}(3c_{ij}-3) \leq \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} + \sum_{k=3}^{c_{ij}+2} x_{ijk} \\ x_{ij4}(4c_{ij}-6) \leq \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} + \sum_{k=3}^{c_{ij}+2} x_{ijk} \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ x_{ij(n-2)}(3c_{ij}-3) \leq \sum_{k=n-c_{ij}-1}^{n-2} x_{ijk} + \sum_{k=n-c_{ij}}^{n-1} x_{ijk} + \sum_{k=n-c_{ij}+1}^{n} x_{ijk} \\ x_{ij(n-1)}(2c_{ij}-1) \leq \sum_{k=n-c_{ij}}^{n-1} x_{ijk} + \sum_{k=n-c_{ij}+1}^{n} x_{ijk} \\ x_{ijn}c_{ij} \leq \sum_{k=n-c_{ij}+1}^{n} x_{ijk} \end{cases} \\ \forall i \in [1,p], \forall j \in [1,m_i], \forall k \in [1,n], x_{ijk} \in \{0,1\} \\ (x_{ijk} = 0 \text{ or } 1 \text{ if specified by agent } i) \end{cases} \tag{5}$$

3.4. Measures of Evaluating Scheduling Solutions

The objective value of the ECS problem would not be zero in general, and solving any ECS problem using different algorithms may result in different optimal solutions (due to the NP-hard nature of the mixed integer programming problem). Therefore, we define a measure to evaluate the accuracy of the optimal scheduling solutions obtained by different algorithms, the deviation ratio, which is defined as the minimum overall deviations (in the optimal solution) divided by the overall consumption in all of the time slots:

$$DevRatio = \frac{\sum_{k=1}^{n} y_k^*}{\Delta} = \frac{\sum_{k=1}^{n} y_k^*}{\sum_{i=1}^{p}\sum_{j=1}^{m_i}(c_{ij}e_{ij})} \tag{6}$$

Hence, we can use the above measure to compare the accuracy of different algorithms used to solve the ECS problems. In addition, the ECS problem minimizes the overall deviation. Thus, we can normalize the overall deviation with the initial overall deviation to examine how much deviation has been reduced in the optimal solution or other solutions in the problem solving process. To this end, we normalize the overall deviation into [0,1] using the following formula:

$$Normalized\ Deviation = \frac{\sum_{k=1}^{n} y_k(\text{Any Solution})}{\sum_{k=1}^{n} y_k(\text{Initial Solution})} \tag{7}$$

In Section 6, we demonstrate the experimental results using the above two measures.

4. Generalized ECS Problem

In this section, we extend the ECS problem to a more general form in which the overall consumption of each appliance (i.e., agent i's j-th appliance) is a variable in range $[a_{ij}, b_{ij}]$ rather than fixing it as c_{ij}. This extension works in the case that some appliances' total number of required in-use time slots is still unknown at the scheduling stage. If we let $\forall i \in [1, p]$, $\forall j \in [1, m_i]$, $a_{ij} = b_{ij} = c_{ij}$, the new problem is then reduced to the original ECS problem. Thus, we name this more general problem as the GECS problem.

In the GECS problem, given c_{ij} and z_{ij}, we denote agent i's j-th's appliance's total number of in-use time slots as x_{ij}. Thus, we have $x_{ij} = \sum_{k=1}^{n} x_{ijk}$, and the overall energy consumptions are:

$$\Delta = \sum_{i=1}^{p} \Delta_i = \sum_{i=1}^{p}\sum_{j=1}^{m_i}(e_{ij}x_{ij}) = \sum_{i=1}^{p}\sum_{j=1}^{m_i}(e_{ij}\sum_{k=1}^{n} x_{ijk}) \tag{8}$$

where Δ is a variable rather than a constant in the GECS problem. Similar to the ECS problem, we can derive the objective function and constraints as below.

4.1. Objective Function

Replacing Δ in the ECS problem's objective function (Equation (1)), we thus have the objective function of the GECS problem:

$$\min: \quad \sum_{k=1}^{n} |\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - \frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{m_i}(e_{ij}\sum_{k=1}^{n} x_{ijk})| \tag{9}$$

4.2. Constraints

Similar to the ECS problem, we derive the constraints for the number of in-use time slots, running appliances in continuous time slots and agents' preferences of the in-use time slots (if available).

4.2.1. Number of Running Time Slots

Note that in the GECS problem, the total number of in-use time slots for each appliance is a variable $x_{ij} \in [a_{ij}, b_{ij}]$. Then, we have the following two groups of constraints:

$$\begin{cases} \forall i \in [1,p], \forall j \in [1,m_i], \sum_{k=1}^{n} x_{ijk} \geq a_{ij} \\ \forall i \in [1,p], \forall j \in [1,m_i], \sum_{k=1}^{n} x_{ijk} \leq b_{ij} \end{cases} \tag{10}$$

4.2.2. Running Appliances in Continuous Time Slots

Similar to the ECS problem, we can obtain all of the inequality constraints for running appliances in continuous time slots as below (details are given in Appendix C):

$$\forall i \in [1,p], \forall j \in [1,m_i]:$$

$$s.t. \begin{cases} x_{ij1} \sum_{k=1}^{n} x_{ijk} \leq \sum_{k=1}^{c_{ij}} x_{ijk} \\ x_{ij2}(2\sum_{k=1}^{n} x_{ijk} - 1) \leq \sum_{k=1}^{\sum_{k=1}^{n} x_{ijk}} x_{ijk} + \sum_{k=2}^{\sum_{k=1}^{n} x_{ijk}+1} x_{ijk} \\ x_{ij3}(3\sum_{k=1}^{n} x_{ijk} - 3) \leq \sum_{k=1}^{\sum_{k=1}^{n} x_{ijk}} x_{ijk} + \sum_{k=2}^{\sum_{k=1}^{n} x_{ijk}+1} x_{ijk} + \sum_{k=3}^{\sum_{k=1}^{n} x_{ijk}+2} x_{ijk} \\ x_{ij4}(4\sum_{k=1}^{n} x_{ijk} - 6) \leq \sum_{k=1}^{\sum_{k=1}^{n} x_{ijk}} x_{ijk} + \sum_{k=2}^{\sum_{k=1}^{n} x_{ijk}+1} x_{ijk} + \sum_{k=3}^{\sum_{k=1}^{n} x_{ijk}+2} x_{ijk} \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ x_{ij(n-2)}(3\sum_{k=1}^{n} x_{ijk} - 3) \leq \sum_{k=n-\sum_{k=1}^{n} x_{ijk}-1}^{n-2} x_{ijk} + \sum_{k=n-\sum_{k=1}^{n} x_{ijk}}^{n-1} x_{ijk} + \sum_{k=n-\sum_{k=1}^{n} x_{ijk}+1}^{n} x_{ijk} \\ x_{ij(n-1)}(2\sum_{k=1}^{n} x_{ijk} - 1) \leq \sum_{k=n-\sum_{k=1}^{n} x_{ijk}}^{n-1} x_{ijk} + \sum_{k=n-\sum_{k=1}^{n} x_{ijk}+1}^{n} x_{ijk} \\ x_{ijn} \sum_{k=1}^{n} x_{ijk} \leq \sum_{k=n-\sum_{k=1}^{n} x_{ijk}+1}^{n} x_{ijk} \end{cases} \tag{11}$$

where $\forall i \in [1,p], \forall j \in [1,m_i], x_{ij} = \sum_{k=1}^{n} x_{ijk}$.

4.3. Problem Formulation

Similar to the optimization model of the ECS problem (Equation (4)), we can mathematically formulate the GECS problem as below:

$$\min: \quad \sum_{k=1}^{n} \left| \sum_{i=1}^{p} \sum_{j=1}^{m_i} (x_{ijk} e_{ij}) - \frac{1}{n} \sum_{i=1}^{p} \sum_{j=1}^{m_i} (e_{ij} \sum_{k=1}^{n} x_{ijk}) \right|$$

$$\forall i \in [1,p], \forall j \in [1,m_i]:$$

$$s.t. \begin{cases} \sum_{k=1}^{n} x_{ijk} \geq a_{ij} \\ \sum_{k=1}^{n} x_{ijk} \leq b_{ij} \\ x_{ij1} \sum_{k=1}^{n} x_{ijk} \leq \sum_{k=1}^{c_{ij}} x_{ijk} \\ x_{ij2}(2\sum_{k=1}^{n} x_{ijk} - 1) \leq \sum_{k=1}^{\sum_{k=1}^{n} x_{ijk}} x_{ijk} + \sum_{k=2}^{\sum_{k=1}^{n} x_{ijk}+1} x_{ijk} \\ x_{ij3}(3\sum_{k=1}^{n} x_{ijk} - 3) \leq \sum_{k=1}^{\sum_{k=1}^{n} x_{ijk}} x_{ijk} + \sum_{k=2}^{\sum_{k=1}^{n} x_{ijk}+1} x_{ijk} + \sum_{k=3}^{\sum_{k=1}^{n} x_{ijk}+2} x_{ijk} \\ x_{ij4}(4\sum_{k=1}^{n} x_{ijk} - 6) \leq \sum_{k=1}^{\sum_{k=1}^{n} x_{ijk}} x_{ijk} + \sum_{k=2}^{\sum_{k=1}^{n} x_{ijk}+1} x_{ijk} + \sum_{k=3}^{\sum_{k=1}^{n} x_{ijk}+2} x_{ijk} \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ x_{ij(n-2)}(3\sum_{k=1}^{n} x_{ijk} - 3) \leq \sum_{k=n-\sum_{k=1}^{n} x_{ijk}-1}^{n-2} x_{ijk} + \sum_{k=n-\sum_{k=1}^{n} x_{ijk}}^{n-1} x_{ijk} + \sum_{k=n-\sum_{k=1}^{n} x_{ijk}+1}^{n} x_{ijk} \\ x_{ij(n-1)}(2\sum_{k=1}^{n} x_{ijk} - 1) \leq \sum_{k=n-\sum_{k=1}^{n} x_{ijk}}^{n-1} x_{ijk} + \sum_{k=n-\sum_{k=1}^{n} x_{ijk}+1}^{n} x_{ijk} \\ x_{ijn} \sum_{k=1}^{n} x_{ijk} \leq \sum_{k=n-\sum_{k=1}^{n} x_{ijk}+1}^{n} x_{ijk} \\ \forall k \in [1,n], x_{ijk} \in \{0,1\} (x_{ijk} = 0 \text{ or } 1 \text{ if specified by agent } i) \end{cases} \tag{12}$$

where $\forall i \in [1,p], \forall j \in [1,m_i], x_{ij} = \sum_{k=1}^{n} x_{ijk}$. Similar to the ECS problem, we can denote the sum of absolute values by additional variables $y_1, \ldots, y_n$. Therefore, after removing the absolute values (similar to Equation (5)), the GECS problem can be transformed to:

$$\min: \sum_{k=1}^{n} y_k$$

$$s.t. \begin{cases} \forall i \in [1,p], \forall j \in [1,m_i], \sum_{k=1}^{n} x_{ijk} \le b_{ij} \\ \forall i \in [1,p], \forall j \in [1,m_i], -\sum_{k=1}^{n} x_{ijk} \le -a_{ij} \\ \forall k \in [1,n], \sum_{i=1}^{p} \sum_{j=1}^{m_i} (x_{ijk} e_{ij} - \frac{1}{n}(e_{ij} \sum_{k=1}^{n} x_{ijk})) - y_k \le 0 \\ \forall k \in [1,n], -\sum_{i=1}^{p} \sum_{j=1}^{m_i} (x_{ijk} e_{ij} - \frac{1}{n}(e_{ij} \sum_{k=1}^{n} x_{ijk})) - y_k \le 0 \\ \forall k \in [1,n], y_k \ge 0 \\ \forall i \in [1,p], \forall j \in [1,m_i]: \\ \begin{cases} \sum_{k=1}^{n} x_{ijk} \ge a_{ij} \\ \sum_{k=1}^{n} x_{ijk} \le b_{ij} \\ x_{ij1} \sum_{k=1}^{n} x_{ijk} \le \sum_{k=1}^{c_{ij}} x_{ijk} \\ x_{ij2}(2\sum_{k=1}^{n} x_{ijk} - 1) \le \sum_{k=1}^{\sum_{k=1}^{n} x_{ijk}} x_{ijk} + \sum_{k=2}^{\sum_{k=1}^{n} x_{ijk}+1} x_{ijk} \\ x_{ij3}(3\sum_{k=1}^{n} x_{ijk} - 3) \le \sum_{k=1}^{\sum_{k=1}^{n} x_{ijk}} x_{ijk} + \sum_{k=2}^{\sum_{k=1}^{n} x_{ijk}+1} x_{ijk} + \sum_{k=3}^{\sum_{k=1}^{n} x_{ijk}+2} x_{ijk} \\ x_{ij4}(4\sum_{k=1}^{n} x_{ijk} - 6) \le \sum_{k=1}^{\sum_{k=1}^{n} x_{ijk}} x_{ijk} + \sum_{k=2}^{\sum_{k=1}^{n} x_{ijk}+1} x_{ijk} + \sum_{k=3}^{\sum_{k=1}^{n} x_{ijk}+2} x_{ijk} \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ x_{ij(n-2)}(3\sum_{k=1}^{n} x_{ijk} - 3) \le \sum_{k=n-\sum_{k=1}^{n} x_{ijk}-1}^{n-2} x_{ijk} + \sum_{k=n-\sum_{k=1}^{n} x_{ijk}}^{n-1} x_{ijk} + \sum_{k=n-\sum_{k=1}^{n} x_{ijk}+1}^{n} x_{ijk} \\ x_{ij(n-1)}(2\sum_{k=1}^{n} x_{ijk} - 1) \le \sum_{k=n-\sum_{k=1}^{n} x_{ijk}}^{n-1} x_{ijk} + \sum_{k=n-\sum_{k=1}^{n} x_{ijk}+1}^{n} x_{ijk} \\ x_{ijn} \sum_{k=1}^{n} x_{ijk} \le \sum_{k=n-\sum_{k=1}^{n} x_{ijk}+1}^{n} x_{ijk} \end{cases} \\ \forall i \in [1,p], \forall j \in [1,m_i], \forall k \in [1,n], x_{ijk} \in \{0,1\} (x_{ijk} = 0 \text{ or } 1 \text{ if specified by agent } i) \end{cases} \tag{13}$$

where $\forall i \in [1,p], \forall j \in [1,m_i], x_{ij} = \sum_{k=1}^{n} x_{ijk}$.

5. Algorithms

In this section, we first describe the overview of the energy consumption scheduling and then present an efficient algorithm for one or multiple agents to effectively solve the ECS (or GECS) problem and implement the optimal scheduling solution for load leveling.

5.1. Overview

As described in Section 1, all of the agents first send their appliances' consumption rates and estimated running times to the problem solver, which can be any agent or a trusted-third party. If the appliances' running times are given as a fixed number of time slots, then an ECS problem will be formulated (as shown in Section 3); if the appliances' running times are given as ranges of time slots, then an GECS problem will be formulated (as shown in Section 4).

For the ECS problem, we propose a novel algorithm denoted as the Temporal Decomposition (TD) to let the problem solver efficiently solve it. The details of the TD algorithm are given in Section 5.2. For the GECS problem, the problem solver first utilizes Linear Programming (LP) relaxation to find out the optimal consumption amount of each appliance. Then, considering the optimal solution of the LP problem as the fixed consumption amount for each appliance, the GECS problem can be converted to an ECS problem, which can be solved using the TD algorithm by the problem solver. The details of solving the GECS problem are given in Section 5.3.

Finally, the problem solver distributes the shares of the optimal solution to the corresponding agents, which are their appliances' specific running time slots. As a result, all of the agents can turn on their appliances in the specific running time slots in the optimal scheduling solution: for all $x_{ijk} = 1$, agent i turns on its j-th appliance in time slot k. Figure 2 demonstrates the flow diagram of the energy consumption scheduling for load leveling (both ECS and GECS problems).

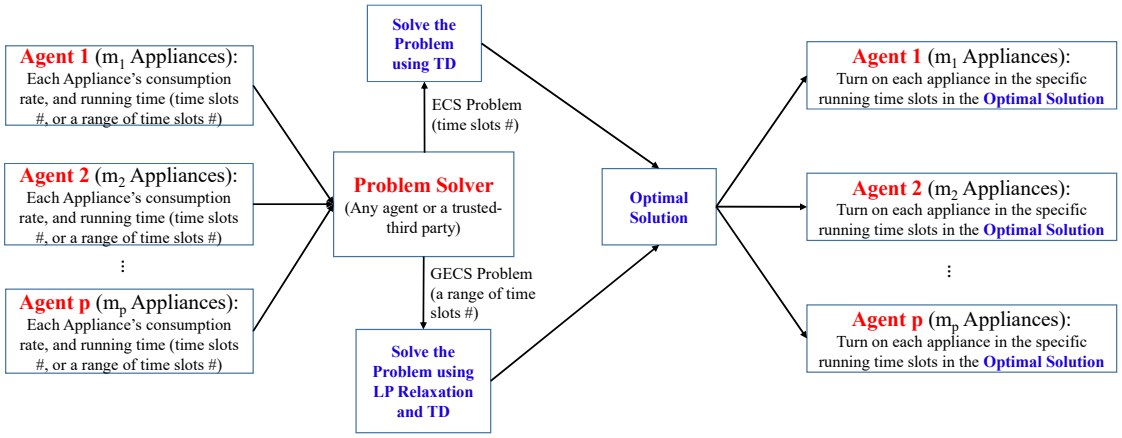

Figure 2. Overview of the energy consumption scheduling for load leveling.

5.2. Solving the ECS Problem with Temporal Decomposition

The ECS problem involves $2n$ global constraints with respect to $\sum_{i=1}^{p}[m_i(n+1)]$ variables shared by all p agents, while $\forall i \in [1,p]$, agent i holds m_i local equality constraints and $m_i n$ local inequality constraints with respect to $m_i \times n$ variables. Since such a Mixed-Integer Programming problem (MIP) includes an overwhelming majority of binary variables with a number of $\sum_{i=1}^{p}(m_i n)$, the commercial solvers, such as CPLEX or GUROBI [35], cannot produce an optimal solution within reasonable time as $\forall i \in [1,p]$, m_i, p and/or n are large. Thus, we design an efficient heuristic algorithm to effectively and efficiently generate optimal or near-optimal solutions.

Specifically, we decompose the ECS problem into n subproblems for n different time slots in the scheduling. The algorithm begins with solving the subproblem regarding the first time slot $k = 1$, and solves all of the subproblems in a temporal sequence. Thus, we denote the algorithm as "temporal decomposition".

Since the constant $\frac{\Delta}{n}$ represents the optimal overall consumption amount for every time slot, the process of optimizing all p agents' appliances' overall consumption in n different time slots is relatively independent. In each subproblem, $\forall k \in [1,n]$, the objective function is simply given as y_k, which is time slot k's share in the ECS problem's original objective function, the deviation between all agents' appliances' consumption in time slot k and the constant optimal amount $\frac{\Delta}{n}$.

More specifically, in the ECS problem, n pairs of global constraints are given for representing the deviation of n different time slots' consumption and $\frac{\Delta}{n}$, respectively, which are independent of each other; n pairs of global constraints do not have any overlapped variables. Thus, $\forall k \in [1,n]$ the k-th subproblem only needs to involve a pair of global constraints (corresponding to time slot k). We can first formulate n decomposed subproblems with only global constraints. $\forall k \in [1,n]$, the k-th subproblem is:

$$\min: \ y_k$$
$$s.t. \begin{cases} \sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - y_k \le \frac{\Delta}{n} \\ -\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - y_k \le \frac{\Delta}{n} \\ y_k \ge 0, \forall i \in [1,p], \forall j \in [1,m_i], x_{ijk} \in \{0,1\} \\ (x_{ijk} = 0 \text{ or } 1 \text{ if specified by agent } i) \end{cases} \quad (14)$$

where the variables in the k-th subproblem are a subset of variables in the ECS problem, which correspond to time slot k. We denote its optimal solution as $\forall i \in [1,p], \forall j \in [1,m_i], x^*_{ijk}$, the optimal values indicating whether all of p agents' appliances are on or off in time slot k. In the meantime, deviation y_k is minimized to y^*_k.

Furthermore, we also have to take into account the ECS problem's local constraints. Recall that each local constraint is created with two criteria:

1. agent i's j-th appliance's total number of in-use time slots is c_{ij} in the scheduling.
2. all c_{ij} in-use time slots are continuous.

Then, we can simplify all of the local constraints according to two groups of rules that assign values for variables $\forall i \in [1, p]$, $j \in [1, m_i]$, $k \in [1, n]$, x_{ijk} in all n decomposed subproblems in which x_{ijk} represents the on/off status of agent i's j-th appliance in time slot k. Specifically, in time slot $k, 1 \leq k \leq n$'s subproblem (note that $k-1$ subproblems have been solved):

- Rules of the number of in-use time slots. For agent i's j-th appliance:
  - Rule 1.1: if $\sum_{u=1}^{k-1} x_{iju}^* = c_{ij}$, then all of the binary variables $\forall u \in [k+1, n]$, x_{iju} in the remaining $(n-k+1)$ subproblems (including the current subproblem) must be zero. This rule means that if an appliance has been on for c_{ij} time slots in the past $(k-1)$ time slots, it must be off in all of the remaining time slots.
  - Rule 1.2: if $c_{ij} - \sum_{u=1}^{k-1} x_{iju}^* = n-(k-1)$, then all of the binary variables $\forall u \in [k, n]$, x_{iju} in the remaining $(n-k+1)$ subproblems (including the current subproblem) must be one. This rule means that if an appliance has been off for $(n-c_{ij})$ time slots in the past $(k-1)$ time slots, it must be on in all of the remaining time slots.
  - Rule 1.3: if $0 < c_{ij} - \sum_{u=1}^{k-1} x_{iju}^* < n-(k-1)$, then all of the binary variables $\forall u \in [1, n]$, x_{iju} in the remaining $(n-k+1)$ subproblems (including the current one) can remain as either zero or one. This rule means that if an appliance has neither been on for c_{ij} time slots nor been off for $(n-c_{ij})$ time slots in the past $(k-1)$ time slots, it can be either on or off in the following time slots.
- Rules of continuous in-use time slots. For agent i's j-th appliance:
  - Rule 2.1: if $\sum_{u=1}^{k-1} x_{iju}^* = c_{ij}$, then the binary variable x_{ijk} in the current subproblem and the binary variables $\forall u \in [k+1, n]$, x_{iju} in all of the remaining subproblems must be zero (the same as Rule 1.1). This rule means that if an appliance has been continuously on for c_{ij} time slots in the past $(k-1)$ time slots, it must be off in all of the remaining time slots.
  - Rule 2.2: if $\sum_{u=1}^{k-1} x_{iju}^* < c_{ij}$ and $x_{ij(k-1)}^* = 1$, then the binary variable x_{ijk} in the current subproblem, and the binary variables $\forall u \in [k+1, k+c_{ij} - \sum_{u=1}^{k-1} x_{iju}^* - 1]$, x_{iju} in the following $(c_{ij} - \sum_{u=1}^{k-1} x_{iju}^* - 1)$ subproblems must be one(the binary variables in c_{ij} continuous subproblems are one). This rule means that if an appliance is on in the most recent time slot and has not been on for c_{ij} time slots in the past $(k-1)$ time slots yet, it must be on in the following $(c_{ij} - \sum_{u=1}^{k-1} x_{iju}^* - 1)$ time slots.
  - Rule 2.3: if $\sum_{u=1}^{k-1} x_{iju}^* < c_{ij}$ and $x_{ij(k-1)}^* = 0$ (no in-use time slot yet), then the binary variable x_{ijk} in the current subproblem can be either zero or one. This rule means that if an appliance has not been on for c_{ij} time slots in the past $(k-1)$ time slots, it must be off in all of the past time slots (due to the characteristics of continuous running). Then, it can be either on or off in the following time slots.

Note that all six rules will be applied to n decomposed subproblems from a global perspective. Since the first group of rules ensures that agent i's j-th appliance's total number of consumption time slots equals c_{ij} and the second group of rules ensures that such c_{ij} in-use time slots are continuous, the compliance of the above two groups of rules is equivalent to meeting all of the local constraints in the ECS problem.

After solving the n subproblems (notice that: (1) without loss of generality, any agent or a centralized site can be the solver; (2) $\forall k \in [1, n]$, the k-th subproblems is jointly formulated by all the p agents; $\forall i \in [1, p]$, agent i inputs its share of the problem corresponding to time slot k; (3) each agent

utilizes all six rules and its local constraints to examine the possible values of their variables in every subproblem), the optimal solutions of all of the subproblems can directly form the optimal solution of the original ECS problem: optimal value $y_1^* + y_2^* + \cdots + y_n^*$; optimal solution $\forall x_{ijk}^*$. The details of the temporal decomposition are presented in Algorithm 1 and Figure 3, and the accuracy of the temporal decomposition algorithm is validated in Section 6.

Algorithm 1: Temporal decomposition.

1 **forall the** *time slot* $k \in [1, n]$ **do**

2 retrieve the optimal values in the previously solved $k-1$ subproblems: $\forall i \in [1, p]$, $\forall j \in [1, m_i]$, $\forall u \in [1, k-1]$, x_{iju}^*

3 check two groups of rules: Rules 1.1, 1.2, 1.3 and 2.1, 2.2, 2.3 with the following values to decide the available binary values for the variables in the current subproblem (k-th):

- $\forall i \in [1, p]$, $\forall j \in [1, m_i]$, c_{ij}
- $\forall i \in [1, p]$, $\forall j \in [1, m_i]$, $\forall u \in [1, k-1]$, x_{iju}^*

solve the k-th subproblem with formulation shown in Equation (14) to obtain the optimal solution $\forall i \in [1, p]$, $\forall j \in [1, m_i]$, x_{ijk}^* and y_k^*

4 return all of the optimal solutions in all k subproblems

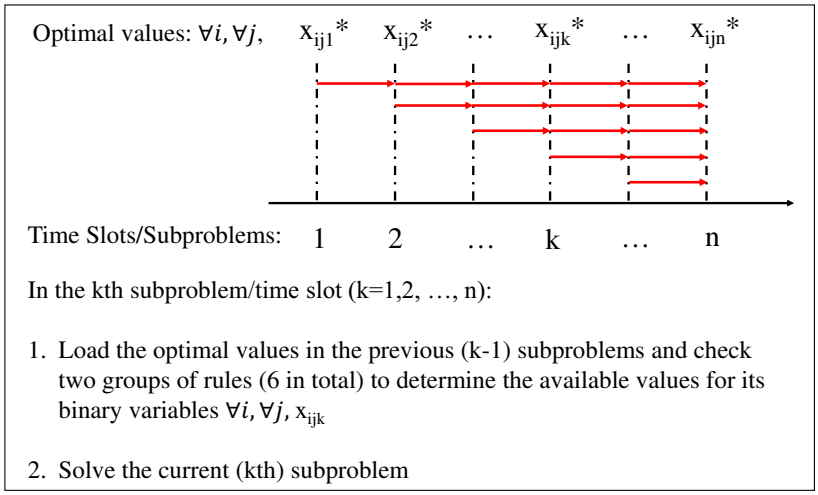

Figure 3. Temporal decomposition.

5.3. Solving the GECS Problem with Linear Programming Relaxation and Temporal Decomposition

Since the constant optimal consumption amount of any time slot $\frac{\Delta}{n}$ in the ECS problem has been changed into $\frac{1}{n}\sum_{i=1}^{p}\sum_{j=1}^{m_i}(e_{ij}\sum_{k=1}^{n}x_{ijk})$ in the GECS problem (which is not a constant), Algorithm 1 cannot be directly applied to solve the GECS problem. To tackle this issue, we propose a two-phase approach to solve the GECS problem: (1) find the optimal consumption amount for each of the appliances; and (2) solve the scheduling problem with the optimal consumption amounts (fixed).

In Phase (1), the problem solver relaxes $\forall k \in [1, n], x_{ijk}$ from binary variable $\{0, 1\}$ to continuous range $[0, 1]$ in the GECS problem and defines new integer variables $\forall i \in [1, p]$, $\forall j \in [1, m_i]$, $\omega_{ij} = \sum_{k=1}^{n} x_{ijk}$ to approximate each appliance's total number of in-use time slots (similar to c_{ij} in the ECS problem). Then, the problem solver can formulate and solve the following LP relaxation problem:

$$\min: \sum_{k=1}^{n} y_k$$

$$s.t. \begin{cases} \sum_{j=1}^{m_1}[x_{1j1} - \frac{1}{n}\omega_{1j}]e_{1j} + \cdots + \sum_{j=1}^{m_p}[x_{pj1} - \frac{1}{n}\omega_{pj}]e_{pj} - y_1 \leq 0 \\ -\sum_{j=1}^{m_1}[x_{1j1} - \frac{1}{n}\omega_{1j}]e_{1j} - \cdots - \sum_{j=1}^{m_p}[x_{pj1} - \frac{1}{n}\omega_{pj}]e_{pj} - y_1 \leq 0 \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ \sum_{j=1}^{m_1}[x_{1jn} - \frac{1}{n}\omega_{1j}]e_{1j} + \cdots + \sum_{j=1}^{m_p}[x_{pjn} - \frac{1}{n}\omega_{pj}]e_{pj} - y_n \leq 0 \\ -\sum_{j=1}^{m_1}[x_{1jn} - \frac{1}{n}\omega_{1j}]e_{1j} - \cdots - \sum_{j=1}^{m_p}[x_{pjn} - \frac{1}{n}\omega_{pj}]e_{pj} - y_n \leq 0 \\ \forall i \in [1,p], \forall j \in [1,m_p], \omega_{ij} = \sum_{k=1}^{n} x_{ijk} \\ \forall i \in [1,p], \forall j \in [1,m_p], a_{ij} \leq \omega_{ij} \leq b_{ij} \\ y_k \geq 0, \forall i \in [1,p], \forall j \in [1,m_i], 0 \leq x_{ijk} \leq 1 \end{cases} \tag{15}$$

After solving the above LP relaxation problem, $\forall i \in [1,p], \forall j \in [1,m_i], \omega_{ij}$ can be fixed as constants with the optimal values in the LP relaxation problem $\forall \omega_{ij}^*$ (which are rounded to integers $\forall \lfloor \omega_{ij}^* \rfloor$). Therefore, the optimal value ω_{ij}^* can serve as the (optimal) total energy consumption of agent i's j-th appliance. As a result, the GECS problem is transformed into an ECS problem.

In Phase (2), the problem solver applies temporal decomposition (Algorithm 1) to solve the transformed GECS problem with the optimal consumption amounts of the appliances (derived from Phase (1)). The GECS problem (viz. an ECS problem) is formulated as below:

$$\min: \quad \sum_{k=1}^{n} |\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - \frac{\Delta}{n}|$$

$$s.t. \begin{cases} \forall i \in [1,p], \forall j \in [1,m_i], \sum_{k=1}^{n} x_{ijk} = \lfloor \omega_{ij}^* \rfloor \\ \forall i \in [1,p], \forall j \in [1,m_i]: \\ \quad \begin{cases} x_{ij1}\lfloor \omega_{ij}^* \rfloor \leq \sum_{k=1}^{\lfloor \omega_{ij}^* \rfloor} x_{ijk} \\ x_{ij2}(2\lfloor \omega_{ij}^* \rfloor - 1) \leq \sum_{k=1}^{\lfloor \omega_{ij}^* \rfloor} x_{ijk} + \sum_{k=2}^{\lfloor \omega_{ij}^* \rfloor + 1} x_{ijk} \\ x_{ij3}(3\lfloor \omega_{ij}^* \rfloor - 3) \leq \sum_{k=1}^{\lfloor \omega_{ij}^* \rfloor} x_{ijk} + \sum_{k=2}^{\lfloor \omega_{ij}^* \rfloor + 1} x_{ijk} + \sum_{k=3}^{\lfloor \omega_{ij}^* \rfloor + 2} x_{ijk} \\ x_{ij4}(4\lfloor \omega_{ij}^* \rfloor - 6) \leq \sum_{k=1}^{\lfloor \omega_{ij}^* \rfloor} x_{ijk} + \sum_{k=2}^{\lfloor \omega_{ij}^* \rfloor + 1} x_{ijk} + \sum_{k=3}^{\lfloor \omega_{ij}^* \rfloor + 2} x_{ijk} \\ \quad \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ x_{ij(n-2)}(3\lfloor \omega_{ij}^* \rfloor - 3) \leq \sum_{k=n-\lfloor \omega_{ij}^* \rfloor - 1}^{n-2} x_{ijk} + \sum_{k=n-\lfloor \omega_{ij}^* \rfloor}^{n-1} x_{ijk} + \sum_{k=n-\lfloor \omega_{ij}^* \rfloor + 1}^{n} x_{ijk} \\ x_{ij(n-1)}(2\lfloor \omega_{ij}^* \rfloor - 1) \leq \sum_{k=n-\lfloor \omega_{ij}^* \rfloor}^{n-1} x_{ijk} + \sum_{k=n-\lfloor \omega_{ij}^* \rfloor + 1}^{n} x_{ijk} \\ x_{ijn}\lfloor \omega_{ij}^* \rfloor \leq \sum_{k=n-\lfloor \omega_{ij}^* \rfloor + 1}^{n} x_{ijk} \end{cases} \\ \forall i \in [1,p], \forall j \in [1,m_i], \forall k \in [1,n], x_{ijk} \in \{0,1\} (x_{ijk} = 0 \text{ or } 1 \text{ if specified by agent } i) \end{cases} \tag{16}$$

where constant $\Delta = \sum_{i=1}^{p}\sum_{j=1}^{m_i}(e_{ij}\lfloor \omega_{ij}^* \rfloor)$. Note that $\forall \lfloor \omega_{ij}^* \rfloor$ will be loaded into the two groups of rules as agent i's i-th appliance's overall consumption time in n time slots (viz. c_{ij} in the ECS problem) for satisfying all of the local constraints. Similar to the ECS problem, after solving the problem, the problem solver distributes the optimal solution to each agent. For all $x_{ijk} = 1$ in the optimal solution, agent i turns on its j-th appliance in time slot k.

6. Experimental Results

In this section, we conduct experiments to compare our temporal decomposition algorithm with the commercial software IBM ILOG CPLEX 12.2 on solving the ECS/GECS problems.

6.1. Dataset

Richardson et al. [36] collected 22 dwellings' power consumption over two years in East Midlands, U.K. In the real dataset, each of the 22 smart meters is associated with 33 appliances with 1,051,200 readings (one reading per minute). We randomly select $p \in [1, 10]$ smart meters (each of which is an agent) and aggregate the readings for every 10 min within two months to generate our experimental data: $2 \times 30 \times 24 \times \frac{60}{10} = 8640$ time slots can be generated (10 min each).

Then, our experiments are conducted on mixed sets of real and synthetic data. Each appliance's consumption amount within any single time slot $\forall e_{ij}$ and total number of in-use time slots $\forall c_{ij}$ are available in the dataset. On the other hand, we randomly select 30%–50% of the appliances as appliances with an adjustable schedule, where the schedule ranges are randomly generated from all of the n time slots. For instance, an appliance (which has adjustable schedule) runs $c_{ij} = 2 \times 6$ time slots (2 h) out of $n = 24 \times 6$ time slots (24 h). We randomly generate a subset of time slots (rather than all of the time slots in 24 h), as the appliance's in-use time slots range specified by the agents with their preferences, e.g., the time slots in the first 8 h. Note that such a synthetic schedule range is expanded from the appliance's in-use time (randomly for 5–10 times) in the real data. In the GECS problem, the range of in-use time slots number $[a_{ij}, b_{ij}]$ is randomly generated with the criterion $\forall i, \forall j, a_{ij} \leq c_{ij} \leq b_{ij}$, where c_{ij} is available in the dataset.

6.2. Settings

We implemented the Mixed-Integer Programming (MIP) solver, IBM ILOG CPLEX (Version 12.2), using MATLAB (version R2015a). In addition, we coded our TD algorithm using the same software MATLAB and also invoked CPLEX while solving any decomposed MIP subproblem. Thus, the comparison between the pure CPLEX solver and our TD algorithm was done in the same coding environment. We denote directly solving the MIP problems using CPLEX as "CPLEX" and our temporal decomposition algorithm as "TD" (in which all of the decomposed subproblems are solved by CPLEX), respectively.

In both ECS and GECS problems, each variable has three different dimensions: $p \in [1, 10]$ (number of agents), $m_1, \ldots, m_p \in [1, 33]$ (number of appliances held by each agent) and $n \in [1, 8640]$ (number of time slots). We test the results in three groups of experiments as below:

- Group 1 (small/medium): testing the accuracy in case that CPLEX can find the exact optimal solutions within reasonable time. Then, (p, m_i, n) is specified as $(1, 20, 12)$, $(1, 20, 24)$, $(2, 20, 12)$, $(2, 20, 24)$, $(5, 5, 12)$, $(5, 5, 24)$, $(5, 5, 48)$, $(5, 10, 12)$, $(5, 10, 24)$ and $(5, 15, 12)$, respectively.
- Group 2 (large): testing the efficiency/scalability and accuracy on varying number of time slots and the number of appliances per agent. Fixing $p = 3$ agents, each agent has 11, 12, 13, ..., 30 appliances, and the number of time slots varies from 1500–6000. The largest MIP problem in this group includes 540,000 binary variables.
- Group 3 (large): testing the efficiency/scalability and accuracy on the varying number of time slots and the number of agents. Fixing the number of appliances held by each agent as 20, the number of agents varies in the range [1, 10], while the number of time slots varies from 1500–6000. The largest MIP problem in this group includes 1,200,000 binary variables.

We run each test for five times and average the results. Notice that only the testing cases in Group 1 could find the optimal solution within 1 h, which is a meaningful stopping point adopted in many other computational studies [37]. Indeed, in Groups 2 and 3, CPLEX failed to find the optimal solution within 5 h. For such cases, we consider the best feasible CPLEX solutions obtained within the 5-h time limit as a surrogate for the optimal solution. Note that we did try to let CPLEX run longer in many cases, but the solution quality provided by CPLEX after 6–10 h had no significant difference in its result in 5 h. If we let CPLEX run even longer, say 48 h, it occasionally could offer a slightly better solution, but the empirical error gap [37] (simply defined as $\frac{\text{TD result—CPLEX result}}{\text{CPLEX result}}$) is still mainly within

±10%. More importantly, in practice, it might be unnecessary to wait for CPLEX to provide us with a slightly better solution with greatly increased runtime.

6.3. Accuracy

In practice, we can consider the optimal solution obtained by the commercial tool CPLEX as the exact optimal solution and then compare the optimal solution returned by our algorithm to that of CPLEX. However, CPLEX can only return the optimal solution for a small or a medium size of the ECS problems (e.g., the experimental Group 1) within reasonable time. We then first look at the deviation ratios (defined in Equation (6)) of the exact solutions (by CPLEX) presented in Table 2. In these cases, our algorithm (TD) can return optimal solutions extremely close to the exact optimal solutions obtained by CPLEX: out of 10 pairs of results, seven pairs are identical, and CPLEX performs slightly better in three pairs.

For large-scale ECS problems, we plot two algorithms' deviation ratios in the experimental Groups 2 and 3 in Figure 4 and conclude the following observations. The deviation ratio decreases as the problem size increases with greater p and/or greater $m_1, \ldots, m_p$ and/or greater n, since it is more likely to further level the overall consumptions in different time slots when more agents and/or more appliances (with adjustable running schedules) and/or more time slots are involved in the scheduling. In these two groups of experiments, the number of variables falls into [30,000, 1,200,000]. Since CPLEX could not find the optimal solution in 5 h, the returned best feasible solution by CPLEX is slightly worse than the near-optimal solution returned by our TD algorithm, as shown in Figure 4.

Convergence of deviation: As shown in Figure 5, we plot the normalized deviation (defined in Equation (7)) of some selected iterations in our TD algorithm, applied to three ECS problems with different sizes (small: 240 binary variables; medium: 3600 binary variables; large: 24,000 binary variables). We can observe the convergence of the deviation minimization process from one to a small number close to zero as below.

Accuracy vs. total number of appliances: Notice that, in the ECS problems, if multiple agents are involved in the scheduling ($p > 1$), they can communicate with each other to schedule their appliances to flatten the overall power consumption. Therefore, given a fixed number of time slots for scheduling, the performance of the accuracy is dependent on the number of overall appliances, regardless of the number of agents and the numbers of appliances held by each agent. This also applies to the GECS problems.

Table 2. Small/medium-scale ECS problem (Group 1): TD vs. CPLEX.

(p, m_i, n)	# of Binary Variables	Deviation Ratio (%)		Runtime (s)	
		CPLEX	TD	CPLEX	TD
(1, 20, 12)	240	12.3	12.3	1815.12	9.27
(1, 20, 24)	480	7.09	7.09	2440.11	23.26
(2, 20, 12)	480	6.83	6.83	2621.88	21.51
(2, 20, 24)	960	2.27	2.27	3275.5	58.34
(5, 5, 12)	300	10.36	10.41	317.65	12.31
(5, 5, 24)	600	6.11	6.18	1216.44	34.57
(5, 5, 48)	1200	1.83	1.83	2351.01	37.82
(5, 10, 12)	600	8.44	8.52	1490.31	33.75
(5, 10, 24)	1200	2.67	2.67	3215.67	68.19
(5, 15, 12)	900	3.01	3.01	2903.87	52.78

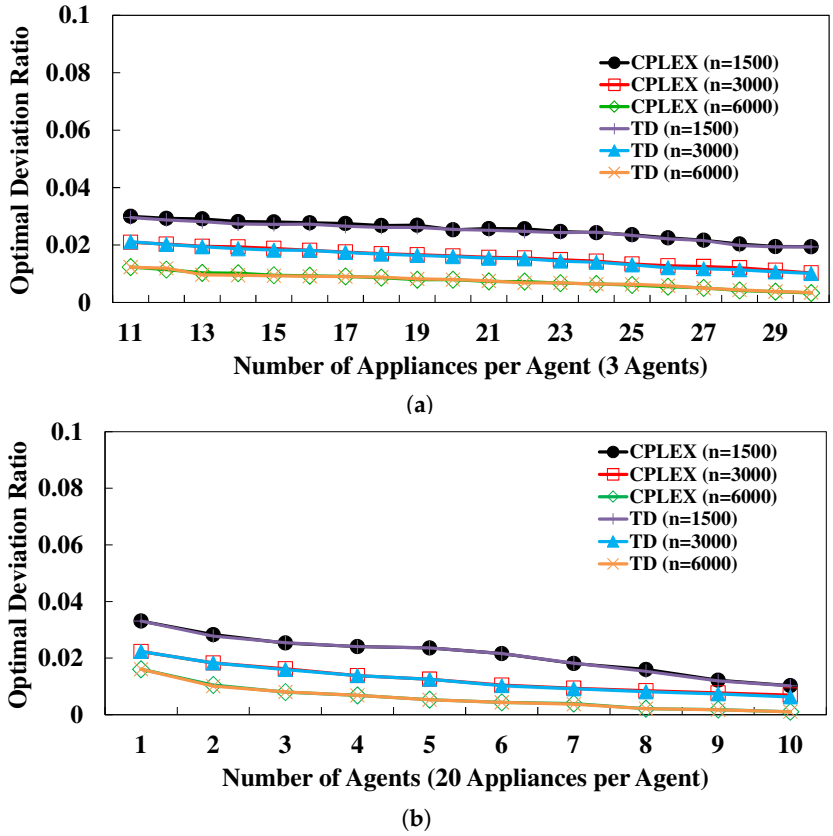

Figure 4. ECS problem (Groups 2 and 3): TD vs. CPLEX (accuracy). (**a**) Group 2 (up to 540,000 variables); (**b**) Group 3 (up to 1,200,000 variables).

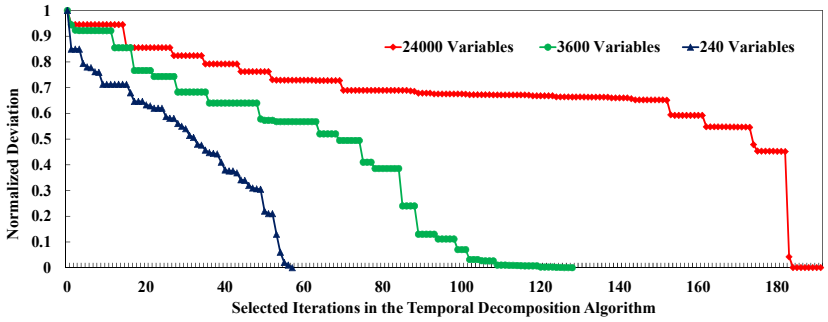

Figure 5. Normalized deviation vs. iterations (temporal decomposition).

6.4. Case Study

Besides the experimental results of deviation, we also conduct a case study to demonstrate the effectiveness of load leveling via our energy consumption scheduling approach. The power consumption data are selected from the dataset collected by Richardson et al. [36] in the U.K. We select a sample house from the 22 houses in the dataset, which includes 30 electric appliances. Out of all of the appliances, we study the load leveling in two cases: (1) all of the appliances can have adjustable schedules; and (2) only 20 appliances can have adjustable scheduling. The power consumption in the dataset has been aggregated from 1 min per reading to 15 min per reading, then we have 15 min as the time slot length for scheduling.

In the experiments, we study the scheduling problems over two time ranges [12:00 a.m.–12:00 p.m.] (midnight to noon) and [12:00 a.m.–12:00 p.m.] (noon to midnight), respectively. Thus, we have $4 \times 12 = 48$ time slots in each scheduling problem. After solving the four ECS problems by our

TD algorithm (two time ranges and two cases of appliances' schedules), we demonstrate the results in Figure 6a,b, respectively. In both figures, "ECS (30)" and "ECS (20)" represent the ECS problem (solved by the TD algorithm) with all 30 appliances involved in the scheduling and with only 20 appliances involved in the scheduling, respectively. "Original" and "ideal" means the original consumption without scheduling and the ideal scheduling (consumption equals the average amount all of the time), respectively. As a result, we can have two observations: (1) the demand load (energy consumption) can be flattened by our ECS problem at different times: for both ECS (30) and ECS (20) and (2) if more appliances have adjustable schedules for the scheduling (e.g., ECS (30)), the demand load (energy consumption) curve can be flattened closer to the ideal case.

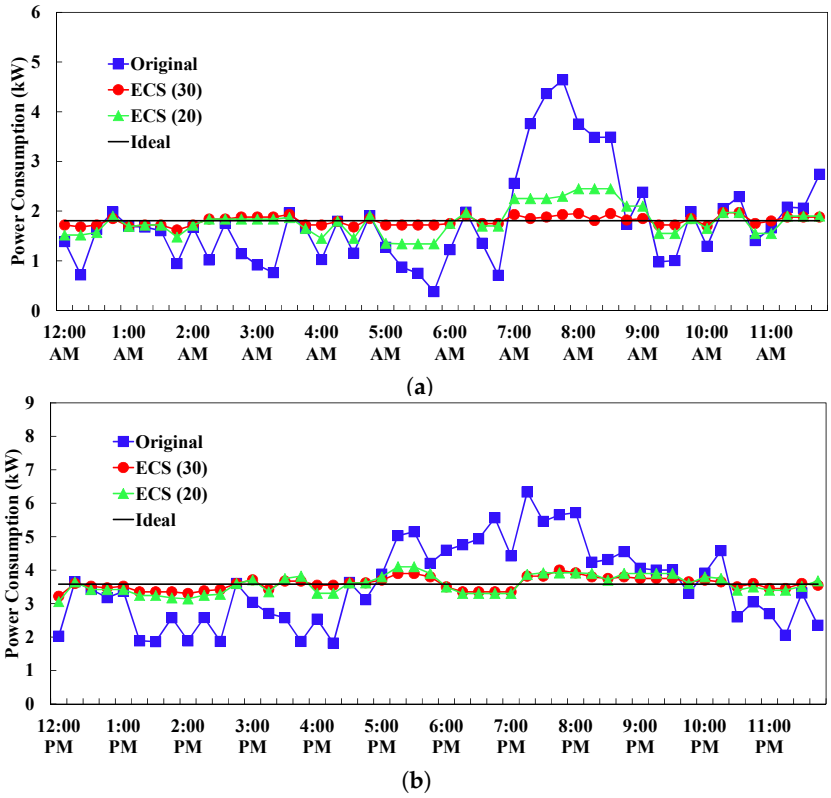

Figure 6. Load leveling (power consumption with scheduling vs. power consumption without scheduling). (**a**) 12:00 a.m.–12:00 p.m. (48 time slots); (**b**) 12:00 p.m.–12:00 a.m. (48 time slots).

6.5. Efficiency and Scalability

Figure 7 shows the runtime for two algorithms (TD and CPLEX) with varying p, $m_1,\ldots,m_p$ and n to solve large-scale ECS problems. CPLEX fails to provide optimal solutions in all of the cases within 5 h. Instead, our algorithm remarkably outperforms CPLEX on efficiency and scalability since it can return a near-optimal solution in significantly less time in almost all of the cases. As the problem size increases along three different dimensions (by increasing values of p, m_i, n), the runtime of temporal decomposition increases extremely slow with a linear trend. On the contrary, the runtime of CPLEX increases exponentially as $p, m_1,\ldots,m_p$ and/or n increases, as shown in Table 2 (note that the runtime of CPLEX in Figure 7 exceeds 10,800 s: 5 h in all of the cases in experimental Groups 2 and 3, then CPLEX is terminated at that point).

In our TD algorithm, for each testing case, the number of times of invoking IBM ILOG CPLEX is fixed (which is the number of time slots n and also the number of subproblems). In other words, the convergence against the solving step number (based on invoking CPLEX) is fixed for every scheduling problem. In the meantime, each time when CPLEX is invoked, the overhead time of interfacing is less than 0.00001 s. Even with large-scale problems, such as $n = 8640$, the total overhead time of interfacing

is still less than 0.1 s, which can be negligible comparing to the overall solving time in those testing cases. Furthermore, the runtime required for solving each subproblem in sequence decreases as the time slot number k increases from 1–n. Take the testing case $(p = 5, m_i = 10, n = 12)$ as an example, solving 12 subproblems using CPLEX requires 5.92, 4.78, 3.89, 3.38, 2.79, 2.45, 2.29, 2.02, 1.85, 1.54, 1.48 and 1.33 s, respectively. Note that the overall interfacing time takes only ~0.00012 s.

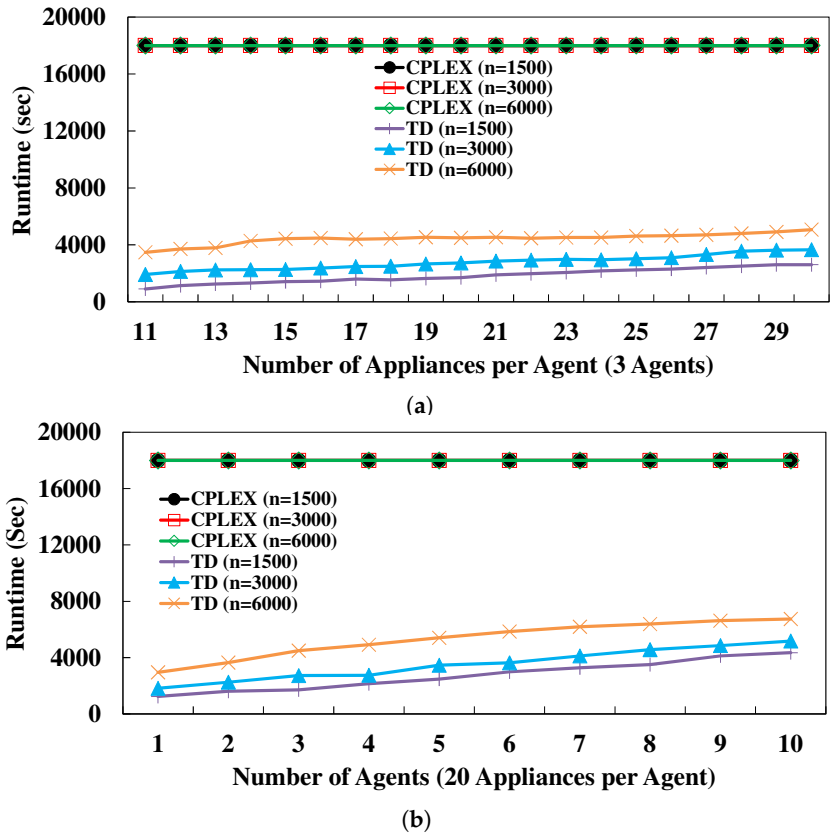

Figure 7. ECS problem (Groups 2 and 3): TD vs. CPLEX (runtime). (**a**) Group 2 (up to 540,000 variables); (**b**) Group 3 (up to 1,200,000 variables).

The highly efficient and scalable feature of our TD algorithm is more practical and accessible on the smart grid since scheduling should be implemented online among multiple agents and accomplished in a very short time. Indeed, online scheduling cannot wait for a couple of days to derive the optimal solution by CPLEX.

6.6. Experimental Results for the GECS Problem

For the GECS problem, we conducted another group of experiments using a similar dataset and obtained a similar set of experimental results. As shown in Table 3, we can draw similar observations for our temporal decomposition algorithm as the ECS problem. For large-scale problems (e.g., 900,000 binary variables), our algorithm only takes 13,423.81 s to obtain an accurate near-optimal solution. However, if we use CPLEX to solve the same GECS problem, the feasible solution obtained after 5 h is still worse than TD's near optimal solution (deviation ratio 0.049% vs. 0.043%).

Table 3. The Generalized Energy Consumption Scheduling (GECS) problem: TD vs. CPLEX.

(p, m_i, n)	# of Binary Variables	Deviation Ratio (%)		Runtime (s)	
		CPLEX	TD	CPLEX	TD
(5, 10, 100)	5000	3.15	2.99	18,000	734.77
(5, 20, 100)	10,000	2.89	2.82	18,000	983.54
(5, 10, 200)	10,000	2.74	2.68	18,000	889.02
(5, 20, 200)	20,000	2.35	2.36	18,000	1593.89
(10, 20, 400)	80,000	1.87	1.91	18,000	2656.12
(10, 30, 400)	120,000	1.56	1.52	18,000	3478.93
(10, 20, 800)	160,000	1.29	1.17	18,000	4348.62
(10, 30, 800)	240,000	0.98	0.93	18,000	5569.11
(15, 20, 1000)	300,000	0.524	0.491	18,000	6156.33
(15, 30, 1000)	450,000	0.235	0.227	18,000	7609.45
(15, 20, 2000)	600,000	0.116	0.114	18,000	9182.6
(15, 30, 2000)	900,000	0.049	0.043	18,000	13,423.81

7. Discussions

7.1. Appliance Categories in Scheduling

Ciabattoni et al. [38] has categorized the residential appliances based on their usage patterns: (1) continuous use appliances (e.g., refrigerator); (2) periodical use appliances without human interaction (e.g., oven and microwave); (3) periodical use appliances with human interaction (e.g., vacuum); (4) multimedia appliances; and (5) lighting. In the real world, different appliances and their usage patterns may influence the scheduling process, as well as implementing the optimal schedules.

First, some appliances may have non-adjustable schedules (e.g., continuous use appliances, such as refrigerator). In this case, when we formulate the ECS or GECS problems, the binary variables derived for the appliances will be fixed as constants based on the given range of time slots. Then, the overall consumption of all of the appliances held by different agents in every time slot will be optimized towards $\frac{\Delta}{n}$. Notice that, since our approach is based on scheduling the usage times for appliances with adjustable usage schedules, if there are too many appliances with fixed usage schedules, the optimal demand profile (time series consumption) may not be extremely close to the perfectly flattened consumption amount (but still better than the demand profile without scheduling). This is a possible limitation of our proposed approach.

Second, some appliances have adjustable schedules (e.g., periodical use appliances with human interaction, such as a vacuum), but their available time slots in the scheduling are unknown beforehand. For instance, if the optimal scheduling is derived for the next day, the energy consumer does not know when to use the vacuum in the next day. In such a case, non-intrusive load monitoring [39] can help identify the usage patterns of the appliance and suggest a range of running time for the appliance (then, a GECS problem should be formulated).

In our studied problem, we assume that the usage time or a range of usage time should be specified prior to the scheduling and try to flatten the demand load based on scheduling the existing appliances that have an estimated running time. If any agent would like to turn on an appliance that has not been involved in the scheduling, we assume that such an appliance does not affect the load significantly in this paper. If such an appliance can lead to an extremely high demand load, all of the agents can communicate with each other and implement the scheduling again immediately, since our algorithm is highly efficient to solve the optimization problem.

7.2. Number of Binary Variables and Load Leveling Performance

Recall that the agent-based ECS and GECS problems are formulated by p agents with $m_1, \ldots, m_p$ appliances, respectively (e.g., each agent represents a household). In general, as more

appliances with an adjustable schedule are involved in the scheduling (each agent holds more appliances with an adjustable schedule or more agents are involved in the scheduling), the performance of load leveling would become better with less aggregated deviation between the actual consumption and the ideal amount $\frac{\Delta}{n}$ in all n time slots. This is simply because more appliances with adjustable schedule (held by different agents) could enlarge the feasible region of optimization problem (which can be transformed to a mixed integer programming problem with linear constraints). In the experiments, we will validate this observation using both temporal decomposition algorithm and the standard solver CPLEX.

7.3. Running Multiple Times

In reality, each time slot can be either long or short, e.g., as long as one day and as short as 5 min [1]. The proposed ECS and GECS problems assume all of the appliances are continuously on in a specified number of time slots. Occasionally, an agent may intend to turn on an appliance multiple times out of the entire n time slots; for instance, in a 12-h scheduling ($n = 12$ and 1 h each time slot), if agent i plans to run the washer (its j-th appliance) for 4 h in total, but 2 h in the morning ($k \in [1, 4]$) and 2 h in the evening ($k \in [9, 12]$). We can formulate two sets of constraints for such appliance in the scheduling as $\forall k \in [1, 4], \sum_{k=1}^{4} x_{ijk} = 2$ and $\forall k \in [9, 12], \sum_{k=1}^{4} x_{ijk} = 2$. Such additional constraints can be formulated without affecting the efficiency of solving the ECS and GECS problems.

7.4. Short-Term Scheduling

Through tuning the length of each time slot to a short time (e.g., 1 min), our ECS/GECS problems and the TD algorithm can also efficiently identify the optimal schedule for appliances in a real-time manner since the performance of the algorithm is dependent on the number of time slots rather than the length of the time slots. Therefore, in short-term scheduling with real-time requirements, we can specify a reasonable number of time slots in each scheduling and iteratively execute the TD algorithm and scheduling to implement load leveling in real time.

8. Conclusions and Future Work

In this paper, we studied the agent-based ECS problem on the smart grid, which flattens the demand load for a single agent or multiple agents via scheduling the usage of agents' appliances at the demand side. We also extended the ECS problem to a more general format, the GECS problem in which each every agent's appliance can have an unknown number of in-use time slots at the scheduling stage. After mathematically modeling these two ECS problems as Mixed-Integer Programming (MIP) problems, we proposed a novel decomposition algorithm to efficiently and accurately solve them. We compared our algorithm with the standard benchmark CPLEX in experiments. As demonstrated in the experimental results (e.g., deviation ratios in the optimal solutions, rate of convergence and runtime), our algorithm is proven to be significantly more practical and accessible (highly efficient and accurate) than CPLEX.

In the future, we will extend the studies of energy consumption scheduling (ECS) problems for load leveling in two ways. On the one hand, we will try to study the bound of our temporal decomposition algorithm and theoretically examine the accuracy of our proposed efficient solver. On the other hand, while solving the ECS and GECS problems, multiple agents on the grid have to share their input data (e.g., each agent's appliances, total number of in-use time slots) and output (i.e., specific agent's usage schedule of their appliances in the optimal solution) to jointly formulate and solve the MIP-based ECS/GECS problems. Such information disclosure would explicitly compromise the consumers' privacy on the power grid [40,41]. We will explore privacy-preserving schemes [42,43] to effectively formulate and efficiently solve the ECS/GECS problem among multiple agents on the smart grid with limited disclosure [44,45]. Furthermore, we will extend the agent-based ECS problems for appliances to the entities with renewable energy sources [46,47], considering each agent and its appliances as a microgrid (which both consumes and generates electricity). Two categories of research

problems will be investigated by integrating the scheduling and microgrids. First, one or multiple agents can schedule not only consumption, but also generation for different applications, such as load leveling [21], power flow analysis and optimization [48,49]. Second, the faults in power flow and the distribution network [50–53] may lead to specific constraints in the scheduling problem. After incorporating such constraints in the ECS problems, we can propose the fault-tolerant scheduling problems for both energy consumption and generation in the context of microgrids.

Acknowledgments: This work is partially supported by the National Science Foundation under Grant No. CNS-1618221 and the FRAP-B Grant in the University at Albany, SUNY. We thank the anonymous reviewers for their constructive comments.

Author Contributions: Yuan Hong formulated the optimization models for the ECS and GECS problems; Shengbin Wang and Yuan Hong designed the TD algorithm for efficiently solving the scheduling problems; Yuan Hong and Shengbin Wang conceived and designed the experiments; Shengbin Wang and Ziyue Huang performed the experiments; Ziyue Huang analyzed the data; Shengbin Wang contributed reagents/materials/analysis tools; Yuan Hong and Shengbin Wang wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Running Appliances in Continuous Time Slots (ECS)

Per Equation (2) (the equality constraints), agent i's j-th appliance should be in-use for c_{ij} time slots out of n time slots where $c_{ij} \leq n$. Recall that $x_{ijk} = 1$ means the appliance is on in time slot k while $x_{ijk} = 0$ means the appliance is off. Then, we can represent all of the possibilities of the c_{ij} continuous time slots in Table A1:

Table A1. c_{ij} continuous time slots: $(n - c_{ij} + 1)$ different possibilities.

Time Slots	**1**	**2**	**3**	**...**	c_{ij}	$c_{ij}+1$	**...**	$n - c_{ij} + 1$	**...**	n
Possibility 1	x_{ij1}	x_{ij2}	x_{ij3}	...	$x_{ijc_{ij}}$					
Possibility 2		x_{ij2}	x_{ij3}	...	$x_{ijc_{ij}}$	$x_{ij(c_{ij}+1)}$				
Possibility 3			x_{ij3}	...	$x_{ijc_{ij}}$	$x_{ij(c_{ij}+1)}$	$x_{ij(c_{ij}+2)}$			
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Possibility $(n - c_{ij} + 1)$							...	$x_{ij(n-c_{ij}+1)}$	...	x_{ijn}

As shown in Table A1, there are $(n - c_{ij} + 1)$ different possibilities for c_{ij} continuous time slots (one possibility per row). Then, there are a set of constraints: if any variable $x_{ijk} = 1$, then all of the variables in exactly one out of all of the possibilities (one out of $(n - c_{ij} + 1)$ rows in Table A1) should be equal to one. More specifically,

- If $x_{ij1} = 1$, then $\sum_{k=1}^{c_{ij}} x_{ijk} = c_{ij}$ must hold (only one possibility). This ensures that such an appliance is continuously on in the first c_{ij} time slots. This constraint is equivalent to:

$$x_{ij1} c_{ij} \leq \sum_{k=1}^{c_{ij}} x_{ijk} \tag{A1}$$

  If $x_{ij1} = 0$, the inequality always holds (no constraint); otherwise $x_{ij1} = 1$, we have $\sum_{k=1}^{c_{ij}} x_{ijk} = c_{ij}$ (cannot be greater than c_{ij}).
- If $x_{ij2} = 1$, then $\sum_{k=1}^{c_{ij}} x_{ijk} = c_{ij}$ or $\sum_{k=2}^{c_{ij}+1} x_{ijk} = c_{ij}$ (exactly one out of two equalities) should hold since there are exactly two possibilities to form c_{ij} continuous time slots. This ensures that such an appliance is continuously on in time slots $[1, c_{ij}]$ or $[2, c_{ij} + 1]$. Indeed, if $\sum_{k=1}^{c_{ij}} x_{ijk} = c_{ij}$

holds, then $\sum_{k=2}^{c_{ij}+1} x_{ijk} = c_{ij} - 1$ and vice versa. Hence, we can combine them together: $\sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} = 2c_{ij} - 1$; and we formulate the constraint as:

$$x_{ij2}[c_{ij} + (c_{ij} - 1)] \leq \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} \tag{A2}$$

- Similarly, if $x_{ij3} = 1$, then $\sum_{k=1}^{c_{ij}} x_{ijk} = c_{ij}$, $\sum_{k=2}^{c_{ij}+1} x_{ijk} = c_{ij}$ or $\sum_{k=3}^{c_{ij}+2} x_{ijk} = c_{ij}$ (exactly one out holds). This ensures that such an appliance is continuously on in time slots $[1, c_{ij}]$, $[2, c_{ij} + 1]$ or $[3, c_{ij} + 2]$. Then, we derive the constraint as:

$$x_{ij3}[c_{ij} + (c_{ij} - 1) + (c_{ij} - 2)] \leq \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} + \sum_{k=3}^{c_{ij}+2} x_{ijk} \tag{A3}$$

- For time slot $s = 4, \ldots, n - 1$, the corresponding constraint can be formulated as:

$$x_{ijs}[sc_{ij} - \sum_{k=1}^{s}(k-1)] \leq \sum_{k=1}^{c_{ij}} x_{ijk} + \sum_{k=2}^{c_{ij}+1} x_{ijk} + \cdots + \sum_{k=s}^{c_{ij}+s-1} x_{ijk} \tag{A4}$$

- If $x_{ijn} = 1$ (last time slot), then $\sum_{k=n-c_{ij}+1}^{n} x_{ijk} = c_{ij}$ holds. This ensures that such an appliance is continuously on in the last c_{ij} time slots. Then, we can formulate the constraint as:

$$x_{ijn}c_{ij} \leq \sum_{k=n-c_{ij}+1}^{n} x_{ijk} \tag{A5}$$

In summary, there are $(n - c_{ij} + 1)$ new inequality constraints derived to ensure running agent i's j-th appliance in c_{ij} continuous time slots. The ECS problem should include all of the constraints (Inequalities (A1)–(A5)) for agent i's appliance j. Thus, for all of the agents $i \in [1, p]$, each of their appliances $\forall j \in [1, m_i]$ has such a set of constraints (Inequalities (A1)–(A5)) to ensure running such an appliance in continuous time slots (c_{ij} in total).

Appendix B. Problem Transformation (ECS)

First, the objective function can be converted (replacing every $|\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - \frac{\Delta}{n}|$ with y_k in the objective function):

$$\min: \sum_{k=1}^{n} |\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - \frac{\Delta}{n}| \iff \min: \sum_{k=1}^{n} y_k \tag{B1}$$

Then, some additional constraints must be added (ensuring that minimizing $\sum_{k=1}^{n} y_k$ can also minimize $\sum_{k=1}^{n} |\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - \frac{\Delta}{n}|$):

$$s.t. \begin{cases} \forall k \in [1, n], |\sum_{i=1}^{p}\sum_{j=1}^{m_i}(x_{ijk}e_{ij}) - \frac{\Delta}{n}| \leq y_k \\ \forall k \in [1, n], y_k \geq 0 \\ \forall i \in [1, p], \forall j \in [1, m_i], \forall k \in [1, n], x_{ijk} \in \{0, 1\} \end{cases} \tag{B2}$$

which are equivalent to:

$$s.t. \begin{cases} \forall k \in [1,n], \sum_{i=1}^{p} \sum_{j=1}^{m_i} (x_{ijk} e_{ij}) - \frac{\Delta}{n} \leq y_k \\ \forall k \in [1,n], -\sum_{i=1}^{p} \sum_{j=1}^{m_i} (x_{ijk} e_{ij}) + \frac{\Delta}{n} \leq y_k \\ \forall k \in [1,n], y_k \geq 0 \\ \forall i \in [1,p], \forall j \in [1,m_i], \forall k \in [1,n], x_{ijk} \in \{0,1\} \end{cases} \tag{B3}$$

Finally, combining the original constraints (Equation (4)) and extra constraints required for problem transformation (Equation (B3)), we can obtain the Mixed-Integer Programming (MIP) problem form for our ECS problem (Equation (5)).

Appendix C. Running Appliances in Continuous Time Slots (GECS)

For agent i's j-th appliance, again its total number of in-use time slots $x_{ij} = \sum_{k=1}^{n} x_{ijk}$ will replace c_{ij} in the ECS problem. Similar constraints (similar to the discussions in Appendix A) can be derived as below:

- If $x_{ij1} = 1$, then $\sum_{k=1}^{x_{ij}} x_{ijk} = x_{ij}$ must hold (only one possibility). This ensures that such an appliance is continuously on in the first x_{ij} time slots. This constraint is equivalent to:

$$x_{ij1} x_{ij} \leq \sum_{k=1}^{x_{ij}} x_{ijk} \tag{C1}$$

  If $x_{ij1} = 0$, the inequality always holds (no constraint); otherwise $x_{ij1} = 1$, we have $\sum_{k=1}^{x_{ij}} x_{ijk} = x_{ij}$ (cannot be greater than x_{ij}).
- If $x_{ij2} = 1$, then $\sum_{k=1}^{x_{ij}} x_{ijk} = x_{ij}$ or $\sum_{k=2}^{x_{ij}+1} x_{ijk} = x_{ij}$ (exactly one out of two equalities) should hold since there are exactly two possibilities to form x_{ij} continuous time slots. This ensures that such an appliance is continuously on in time slots $[1, x_{ij}]$ or $[2, x_{ij}+1]$. Indeed, if $\sum_{k=1}^{x_{ij}} x_{ijk} = x_{ij}$ holds, then $\sum_{k=2}^{x_{ij}+1} x_{ijk} = x_{ij} - 1$ and vice versa. Hence, we can combine them together: $\sum_{k=1}^{x_{ij}} x_{ijk} + \sum_{k=2}^{x_{ij}+1} x_{ijk} = 2x_{ij} - 1$, and formulate the constraint as:

$$x_{ij2}[x_{ij} + (x_{ij} - 1)] \leq \sum_{k=1}^{x_{ij}} x_{ijk} + \sum_{k=2}^{x_{ij}+1} x_{ijk} \tag{C2}$$

- Similarly, if $x_{ij3} = 1$, then $\sum_{k=1}^{x_{ij}} x_{ijk} = x_{ij}$ or $\sum_{k=2}^{x_{ij}+1} x_{ijk} = x_{ij}$ or $\sum_{k=3}^{x_{ij}+2} x_{ijk} = x_{ij}$ (exactly one out of three equalities holds). This ensures that such an appliance is continuously on in time slots $[1, x_{ij}]$, $[2, x_{ij}+1]$ or $[3, x_{ij}+2]$. Then, we can formulate the constraint as:

$$x_{ij3}[x_{ij} + (x_{ij} - 1) + (x_{ij} - 2)] \leq \sum_{k=1}^{x_{ij}} x_{ijk} + \sum_{k=2}^{x_{ij}+1} x_{ijk} + \sum_{k=3}^{x_{ij}+2} x_{ijk} \tag{C3}$$

- for time slot $s = 4, \ldots, n-1$, the corresponding constraint can be formulated as:

$$x_{ijs}[s x_{ij} - \sum_{k=1}^{s} (k-1)] \leq \sum_{k=1}^{x_{ij}} x_{ijk} + \sum_{k=2}^{x_{ij}+1} x_{ijk} + \cdots + \sum_{k=s}^{x_{ij}+s-1} x_{ijk} \tag{C4}$$

- if $x_{ijn} = 1$ (last time slot), then $\sum_{k=n-x_{ij}+1}^{n} x_{ijk} = x_{ij}$ holds. This ensures that such an appliance is continuously on in the last x_{ij} time slots. Then, we can formulate the constraint as:

$$x_{ijn} x_{ij} \leq \sum_{k=n-x_{ij}+1}^{n} x_{ijk} \tag{C5}$$

Similar to the ECS problem, there are $(n - x_{ij} + 1)$ new inequality constraints derived to ensure running agent i's j-th appliance in x_{ij} continuous time slots. The GECS problem should include all of the constraints (Inequalities (C1)–(C5)) for agent i's appliance j. Thus, for all of the agents $i \in [1, p]$, each of its appliances $\forall j \in [1, m_i]$ has such a set of constraints (Inequalities (C1)–(C5)) to ensure running such an appliance in continuous time slots (x_{ij} in total).

References

1. Fang, X.; Misra, S.; Xue, G.; Yang, D. Smart Grid—The New and Improved Power Grid: A Survey. *IEEE Commun. Surv. Tutor.* **2012**, *14*, 944–980.
2. Aung, Z.; Toukhy, M.; Williams, J.R.; Sanchez, A.; Herrero, S. Towards Accurate Electricity Load Forecasting in Smart Grids. In Proceedings of the 4th International Conference on Advances in Databases, Knowledge, and Data Applications, Saint-Gilles, Belgium, 29 February–5 March 2012.
3. Lin, H.Y.; Tzeng, W.G.; Shen, S.T.; Lin, B.S.P. A Practical Smart Metering System Supporting Privacy Preserving Billing and Load Monitoring. In Proceedings of the 10th International Conference (ACNS 2012), Singapore, 26–29 June 2012; pp. 544–560.
4. Chu, C.K.; Liu, J.K.; Wong, J.W.; Zhao, Y.; Zhou, J. Privacy-preserving Smart Metering with Regional Statistics and Personal Enquiry Services. In Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIACCS 2013), Hangzhou, China, 8–10 May 2013; pp. 369–380.
5. Salinas, S.; Li, M.; Li, P. Privacy-preserving Energy Theft Detection in Smart Grids: A P2P Computing Approach. *J. Sel. Areas Commun.* **2013**, *31*, 257–267.
6. Masters, G.M. *Renewable and Efficient Electric Power Systems*, 2nd ed.; Wiley: Hoboken, NJ, USA, 2013.
7. Davidoff, S.; Lee, M.K.; Yiu, C.; Zimmerman, J.; Dey, A.K. Principles of Smart Home Control. In Proceedings of the 8th International Conference on Ubiquitous Computing, Orange County, CA, USA, 17–21 September 2006; pp. 19–34.
8. Ullah, M.N.; Mahmood, A.; Razzaq, S.; Ilahi, M.; Khan, R.D.; Javaid, N. A Survey of Different Residential Energy Consumption Controlling Techniques for Autonomous DSM in Future Smart Grid Communications. *J. Basic Appl. Sci. Res.* **2013**, *3*, 1207–1214.
9. Lee, J.; Kim, H.-J.; Park, G.-L.; Kang, M. Energy Consumption Scheduler for Demand Response Systems in the Smart Grid. *J. Inf. Sci. Eng.* **2011**, *27*, 197–211.
10. Barker, S.; Mishra, A.; Irwin, D.; Shenoy, P.; Albrecht, J. SmartCap: Flattening peak electricity demand in smart homes. In Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communications (PerCom 2012), Lugano, Switzerland, 19–23 March 2012; pp. 67–75.
11. Barker, S.; Mishra, A.; Irwin, D.; Cecchet, E.; Shenoy, P.; Albrecht, J. Smart*: An Open Data Set and Tools for Enabling Research in Sustainable Homes. In Proceedings of the ACM 2012 Workshop on Data Mining Applications in Sustainability, Beijing, China, 12 August 2012.
12. Maharjan, S.; Zhu, Q.; Zhang, Y.; Gjessing, S.; Basar, T. Dependable Demand Response Management in the Smart Grid: A Stackelberg Game Approach. *IEEE Trans. Smart Grid* **2013**, *4*, 120–132.
13. Albadi, M.; El-Saadany, E. Demand Response in Electricity Markets: An Overview. In Proceedings of the 2007 IEEE Power Engineering Society General Meeting, Tampa, FL, USA, 24–28 June 2007; pp. 1–5.
14. Logenthiran, T.; Srinivasan, D.; Shun, T.Z. Demand Side Management in Smart Grid Using Heuristic Optimization. *IEEE Trans. Smart Grid* **2012**, *3*, 1244–1252.
15. Paterakis, N.G.; Erdinc, O.; Bakirtzis, A.G.; Catalao, J.P.S. Optimal Household Appliances Scheduling Under Day-Ahead Pricing and Load-Shaping Demand Response Strategies. *IEEE Trans. Ind. Inf.* **2015**, *11*, 1509–1519.
16. Paterakis, N.G.; Tascikaraoglu, A.; Erdinc, O.; Bakirtzis, A.G.; Catalao, J.P.S. Assessment of Demand-Response-Driven Load Pattern Elasticity Using a Combined Approach for Smart Households. *IEEE Trans. Ind. Inf.* **2016**, *12*, 1529–1539.
17. Liu, W.; Wu, Q.; Wen, F.; Ostergaard, J. Day-Ahead Congestion Management in Distribution Systems Through Household Demand Response and Distribution Congestion Prices. *IEEE Trans. Smart Grid* **2014**, *5*, 2739–2747.
18. Sarker, M.R.; Ortega-Vazquez, M.A.; Kirschen, D.S. Optimal Coordination and Scheduling of Demand Response via Monetary Incentives. *IEEE Trans. Smart Grid* **2015**, *6*, 1341–1352.

19. Nourai, A.; Kogan, V.I.; Schafer, C.M. Load Leveling Reduces T & D Line Losses. *IEEE Trans. Power Deliv.* **2008**, *23*, 2168–2173.
20. Patnaik, L.M.; Iyer, K.V. Load-leveling in Fault-tolerant Distributed Computing Systems. *IEEE Trans. Softw. Eng.* **1986**, *SE-12*, 554–560.
21. Safdarian, A.; Fotuhi-Firuzabad, M.; Lehtonen, M. A Distributed Algorithm for Managing Residential Demand Response in Smart Grids. *IEEE Trans. Ind. Inf.* **2014**, *10*, 2385–2393.
22. Lin, Y.; Tsai, M. An Advanced Home Energy Management System Facilitated by Nonintrusive Load Monitoring With Automated Multi-objective Power Scheduling. *IEEE Trans. Smart Grid* **2015**, *6*, 1839–1851.
23. Lu, H.; Zhang, M.; Fei, Z.; Mao, K. Multi-Objective Energy Consumption Scheduling in Smart Grid Based on Tchebycheff Decomposition. *IEEE Trans. Smart Grid* **2015**, *6*, 2869–2883.
24. Paterakis, N.G.; Mazza, A.; Santos, S.F.; Erdinc, O.; Chicco, G.; Bakirtzis, A.G.; Catalao, J. Multi-Objective Reconfiguration of Radial Distribution Systems using Reliability Indices. In Proceedings of the 2016 IEEE/PES Transmission and Distribution Conference and Exposition (TD), Dallas, TX, USA, 3–5 May 2016; p. 1.
25. Chetto, M. Optimal Scheduling for Real-Time Jobs in Energy Harvesting Computing Systems. *IEEE Trans. Emerg. Top. Comput.* **2014**, *2*, 122–133.
26. Wang, Y.; Li, K.; Chen, H.; He, L.; Li, K. Energy-Aware Data Allocation and Task Scheduling on Heterogeneous Multiprocessor Systems With Time Constraints. *IEEE Trans. Emerg. Top. Comput.* **2014**, *2*, 134–148.
27. Lin, M.; Pan, Y.; Yang, L.T.; Guo, M.; Zheng, N. Scheduling Co-Design for Reliability and Energy in Cyber-Physical Systems. *IEEE Trans. Emerg. Top. Comput.* **2013**, *1*, 353–365.
28. Ahmed, M.S.; Mohamed, A.; Homod, R.Z.; Shareef, H. Hybrid LSA-ANN Based Home Energy Management Scheduling Controller for Residential Demand Response Strategy. *Energies* **2016**, *9*, 716.
29. Cerquides, J.; Picard, G.; Rodríguez-Aguilar, J.A. Designing a Marketplace for the Trading and Distribution of Energy in the Smart Grid. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Istanbul, Turkey, 4–8 May 2015; pp. 1285–1293.
30. Nijs, F. Dynamic Capacity Control and Balancing in the Medium Voltage Grid. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Istanbul, Turkey, 4–8 May 2015; pp. 1979–1980.
31. Gomez-Sanz, J.J.; Garcia-Rodriguez, S.; Cuartero-Soler, N.; Hernandez-Callejo, L. Reviewing Microgrids from a Multi-Agent Systems Perspective. *Energies* **2014**, *7*, 3355-3382.
32. Cha, H.J.; Won, D.J.; Kim, S.H.; Chung, I.Y.; Han, B.M. Multi-Agent System-Based Microgrid Operation Strategy for Demand Response. *Energies* **2015**, *8*, 14272–14286.
33. Agrawal, P.; Kumar, A.; Varakantham, P. Near-Optimal Decentralized Power Supply Restoration in Smart Grids. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Istanbul, Turkey, 4–8 May 2015; pp. 1275–1283.
34. Strawser, D.; Williams, B.; Inam, W. A Market for Reliability for Electricity Scheduling in Developing World Microgrids. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Istanbul, Turkey, 4–8 May 2015; pp. 1833–1834.
35. Gurobi Optimization, I. *Gurobi Optimizer Reference Manual*; Gurobi Optimization Inc.: Houston, TX, USA, 2015.
36. Richardson, I.; Thomson, M.; Infield, D.; Clifford, C. Domestic Electricity Use: A high-Resolution Energy Demand Model. *Energy Build.* **2010**, *42*, 1878–1887.
37. Lei, L.; Pinedo, M.; Qi, L.; Wang, S.; Yang, J. Personnel Scheduling and Supplies Provisioning in Emergency Relief Operations. *Ann. Oper. Res.* **2015**, *235*, 487.
38. Ciabattoni, L.; Ferracuti, F.; Grisostomi, M.; Ippoliti, G.; Longhi, S. Fuzzy Logic Based Economical Analysis of Photovoltaic Energy Management. *Neurocomput* **2015**, *170*, 296–305.
39. Hart, G.W. Nonintrusive Appliance Load Monitoring. *Proc. IEEE* **1992**, *80*, 1870–1891.
40. Hong, Y.; Goel, S.; Liu, W. An Efficient and Privacy Preserving Scheme for Energy Exchange among Smart Microgrids. *Int. J. Energy Res.* **2016**, *40*, 313–331.
41. Sankar, L.; Rajagopalan, S.R.; Mohajer, S.; Poor, H.V. Smart Meter Privacy: A Theoretical Framework. *IEEE Trans. Smart Grid* **2013**, *4*, 837–846.

42. Hong, Y.; Vaidya, J.; Lu, H.; Karras, P.; Goel, S. Collaborative Search Log Sanitization: Toward Differential Privacy and Boosted Utility. *IEEE Trans. Dependable Secure Comput.* **2015**, *12*, 504–518.
43. Hong, Y.; Vaidya, J.; Wang, S. A Survey of Privacy-aware Supply Chain Collaboration: From Theory to Applications. *J. Inf. Syst.* **2014**, *28*, 243–268.
44. Hong, Y.; Vaidya, J.; Lu, H. Secure and Efficient Distributed Linear Programming. *J. Comput. Secur.* **2012**, *20*, 583–634.
45. Hong, Y.; He, X.; Vaidya, J.; Adam, N.; Atluri, V. Effective Anonymization of Query Logs. In Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM), Hong Kong, China, 2–6 November 2009; pp. 1465–1468.
46. Hong, C.M.; Ou, T.C.; Lu, K.H. Development of Intelligent MPPT (Maximum Power Point Tracking) Control for a Grid-Connected Hybrid Power Generation System. *Energy* **2013**, *50*, 270–279.
47. Ou, T.C.; Hong, C.M. Dynamic Operation and Control of Microgrid Hybrid Power Systems. *Energy* **2014**, *66*, 314–323.
48. Ou, T.C.; Tsao, T.P.; Lin, W.M.; Hong, C.M.; Lu, K.H.; Tu, C.S. A Novel Power Flow Analysis for Microgrid Distribution System. In Proceedings of the 2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA), Melbourne, Australia, 19–21 June 2013; pp. 1550–1555.
49. Ou, T.C.; Su, W.F.; Liu, X.Z.; Huang, S.J.; Tai, T.Y. A Modified Bird-Mating Optimization with Hill-Climbing for Connection Decisions of Transformers. *Energies* **2016**, *9*, 671.
50. Ou, T.C. Ground Fault Current Analysis with A Direct Building Algorithm for Microgrid Distribution. *Int. J. Electr. Power Energy Syst.* **2013**, *53*, 867–875.
51. Ou, T.C. A Novel Unsymmetrical Faults Analysis for Microgrid Distribution systems. *Int. J. Electr. Power Energy Syst.* **2012**, *43*, 1017–1024.
52. Ou, T.C.; Chuang, S.J.; Hong, C.M.; Wu, R.C.; Tsao, T.P.; Chen, C.Y. Self-Regulation Ground Faults Model for Microgrid Distribution. *ICIC Express Lett. Part B Appl.* **2015**, *6*, 3225–3230.
53. Lin, W.M.; Ou, T.C. Unbalanced distribution network fault analysis with hybrid compensation. *IET Gener. Transm. Distrib.* **2011**, *5*, 92–100.

© 2017 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (http://creativecommons.org/licenses/by/4.0/).