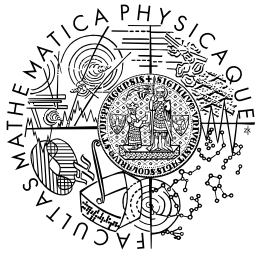

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

František Nesveda

Demand Management in Smart Grids

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis:	Mgr. Martin Pilát, Ph.D.
Study programme:	Computer Science
Study branch:	Artificial Intelligence

Prague 2019

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

I would like to express my gratitude to Mgr. Martin Pilát, PhD., for his scholarly leadership and for the time and advice he has given me during the writing of this thesis. I would also like to extend my thanks to my family and friends, who have supported and encouraged me greatly during my studies.

Title: Demand Management in Smart Grids

Author: František Nesveda

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Martin Pilát, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: With the rapid adoption of electric vehicles and the rise of power generation from renewable sources, intelligent management of power demand on a household level is gaining importance. Current algorithms used for that purpose have negative privacy implications and focus only on controlling the charging of electric vehicles while ignoring other appliances. We describe a decentralized algorithm designed to control the power demand of different types of household appliances along with the charging of electric vehicles while preserving the privacy of the subscribers. We also present a smart grid simulator to evaluate the algorithm's effectiveness along with results of simulating a scale model of the power grid of the state of Texas.

Keywords: demand management, smart grid, electric vehicles

Contents

Introduction

While the widespread adoption of electric vehicles in the future could serve to reduce our reliance on fossil fuels, provided that the energy used to charge those vehicles comes from cleaner sources, the increased electricity usage that would come as a side effect could constitute a challenge to the electrical grid [1, 2, 3, 4]. Most trips by household-owned vehicles are trips to work and back [5], and most people plug in their car to charge it as soon as they arrive home [6]. As the bulk of the people arrive home around the same time, this can cause peaks in electricity consumption in the afternoon hours.

Because traditional coal or nuclear power plants have a long ramp-up time, they are ill-suited to be used to cover these sudden peaks, and specialized peaking power plants such as gas turbines or pumped hydroelectric energy storage must be used, which is costly and not always feasible [7].

The situation is further complicated as power plants utilizing renewable energy sources are being introduced to the grid. These plants, such as solar panel farms or wind turbines, often have an uncontrollable and intermittent power output, requiring even more energy storage to compensate [7].

To combat these negative effects, there arises a need to be able to better control the electricity demand of the grid by matching the power requirements of electricity subscribers to the generation profile of the power plants.

Related work

Several strategies have been proposed to control the charging of electric vehicles to suit the power distributors.

Wu et al. [8] propose a demand control scheme where EV owners unite into aggregators who then purchase electricity at the day-ahead market, with each aggregator formulating the demands of EVs into a linear programming problem, and based on its solution decides what amount of electricity to buy in which period so as to minimize the total cost of charging the vehicles.

Vayá and Andersson [9] propose a decentralized algorithm for minimizing the total price of power generation using time-of-use electricity pricing variable at the tranmission node level, determining the optimal pricing by solving a multiperiod optimal power flow problem assembled from the charging requirements of the vehicles.

Karfopoulos and Hatziargyriou [10] propose a multi-agent system which pits the grid and the EVs against each other in a non-cooperative, dynamic game, where the objective of the EVs is to minimize their charging cost, and the objective of the grid is to minimize the peaks and valleys in the total load, adjusting the electricity prices after each round to achieve it.

Ahn et al. [11] present an algorithm to find an optimal solution to the problem of EV charging by first formulating and solving it as a linear programming problem centrally, based on estimated EV behaviour, and then mimicking the behaviour of individual EVs in the optimal solution by the actual EVs.

Gan et al. [12] propose a system for finding an optimal solution to the charging problem using an iterative algorithm to negotiate charging rates with the util-

ity provider, with the EVs deciding their charging rates based on a price signal broadcast by the utility, who based on the charging rates of the EVs iteratively updates the price signal until the charging rates converge.

In a similar vein, Li et al. [13] propose an on-line iterative algorithm coordinating the charging of EVs, this time based only on the current and past load on the grid, with the EVs again making their own decision about when to charge based on a charging reference signal iteratively calculated by the distributor, who updates the signal based on the reported charging rates of the EVs.

Rigas et al. [14] identify several key issues with the current strageties, one of them being their privacy implications, as most of them require the EV owners to inform the distributor or another third party about their travel schedule and charging needs, which some might not be willing to do. Another weak point is that the strategies control only electric cars, and do not take into account other appliances that could be controlled. Furthemore, most of the strategies require that subscribers are willing to adhere to them, even though it might not be directly beneficial to them.

Goals of the thesis

In recognition of the issues with previous solutions, we set two goals for this thesis — first, to present a demand control system which tackles these issues, specifically: protects the privacy of the subscribers, allows for control of multiple appliance types, and makes cooperation with the utility distributor beneficial to the subscribers.; and second, to develop a smart grid simulator to evaluate the performance of the presented system on real-world data.

The rest of this thesis is organized as follows: in Chapter 1 we describe the demand control algorithm we designed; in Chapter 2 we describe the smart grid simulator we created to test the performance of the algorithm; in Chapter 3 we present results of simulations we performed to evaluate our algorithm.

1. Algorithm

1.1 Overview

To fulfill the first goal we set in the previous chapter, in designing our demand control algorithm, we decided to employ the strategy of motivating the subscribers to use energy at specific times by means of time-of-use electricity price tariffs, with different timing of low and high prices for each connected household.

The electricity distributor decides on a total target electricity demand profile which it would like to achieve on its grid according to its needs (e.g. a flat demand profile for a grid consisting of power plants with slow demand-response characteristics, or a profile with higher demand during specific intervals for a grid containing solar plants or wind turbines when expecting a lot of sunlight or wind). A part of this demand does not come from subscribers whose demand can be influenced easily, but instead from other sectors like businesses and industrial customers. The part of the demand the distributor can influence, like the demand of households, can be used for balancing the demand of the rest of the grid to achieve the desired total demand. The distributor must therefore separate the total target demand profile into the uninfluenceable base demand and the influenceable target household demand. The electricity prices for each household are then assigned so that more households get a high price when their demand should be low, and a low price when their demand should be high. The electricity prices are finally distributed to the households, and the decision on when to turn their appliances on is left to them, expecting them to act in their own interest and optimize their appliance usage to achieve the lowest total cost of electricity.

This achieves the goals described earlier — privacy of the subscribers is protected, since they do not need to divulge the information about the usage of their appliances; multiple types of appliances can be controlled, since they only need to be able to adjust their operation to the electricity price; and the subscribers are incentivized to use energy in coordination with the needs of the distributor, as it will also be cheaper for them.

1.2 Requirements

To set the total target electricity demand profile for a given time interval which it would like to control, the electricity distributor needs to know the ideal power generation profile of the power plants on the grid, and the predicted total power demand of the grid for that interval, We assume that the distributor has this information available, as it is already needed to control the power generation on the grid even without using our algorithm.

The distributor also needs to know the distribution of power demand between subscribers which are taking part in the demand control scheme and the rest of the grid, to determine the base demand on the grid which the demand control scheme will then balance. This can be determined by calculating the portion of customers which are taking part in the scheme, and by analyzing their behaviour through subscriber research programs like [6].

Furthermore, we assume we have access to statistics of subscriber behaviour (when do subscribers have their EVs available for charging etc.), appliance usage (how much power do appliances need on average each day) and appliance ownership (what ratio of households owns an appliance of a given type). These can be obtained from subscriber research programs like [5] or [6].

1.3 Algorithm design

The algorithm works with the idea that households will use their appliances when electricity is cheap rather than when it is expensive. Therefore the ratio of households with a lower electricity price at a given time should depend on the desired power demand the households should have at that time, as electricity consumption will be higher at times when more households have a lower price compared to times when they have a higher price.

1.3.1 Basic algorithm

We started with a simple algorithm directly translating the target household demand into electricity prices — the higher the target demand is in a given time slot, the more households will get cheap electricity in that slot. This simple algorithm is shown in Algorithm 1.

Algorithm 1 The simple price assignment algorithm for a single day

INPUT:

$\tau = \{1..1440\}$ ... Time slots for which to calculate the price ratio
$TargetDemand_t, t \in \tau$ ... Target power demand of the households at time t
$Households$ ... List of households connected to the grid
$LowerPrice$ ... The lower electricity price
$HigherPrice$ ... The higher electricity price

ALG:

for each $t \in \tau$ **do**
$PriceRatio_t \leftarrow \frac{TargetDemand_t}{max_{t' \in \tau}(TargetDemand_{t'})}$
end for
for each $Household$ in $Households$ **do**
for each $t \in \tau$ **do**
$r \leftarrow$ random number between 0 and 1
if $priceRatio_t < r$ **then**
$price_t \leftarrow LowerPrice$
else
$price_t \leftarrow HigherPrice$
end if
end for
SetPrices($Household, price$)
end for

1.3.2 Improvements

The simple algorithm is very crude, and it does not take into account subscriber behaviour and needs. For instance, electricity price is not the only influence on the power demand, as the demand will also be higher at times when more appliances are being used (e.g. at night when EVs are at home, plugged in), compared to times when the appliances are unavailable (during the day when the EVs are away). This can cause further imbalances in the demand, as not all of the households which get lower electricity prices in a given time slot will be able to utilize that lower price, and will thus make the total demand lower than desired. To mitigate that, more households should get a lower electricity price when less appliances are available for usage, to encourage those appliances that are available to cover for those which are unavailable.

To be able to take appliance availability into account, we can utilize the subscriber research programs described in Section 1.2. From the data gathered by those programs, we can calculate the information needed for our purposes — the portion of households which own an appliance of a given type (e.g. what portion of households own an air conditioning unit), the ratio of appliances of a given type which are available for usage at a given time (e.g. how many EVs are plugged in at home at a given time), and the average power usage of an appliance of a given type during a given interval (e.g. how much power does an average freezer use per day). The processing of statistics from the subscriber research programs is described in Algorithm 2.

Algorithm 2 Processing statistics for a single day

INPUT:

$\tau = \{1..1440\}$ ... Time slots for which to process statistics

$Households$ ... The households for which statistics are available

$ApplianceTypes$ ... The types of appliances for which to process statistics

$\textsc{Available}(h, a, t)$ ... A relation determining if appliance a of household h is available for usage at time t

$\textsc{Owns}(h, a)$ ... A relation determining if household h owns an appliance of type a

$\textsc{Demand}(h, a, \tau)$ ... Function determining how much power did appliance a of household h use during the time interval τ

ALG:

for each a in $ApplianceTypes$ **do**

$H_o \leftarrow \{h \in Households \mid \textsc{Owns}(h, a)\}$

$OwnershipRatio_a \leftarrow \dfrac{|H_o|}{|Households|}$

for each $t \in \tau$ **do**

$ApplianceAvailabilityRatio_{a,t} \leftarrow \dfrac{|\{h \in H_o \mid \textsc{Available}(h, a, t)\}|}{|H_o|}$

end for

$ApplianceTypeDemand_{a,\tau} \leftarrow \dfrac{\sum_{h \in H_o} \textsc{Demand}(h, a, \tau)}{|H_o|}$

end for

Subscribers can also take issue with the way electricity prices are assigned to them. First, if the price changes too often, many appliances based around charging a battery, like electric vehicle chargers, could turn on and off too frequently to always charge only during the intervals with lower prices, potentionally leading to faster deterioration of their batteries due to charging in too short bursts. One option to solve this issue would be to lower the resolution of the algorithm, to change the prices with a lower frequency than one minute (e.g. every hour), to enable these appliances to charge for longer uninterrupted intervals. Unfortunately, this would induce spikes in energy usage at the start of each interval, as a large group of subscribers would switch from a high price to a low price, leading them all to turn their appliances on simultaneously, causing them to all start drawing power at the same time. To avoid that, we keep the one minute resolution of the algorithm, but assign the lower electricity prices in longer, continuous batches (e.g. one hour long) that do not need to start at regular intervals. Second, subscribers may be troubled with having an uncertain amount of cheaper electricity every day, as due to the randomness of assigning prices, some subscribers could have considerably less cheaper electricity on some days than other subscribers, leading to inequality concerns and other social issues. To alleviate these concerns, we incorporate a guarantee of a minimal daily amount of cheaper electricity (e.g. at least eight hours per day) into the price assignment algorithm.

1.3.3 Final algorithm

The algorithm is separated into two parts: determining the ratio of households which should get a lower electricity price at a given time, and assigning the actual electricity prices to the households based on that ratio.

In the first part, the price ratio calculating algorithm takes as input the target demand profile of the households it is controlling, as set by the distributor; the ownership ratios of each of the appliance types that can be controlled, as estimated from statistics of subscriber behaviour; the total predicted power demand by each of the appliance types, as estimated from statistics of appliance usage; and a predicted ratio of appliances of a given type available for usage at a given time, as estimated from statistics of subscriber behaviour. The sum of the target demand should be the same as the sum of total predicted demand of all the controlled appliances.

The algorithm works with an one minute resolution, separating a day into 1440 slots, for each of which the price ratio is calculated.

The algorithm first calculates the total appliance availability ratio for each of the time slots as a weighted average of the estimated availability ratios of each of the controlled appliance types, weighted by the fraction by which each of them is expected to contribute to the total target demand.

Next, the target demand profile is inversely scaled by this total appliance availability ratio, to combine the influences of both the target demand and the availability of the appliances. The resulting scaled target demand profile is then divided by its maximum to scale it to a [0, 1] range, and set as the desired ratio of households which should get a lower electricity price at a given time.

Algorithm 3 shows the price ratio calculation algorithm for a single day.

Algorithm 3 The price ratio calculation algorithm for a single day

INPUT:

$\tau = \{1..1440\}$ ... Time slots for which to calculate the price ratio

$ApplianceTypes$ ... List of appliance types for which statistics are available

$OwnershipRatio_a, a \in ApplianceTypes$ ... The portion of households that own an appliance of type a

$ApplianceAvailabilityRatio_{a,t}, a \in ApplianceTypes, t \in \tau$ ... The ratio of appliances of type a available for usage at time t

$ApplianceTypeDemand_{a,\tau}$ ... Predicted total demand of all appliances of type a for the calculated interval

$TargetDemand_t, t \in \tau$ ... Target electricity demand at time t

ALG:

for each $t \in \tau$ **do**

$TotalAvailabilityRatio_t \leftarrow 0$

for a in $ApplianceTypes$ **do**

$TotalAvailabilityRatio_t \mathrel{+}=$
$OwnershipRatio_a \times ApplianceAvailabilityRatio_{a,t}$
$\times ApplianceTypeDemand_{a,\tau}$

end for

$$UnscaledPriceRatio_t \leftarrow \frac{TargetDemand_t}{TotalAvailabilityRatio_t}$$

end for

for each $t \in \tau$ **do**

$$PriceRatio_t \leftarrow \frac{UnscaledPriceRatio_t}{\max_{t' \in \tau}(UnscaledPriceRatio_{t'})}$$

end for

return $PriceRatio$

In the second part, the algorithm assigns electricity prices to the households based on the previously calculated price ratio. The algorithm again works with an one minute resolution, separating a day into 1440 slots, for each of which the electricity prices will be calculated.

Since the subscribers expect a guaranteed minimal amount of cheaper electricity per day, and the electricity prices not to change too rapidly, we assign to the households electricity prices that are stable in longer intervals (e.g. at least 60 minute intervals of cheaper electricity), ensuring that each household gets at least a set amount of time having cheaper electricity (e.g. at least 480 minutes per day), while also respecting the previously calculated ratio of households with cheaper or more expensive electricity at a given time.

The price assignment algorithm takes as input a list of households for which to assign the price; the previously calculated price ratio for each minute of the processed day; the minimum number of minutes per day for which each household will get a cheaper electricity price; the minimum length of each continuous interval of cheaper electricity; and the electricity prices during the cheaper and during the more expensive intervals.

The algorithm assigns prices to each household separately. The price assignment process for a household starts with the electricity price high during the whole day, iteratively adding randomly generated cheaper intervals until

there are enough cheaper minutes during that day to satisfy the specified requirements. The cheaper intervals are generated so that their midpoint is selected randomly from the available time slots, with the probability distribution of the slots corresponding to the previously calculated price ratio.

Algorithm 4 shows the price assignment algorithm for a single day.

Algorithm 4 The price assignment algorithm for a single day from the precalculated price ratio

INPUT:

$\tau = \{1..1440\}$ ... Time slots for which to calculate prices
$Households$ ... List of households connected to the grid
$PriceRatio_t, t \in \tau$ ... The previously calculated ratio of households which should get a lower electricity price at time t
$CheapMinuteCount$ ... The minimum number of minutes per day for which each household will get a cheaper electricity price
$CheapIntervalLength$ ... The length of each cheaper interval in minutes
$LowerPrice$ ... The electricity price during the cheaper intervals
$HigherPrice$ ... The electricity price during the more expensive intervals

ALG:

for each $t \in \tau$ **do**
$Probabilities_t \leftarrow \frac{PriceRatio_t}{\sum_{t' \in \tau} PriceRatio_{t'}}, t \in \tau$
end for
for each $Household$ in $Households$ **do**
for each $t \in \tau$ **do**
$cheap_t \leftarrow 0$
end for
while $\sum_{t \in \tau} cheap_t < CheaperMinuteCount$ **do**
$intervalMid \leftarrow$ random slot from τ with distribution $Probabilities$
$startPos \leftarrow \text{MAX}(\text{MIN}(\tau), intervalMid - \lceil CheapIntervalLength/2 \rceil)$
$endPos \leftarrow \text{MIN}(intervalMid + \lfloor CheapIntervalLength/2 \rfloor, \text{MAX}(\tau))$
$cheap_t \leftarrow 1, t \in \{startPos..endPos\}$
end while
for each $t \in \{t \mid cheap_t = 0\}$ **do**
$price_t \leftarrow HigherPrice$
end for
for each $t \in \{t \mid cheap_t = 1\}$ **do**
$price_t \leftarrow LowerPrice$
end for
SetPrices($Household, price$)
end for

2. Simulator

For evaluating the effectiveness of the demand control algorithm, we created a simulator of the smart grid. Its structure was designed to resemble a real electrical distribution grid, with each part of the grid being a separate entity, operating independently, keeping its own clock and performing calculations exactly when needed, according to an environment-provided clock signal. This was done to allow for easier possible future extension of the simulator or its separation into multiple programs communicating over some network interface to enable simulating communication delays and other physical limitations of the distribution grid.

The simulator implements the algorithm described in chapter 1 and two simpler demand control algorithms used for comparison with the first one — an algorithm where appliances stretch their power demand across their whole usage interval, and an algorithm where they perform their operation as early as possible.

The simulator works with data from the Pecan Street organization [6] and the National Household Travel Survey [5].

2.1 Program structure

2.1.1 Overview

The simulator is a Python application consisting of several classes and modules, the main ones being:

Simulator The main class of the program, which creates the grid and houses, connects them together, then runs the simulation and optionally writes the results to disk.

Grid Represents the electrical grid. Takes care of estimating the base demand, calculating the target demand and price ratios, and collecting the resulting demands of the houses.

Connection Represents the connection between the grid and a house. Takes care of selecting the cheap price intervals for the connected house and sending it the resulting price profile, and provides a general communication interface between the grid and the house.

House Represents a house connected to the grid. Contains several appliances, and takes care of directing their simulation.

Appliance An abstract class representing a household appliance or machine. Takes care of calculating its electricity usage according to several optimization behaviours.

There are currently three categories of appliances available for simulation:

Battery An appliance category whose power draw comes from charging a battery, which has a negligible power draw when not being used, and when used it cannot be plugged in. Currently the only simulated appliance of this category is an electric car.

Accumulator An appliance category that works by accumulating energy, which then gets used gradually over time, while the appliance is still connected to electricity. The simulated appliances of this category are air conditioning, electrical heating, refrigerators and storage water heaters.

Machine An appliance category which can be turned on at any point during a specific interval, must finish by some time limit, and once turned on runs its predefined program. The simulated appliances of this category are dishwashers and washing machines.

There are also several modules used as an interface to access the data about appliance usage and grid demand records:

applianceStatistics A module containing several classes providing information about the characteristics and usage statistics of various appliances.

gridStatistics A module providing access to the demand records of the model electrical grid.

The simulator also heavily utilizes the `Profile` class implementing a time-indexable array of float values for representing demand profiles and other time series and containing special methods for manipulating those series.

2.1.2 Simulator

The main class of the program, which contains only one static method, *run*, used for executing the simulation.

run Creates a `Grid`, an user-defined number of `House`s and connects them together using `Connection`s. Then sets the current date and time in the `Grid` and `House`s to the user-defined starting datetime of the simulation. Then, according to the user-defined length of the simulation, repeatedly calls the `tick()` method of the `House`s and the `Grid` to tell them to move forward one day in time and perform all the necessary operations for that day. After the required number of days is simulated, it collects the recorded demands and the electricity price ratio from the `Grid`, optionally saves those results to a user-defined CSV file, and sets the results as its return value.

2.1.3 Grid

A class representing the smart grid. Takes care of estimating the base demand, calculating the target demand and price ratio, and collecting the resulting demands of the houses. Its main methods are:

setUp Prepares the grid for the simulation — sets the correct time, predicts the base demand and calculates the target demand and price ratio for a long enough interval in the future for the simulation to run.

tick Performs all the necessary actions of the grid for one day of the simulation — collects the simulation results calculated by the households for the previous

day, predicts one more day of the base demand, calculates one more day of the target demand and the price ratio, and distributes the price ratio to the `Connection`s to the houses.

predictBaseDemand Predicts the power demand of the rest of the grid, without the connected houses, for a specified interval. Currently just substracts the expected demand of the households from the estimated demand of the whole grid.

calculateTargetDemand Calculates the power demand the connected houses should try to achieve during a specified interval. Currently just smooths out the base demand by interpolating between its peaks and then scales the result according to the expected demand of the households. In a real grid this would be calculated to match the expected output by the power plants.

calculatePriceRatio Calculates the cheap price ratios for a speficied interval based on the target demand and appliance availability statistics according to the algorithm described in chapter 1.

distributePriceRatios Distributes the calculated price ratios to all the `Connection`s to houses.

collectDemands Collects the power demands from all the connected households and stores them.

2.1.4 Connection

A class representing a connection between the electric grid and a household. Serves mostly as an intermediary between the `Grid` and the connected `House`, facilitating the exchange of electricity prices and demand information. Its only real purpose is to calculate the actual electricity price for the connected `House` based on the price ratio supplied by the `Grid`.

The price generation is separated into two parts — generating random intervals of a predefined length during which the electricity price will be lower according to the price ratio provided by the grid, and setting those prices based on the intervals. These two actions are separated to enable easier overlapping of lower price intervals at the edges of simulated days.

The main methods of the class are:

setUp Prepares the connection for the simulation — sets the correct time, generates the electricity prices for the first two days of the simulation and sends those prices to the connected house.

tick Performs all the necessary actions of the connection for one day of the simulation — generates the electricity prices for one more day of the simulation and sends them to the connected house.

generateRandomCheaperIntervals For a given time range, generates the intervals in which the electricity price will be lower, according to probabilities given by the price ratio for that time range. These can reach up to one hour over the edges of the given time range, to enable their distribution to correspond to the price ratio.

generatePriceProfile Based on the intervals of lower price generated earlier, generate the actual electricity prices.

sendPriceProfile Send the generated electricity prices to the connected house.

2.1.5 House

A class representing a household connected to the simulated grid. Serves as a container of appliances, forwarding the electricity prices set by the grid to the household to them, and aggregating their electricity demand to be retrieved by the grid. The class also has a static method `House.random()`, serving to create a `House` instance containing random appliances generated according to the appliance ownership statistics specified in `applianceStatistics.ownershipRatios`.

2.1.6 Appliance

`Appliance` is an abstract class representing household appliances and devices, like electric cars, water heaters, dishwashers and similar. It contains methods for simulating their operation using different optimization algorithms, namely:

setUp Prepares the appliance for the simulation — sets the correct time and generates random information about the usage of the appliance for the first day of the simulation.

tick Performs all the necessary actions of the appliance for one day of the simulation — generates random usage for one more day, calculates the power demand according to each of the demand optimization algorithms, and moves the appliance clock one day ahead.

generateUsage Generates random usage for the appliance for a specified time interval based on usage statistics information contained in its `usageStatistics` member variable.

calculateSmartDemand Calculates the power demand the appliance would have in a specified interval if it tried to minimize the total electricity cost according to the price profile for the household, and saves it in its `smartDemand` member variable.

calculateUncontrolledDemand Calculates the power demand the appliance would have in a specified interval if it tried to perform its function as early as possible, and saves it in its `uncontrolledDemand` member variable.

calculateSpreadOutDemand Calculates the power demand the appliance would have in a specified interval if it tried to spread out its electricity demand over the whole interval its available for usage, and saves it in its `spreadOutDemand` member variable.

Appliance categories

There are three main appliance categories, each performing a different function.

Battery An abstract class for battery-based appliances or devices, e.g. an electric car. This category of appliance has only one parameter, its charging power.

Overview The class represents an appliance with a battery that needs to be recharged after every use. The owners of the appliance use it at most once a day during some usage interval, and the rest of the day the appliance is available for charging. For simplification of the simulation, the appliance always charges using its maximum charging power. For each simulated day a connection and disconnection time is randomly generated, along with the amount of energy needed to recharge the battery after it is connected. The simulation then decides when will the battery be charged.

Usage generation When generating random usage for a specified interval, based on the data from the appliance type's usage statistics, a random usage interval (a disconnection and connection time) and a random needed charge amount are generated for each day of the specified interval. For simplification of the simulation, it is assumed that the usage interval represents the time between the first appliance disconnection of the day and the last connection of the day, and that the usage interval always ends on the same day it began, therefore the appliance is always available for charging over midnight. To simulate an usage interval that spans multiple days, it is possible to split the usage interval to multiple intervals covering only a single day, and set the required charge to zero on all but the last of those intervals.

Demand calculation Each day of the simulated interval, the appliance is considered available for charging between the end of its usage interval on that day and the start of its usage interval on the next day. If the appliance would not be able to be charged in time for its next usage even when charging with the maximum available power, the appliance will charge for the whole availability interval. Otherwise, the appliance calculates the demand using each of the implemented simulation algorithms.

When simulating operation with the intention of minimizing the total electricity cost, the appliance determines the amount of minutes it would need to charge for so that it would get charged fully, and then chooses that many minutes from the availability interval, choosing the cheapest minutes available.

When simulating operation with the intention of charging the battery fully as early as possible, the appliance simply starts charging as soon as it is connected and charges until fully charged.

When simulating operation with the intention of spreading out its electricity demand over the whole availability interval, the appliance simply draws a constant amount of electricity each minute of its availability interval, so that at the disconnection time it is fully charged.

Accumulator An abstract class for appliance types which work by accumulating energy in some medium, e.g. electrical household heatings or refrigerators. This category of appliances has two parameters, its charging power and its energy capacity.

Overview The appliance represents a system of energy storage, with the energy typically stored in the form of heat. The amount of energy stored is kept within some limits (i.e. lower and upper temperature limits on a thermostat), the difference between which is represented by the energy capacity parameter. During the run of the appliance, the stored energy slowly escapes from the system (i.e. through heat dissipation), requiring more energy to be put in to keep within the limits.

Usage generation When generating random usage for a specified interval, a random discharging profile (how much energy escapes the system each minute of the interval) is generated for that interval based on the data from the appliance type's usage statistics. The statistics reflect the different rates of energy loss during different periods (a house loses heat faster during winter than during summer).

Demand calculation When simulating the operation of the appliance, the amount of energy stored in the appliance must never fall below zero and never exceed its capacity (i.e. the temperature in a house never falls below some lower limit and never exceeds some upper limit specified by the inhabitants). For simplification of the simulation, it is assumed the amount of energy stored in the appliance is desired to be constant (i.e. a stable temperature in a house or a refrigerator) and that the amount of energy lost at any given minute does not depend on the amount of energy stored in the appliance, or that this dependence is negligible.

When simulating operation with the intention of minimizing the total electricity cost, the appliance chooses the minutes in which the appliance should be charging its accumulator using the greedy algorithm described in [15]. We use the non-optimized, quadratic-time version of the algorithm, because due to its implementation utilizing NumPy optimizations it is executing faster than its asymptotically faster versions utilizing prefix sum trees or the Disjoint-set data structures.

Because of the algorithm's goal to minimize the electricity cost, it leaves the appliance at the end of the calculated interval with as little energy as possible. With the iterative architecture of the grid simulator, this would cause an issue where at the start of each simulated interval, the appliances would need to charge disproportionately more to compensate for the low amount of energy stored at the end the of previous interval, causing spikes in the electricity demand. To avoid that, we calculate the power demand for a longer interval than needed, and then keep only the part of the calculated demand that is needed.

When simulating operation with the intention of performing its function as early as possible, the appliance simulates the progression of the amount of energy stored, lowering the amount of energy stored each minute based on the generated usage, and whenever turning the appliance on would not cause the stored energy to exceed the appliance capacity, the appliance turns on.

When simulating operation with the intention of spreading out its electricity demand over the whole availability interval, the appliance simulates a thermostat-based charging, where any time the amount of energy stored in the appliance falls to zero, the appliance starts charging its accumulator until its fully charged again.

Machine An abstract class for appliance types which work by running a predefined program, and can be set to run at any point in a specified interval, e.g. dishwashers or washing machines.

Overview The machines simulated by this class have some program (represented by its length and by the amount of power the machine draws at each minute of the program) which needs to be run uninterrupted until it is complete. They can be set to run with a delayed start, with their owner specifying the earliest time they can start and the time they need to be finished by (typically the owner will want to run them overnight, to be finished in the morning).

Usage generation When generating random usage for a specified interval, based on the data from the appliance type's usage statistics, for each day of that interval it is randomly decided if the appliance will be used that day, and if it will, a random operation program (how much electricity does the appliance draw each minute of its operation) and a random availability interval (at what time can the appliance start at the earliest and by what time must it finish) are generated for that day.

Demand calculation If the difference between the earliest possible starting time and latest possible finishing time is not long enough for the given operation program on that day, the appliance simply runs as early as possible. Otherwise, it decides when to run according to the desired behaviour specified by the simulator.

When simulating operation with the intention of minimizing the total electricity cost, the appliance calculates the cost of electricity its operation would have for each possible starting minute in the availability interval, and chooses the cheapest of those minutes.

When simulating operation with the intention of performing its function as early as possible, the appliance simply turns on in the first minute of its availability interval.

When simulating operation with the intention of spreading out its electricity demand over the whole availability interval, the appliance simply runs in the middle of its availability interval.

Appliance types

For each appliance category, there are one or more appliance types subclassing that category's base class. These are:

Battery `Car`

Accumulator `AirConditioning`, `ElectricalHeating`, `Fridge`, `WaterHeater`

Machine `Dishwasher`, `WashingMachine`

These subclasses only pair the right appliance category with the right appliance statistics object.

2.1.7 Profile

An utility class representing a time-series of float values. Enables transparent access to the stored values with time indexing. The values are stored in a NumPy array `Profile.values`, and the datetime belonging to the first value is stored in `Profile.startingDT`. Main methods of the class are:

get Retrieves the values for a specified interval from the Profile, filling the missing values with 0.

set Puts new values to a specified interval in the Profile.

add Increases the values stored for a specified interval by specified new values.

transition Changes the values stored in the Profile for a specified interval, smoothly transitioning them to the specified new values.

prune Removes unneeded stored values up to a specified point in time, to save memory.

fromCSV Creates a Profile from a CSV file, where the first column in the file specifies the datetime of values and the second column the actual values.

2.1.8 Appliance statistics

The simulator also contains a module `applianceStatistics`, which contains data about typical appliance characteristics and usage patterns for all the simulated appliance types. This data is stored in classes `BatteryStatistics`, `AccumulatorStatistics` and `MachineStatistics` corresponding to each of the appliance categories in the simulator.

BatteryStatistics Contains statistics for appliance types of the `Battery` class. The main statistics are:

chargingPowers List of possible maximum charging powers of the appliance type

usageProbabilities Probabilities that the appliance will get used on a given day

neededCharges List of possible charges which the appliance might need after usage on a given day

averageNeededCharge Average charge that is required by the appliance type on a given day

usageIntervals Intervals in which the appliance might be used on a given day

availabilityProfile Profile of the ratio of appliances that are available for charging at any given date and time

AccumulatorStatistics Contains statistics for appliance types of the `Accumulator` class. The main statistics are:

chargingPowers List of possible charging powers for the appliance type

averageChargingPower Average charging power of the appliance type

capacityParameters Mean appliance accumulator capacity and its standard deviation

dischargingProfile The average discharging profile of the appliance type at any given date and time

averageDailyCharge Average needed charge of the appliance type for any given day

MachineStatistics Contains statistics for appliance types of the `Machine` class. The main statistics are:

startAfterParameters Mean minute of the day that the machine can start after, and its standard deviation

finishByParameters Mean minute of the day that the machine must finish by, and its standard deviation

usageProbabilities Probabilities that the machine will get used on a given day

usageProfiles Possible power usage profiles of the machine type, corresponding to the machine programs

averagePowerNeeded Average power needed by the machine type on a given day

Each of these classes also contains corresponding methods to randomly generate each characteristic based on the contained statistics, and methods to load these statistics from a file.

These classes are then each instantiated into objects for each of the appliance types, namely `carStatistics`, `airConditioningStatistics`, `electricalHeatingStatistics`, `fridgeStatistics`, `waterHeaterStatistics`, `dishwasherStatistics` and `washingMachineStatistics`.

Also contained in this module is an object `ownershipRatios` containing information about the ratio of households which own a given appliance type.

2.1.9 Grid Statistics

The simulator also contains a module `gridStatistics`, which contains data about the forecasted and actually recorded demand of some real distribution grid, currently the Texas distribution grid[1], and the average demand of a household from the Pecan Street research network.

2.1.10 Data gathering

To be able to simulate the operation of the grid, we must first obtain the data based on which the grid will be created. To that end there is a subfolder `data` inside the main `simulator` folder, in which there are several scripts for downloading and parsing the data from the Internet. The data is downloaded from the PostgreSQL database of the Pecan Street organization [6] and from the webpages of the National Household Travel Survey [5].

In the `data` folder there is a `config.txt` file containing two lines with directives to configure the time span of the data to be downloaded.

There are also three subfolders, `manual`, `dataport` and `nhts`, one for each of the data sources.

Manual contains files with data that needs to be manually filled out, those being `priceConfig.json` with configuration of the electricity prices to be used in the simulator, `ownershipRatios.json` containing configuration of the ratios of households that own appliances of a given type, and `applianceCapacities.json` containing configuration of the parameters based on which random energy capacities for random appliances are generated.

Dataport contains scripts for downloading data from the *Dataport* database of the Pecan Street organization.

The credentials to access the database are configured in the `KEY` file inside that folder.

There are five kinds of data downloaded from the *Dataport* database, each having its own subfolder in the `dataport` folder containing a script `getData.py` for downloading that data:

accumulators Data about the power usage of accumulator-based appliances like household heating systems or air conditioning units

[1] `https://www.ercot.com/`

cars Data about the charging powers and charging requirements of electric cars

machines Data about the power draw profiles of household machines like dishwashers or washing machines

ercot Data about the power demand and demand predictions of the Texas power grid

households Data about the power average power usage of households

NHTS contains scripts for downloading data from the National Household Travel Survey project.

The data is downloaded by the script `getData.py` in that folder. The script downloads the whole data archive from the NHTS website, and then parses out data about how many cars do households own, the probability of the cars' usage and the intervals in which the cars are being used. Some households own up to 12 cars, but because cars beyond the fourth car in a household tend to get used only rarely and the data about them is very sparse, we keep the data only from the first four cars in a household.

2.2 Software libraries

The simulator uses several third party libraries for its operation, namely:

NumPy[2] for memory-efficient floating-point array storage and manipulation

SciPy[3] for peak detection in various time series

Pandas[4] for easy manipulation of CSV files

There are some additional libraries used for downloading the data:

Requests[5] for enabling easier access to the web resources of the National Household Travel Survey [5]

Psycopg[6] for accessing the PostgreSQL database of the Pecan Street organization [6]

SQLAlchemy[7] for enabling interaction between the Pandas and Psycopg libraries

Additionally, Jupyter[8] and Matplotlib[9] are used for the presentation of the results of the simulation.

These libraries were chosen for their maturity, ubiquity and interoperability.

[2] `https://www.numpy.org/`
[3] `https://www.scipy.org/`
[4] `https://pandas.pydata.org/`
[5] `https://www.python-requests.org/`
[6] `http://initd.org/psycopg/`
[7] `https://www.sqlalchemy.org/`
[8] `https://www.jupyter.org/`
[9] `https://www.matplotlib.org/`

2.3 Usage

2.3.1 Requirements

The simulator can be run in an UNIX environment containing the *Bash* shell[10] and a Python[11] interpreter version *3.6* or newer.

Several third-party Python libraries are required, these are listed in the file `requirements.txt` and available for installation with the shell command `pip3 install -r requirements.txt`.

2.3.2 Data download

Before the actual usage of the simulator, the data necessary for the simulation must be downloaded first. This is performed in three steps:

1. In the file `simulator/data/config.txt`, put the desired dates in the `fromdate` and `todate` fields, in the format `YYYY-MM-DD`. For a succesful simulation, the data must be downloaded for an interval starting one week before the starting date of the simulation and ending one week after the ending date of the simulation.

2. In the file `simulator/data/dataport/KEY`, put the username and password to access the *Dataport* database in the `username` and `password` fields, respectively. These access credentials can be obtained at `https://dataport.pecanstreet.org/`.

3. Run the script `downloadData.sh` in a terminal. Depending on the length of the downloaded interval, the download can take several hours and use several gigabytes of data. Progress of the download is printed to the terminal during the run of the script.

2.3.3 Running the simulator

After the data is downloaded, the simulator can be executed by using the command `./run.py` *`startingDate simulationLength householdCount outputFolder`* in a terminal while being in the directory with the project, while replacing `startingDate` with the date from which to run the simulation in the format `YYYY-MM-DD`, `simulationLength` with the number of days to simulate, `householdCount` with the number of households to simulate and `outputFolder` with the destination folder in which the simulation results should be saved, for example: `./run.py 2018-01-01 365 10000 out/`.

While the simulation is running, the simulator prints information about its progress to the terminal.

When the simulation finishes, the results are saved in the specified folder in two files, `desc.txt` and `data.csv`. The file `desc.txt` contains information about the simulation parameters, and the file `data.csv` is a standard comma-separated values file containing the simulation results organized to eight columns:

[10] `https://www.gnu.org/software/bash/`
[11] `https://www.python.org/`

Datetime The date and time of the result

PredictedBaseDemand The predicted grid base demand at that date and time (in kilowatts)

ActualBaseDemand The actual grid base demand at that date and time (in kilowatts)

TargetDemand The target power demand at that date and time (in kilowatts)

SmartDemand The power demand of the simulated households at that date and time if the smart algorithm was used (in kilowatts)

UncontrolledDemand The power demand of the simulated households at that date and time if the uncontrolled algorithm was used (in kilowatts)

SpreadOutDemand The power demand of the simulated households at that date and time if the spread out algorithm was used (in kilowatts)

PriceRatio The ratio of how many households should have a lower electricity price at that date and time

2.3.4 Displaying the results

To display the simulation results, one can use the included Jupyter notebook `Results.ipynb`. After opening the notebook, first the folder with the simulation results must be specified in the variable `resultsFolder` in the third code cell, and then the cells in the notebook can be executed to show the results.

3. Results

Using the provided simulator, we evaluated the performace of the proposed demand optimalization algorithm (referred to as the *smart algorithm* in the results) compared to two other algorithms — a simple optimization algorithm where appliances spread out their power demand across their whole use period (referred to as the *spread out algorithm* in the results), and an algorithm without optimization (referred to as the *uncontrolled algorithm* in the results), where the appliances behave as if they were not intelligent at all and perform their operation without regard for the preferences of the distributor, which is what would happen with most appliances currently on the market.

We evaluated the algorithm on a model grid based on the state of Texas, modelled using data from the Electric Reliability Council of Texas (ERCOT) [16], the Pecan Street organization [6] and the National Household Travel Survey (NHTS) [5].

For the actual base demand of the grid, we used the actual demand of Texas as recorded by ERCOT, reduced by the demand of households on the grid inferred from data reported by the Pecan Street organization. For the predicted base demand of the grid, we used demand predictions performed by ERCOT, again reduced by the demand of households on the grid. Since we need data about the predicted demand of households, and Pecan Street provides only data about their actual demand, which does not include any prediction errors, we introduced an artificial 10% error into the data to model the error of actual predictions.

We simulated a model of the Texas grid scaled down to 10 000 households. Each household contained a set of appliances which were available for the simulator to control, ranging from electric cars through household heating and air conditioning to dishwashers and washing machines, with the appliances being assigned according to ownership ratios reported by the Residential Energy Consumption Survey [17]. The characteristics of the appliances for the households were generated based on measurements of household appliance power usage performed by the Pecan Street organization, and the usage of household vehicles was generated based on statistics of citizen commute as recorded by the NHTS, acting as if every car owned by the households was an electric one.

The target demand of the households was set as demand with a curve that smooths out the peaks and valleys in the base demand, while at the same time being high enough to cover for the predicted demand of the simulated households. The target demand was shaped with regards to the predicted base demand, therefore when applied to the actual base demand, which may differ from the predicted base demand, it is not as smooth as originally intended.

To best represent the electricity pricing on the Texas grid, based on the pricing analysis published by ERCOT, we chose 0.05 USD/kWh and 0.15 USD/kWh as the lower and higher electricity prices in the simulation, respectively.

3.1 Select intervals

We present a few select intervals from the results of the simulation of the year 2018 demonstrating the performance of the algorithm.

The algorithm works well during mild weather conditions, when the households do not need to heat or use air conditioning. Figure 3.1 shows the simulation results for 9 days in February 2018, Figure 3.2 shows results at the turn of April and May. The power demand of the households operating according to the smart algorithm has much smaller peaks and valleys and is much closer to the target demand than the power demand according to the uncontrolled algorithm, and offers a slight improvement over the spread out algorithm as well.

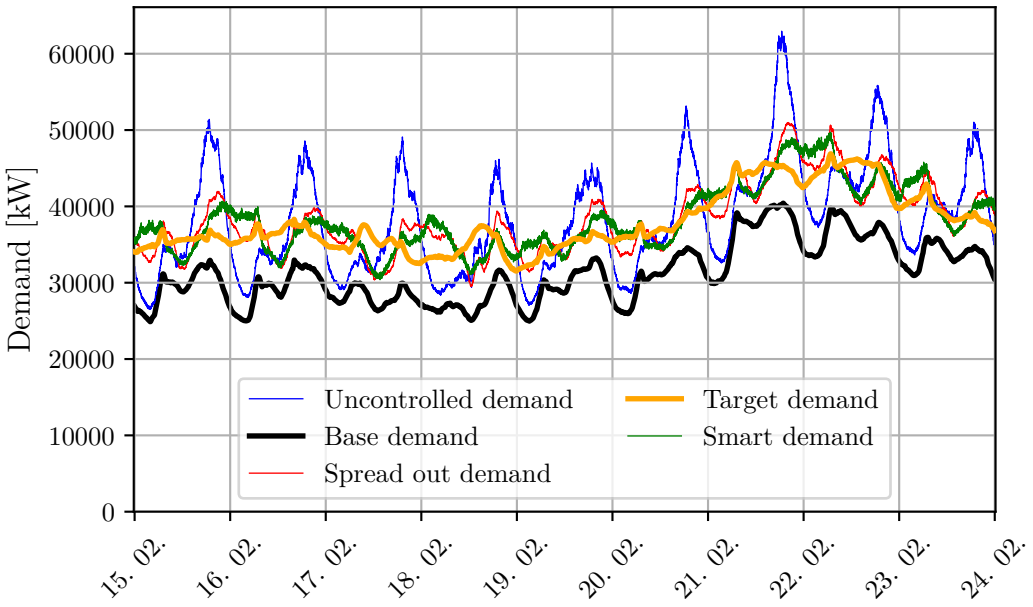

Figure 3.1: Results of the simulation from February 15 to February 24

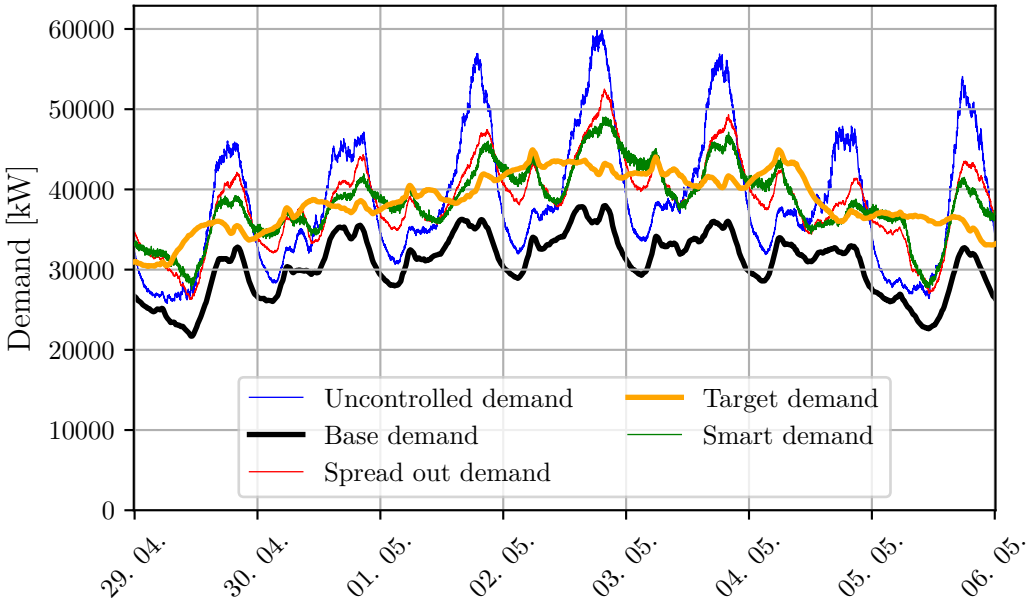

Figure 3.2: Results of the simulation from April 29 to May 6

During winter, the load-balancing performance of the algorithm is slightly worse, as seen in Figure 3.3. The peaks and valleys in the power demand are larger, usually because there are not enough cars available for charging which

could fill them by charging themselves, and because the energy capacity of household heating systems is too small to offset the heating by too long. Even so, the algorithm still performs considerably better than the uncontrolled algorithm, and slightly better than the spread out algorithm.

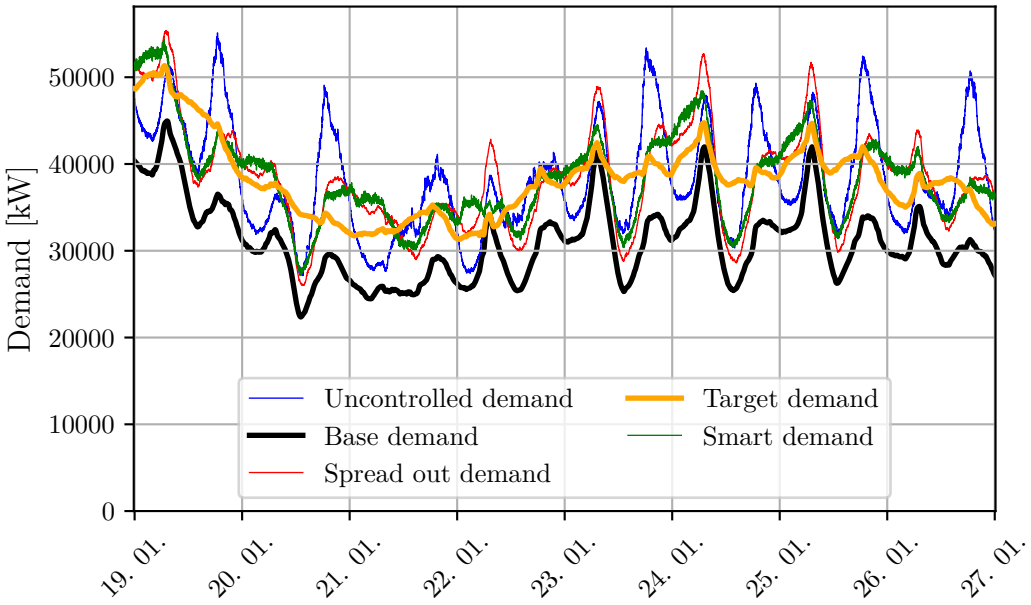

Figure 3.3: Results of the simulation from January 19 to January 27

In summer, the load-balancing performance gets even worse, as seen in Figure 3.4. The peaks in the demand in the afternoons and evenings correspond to the increased power usage caused by air conditioning, and because typical air conditioning units have a very small capacity to accumulate energy, their demand is hard to control.

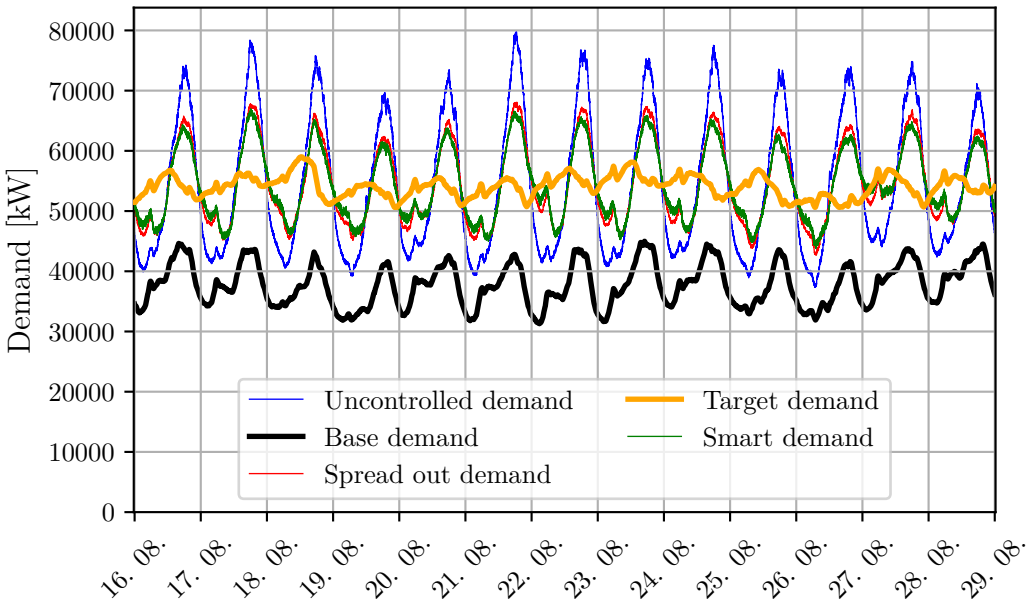

Figure 3.4: Results of the simulation from August 16 to August 29

Compared to the uncontrolled algorithm, the peaks and valleys are consid-

erably smaller with the smart algorithm, but because the need to fill valleys and therefore lower electricity price for households mostly coincides with the cars being available to charge, the performance of the smart algorithm is only slightly better than that of the spread out algorithm.

In September 2018, hurricane Florence hit Texas and caused adverse and unpredictable weather conditions. Because of that, the power demand predictions were unreliable, and as a result the target demand set by the simulator was not flat as desired, as shown in Figure 3.5.

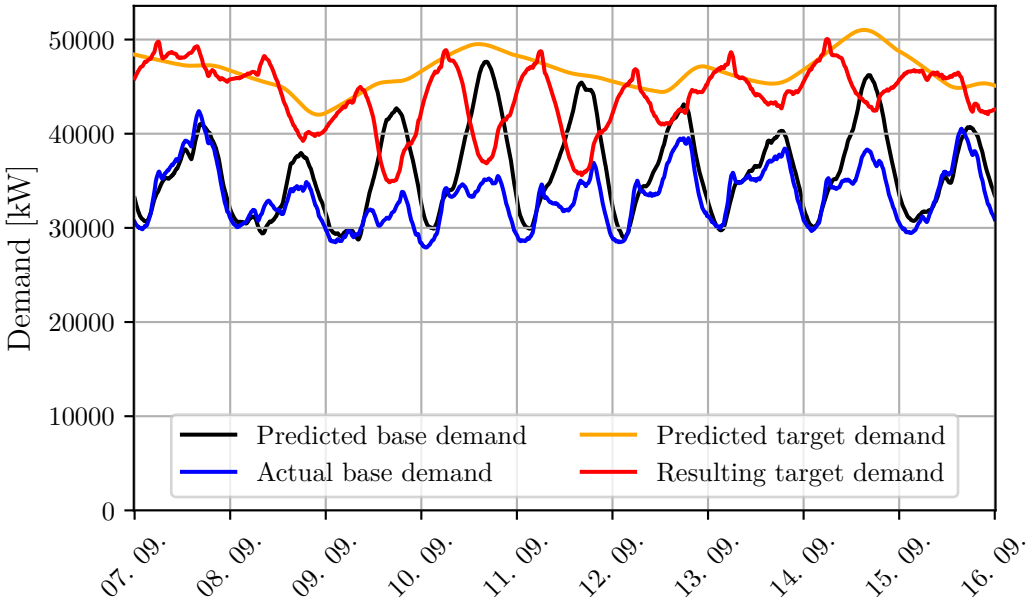

Figure 3.5: Base demand prediction and actual base demand during hurricane Florence

The simulation results during the hurricane are affected by the bad predictions as well, with the smart demand trying to approach the fluctuating target demand, as shown in Figure 3.6. It is still considerably flatter than the uncontrolled demand, and slightly flatter than the spread out demand as well.

To better compare the performance of the different algorithms, we also calculated a root mean squared error (RMSE) of the resulting demand compared to the target demand for each of the displayed intervals. These are shown in Table 3.1.

Table 3.1: Table of root mean square errors of different algorithms

Interval	Smart alg. RMSE [kW]	Spread out alg. RMSE [kW]	Uncontrolled alg. RMSE [kW]
January 19 to January 27	3349.0	4234.5	4878.9
February 15 to February 24	2481.7	2802.8	5677.8
April 29 to May 6	2968.1	4035.6	7396.3
August 16 to August 29	6215.0	7072.5	11487.0
September 7 to September 16	3711.2	5047.6	9706.3

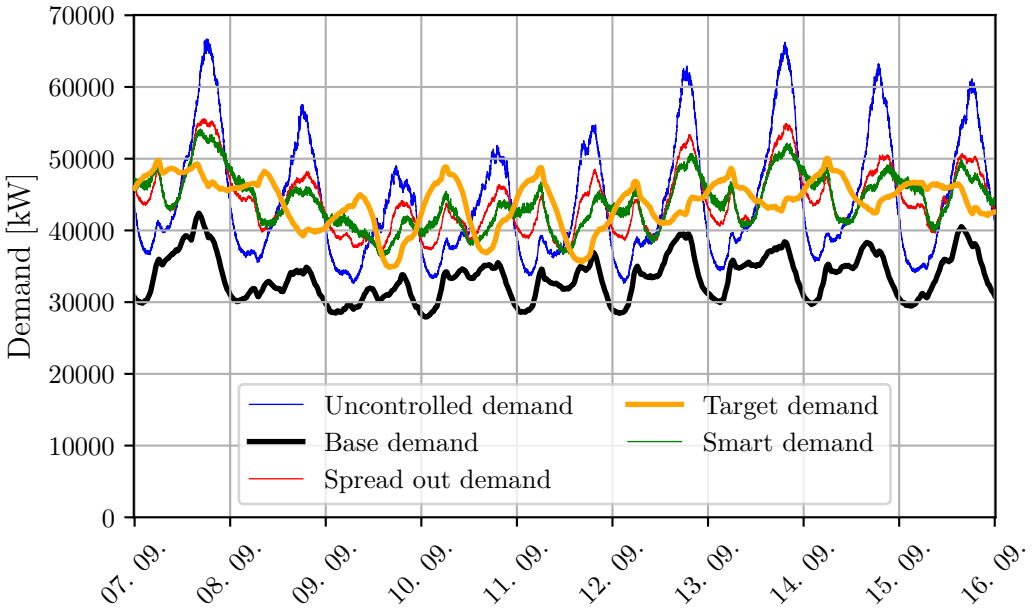

Figure 3.6: Results of the simulation from September 7 to September 16

Our demand control algorithm performs significantly better than the two other algorithms, hitting the target about twice as well as the uncontrolled algorithm, and significantly better even than the spread out algorithm.

3.2 Price settings comparison

We also evaluated the performance of the algorithm with different settings both for the minimum daily amount of cheap electricity for households and the minimum cheap electricity interval length could choose to use.

We tried several combinations of settings, of which a few select combinations are presented in Figure 3.7.

Setting `08x01` represents a configuration of a minimum daily amount of 8 hours of cheap electricity split into at least 1-minute long intervals; `08x60` represents a minimum of 8 hours of cheap electricity split into at least 60-minute long intervals, `16x60` represents a minimum of 16 hours of cheap electricity split into at least 60-minute long intervals, and `00x01` represents a special case where there is no minimum amount of cheap electricity per day which does not need to be continuous at all, but instead the simulator chooses the time slots in which to give households cheap electricity directly based on the price ratio so that it can achieve the desired price ratio perfectly (as it does in Algorithm 1).

We found that allowing for more precise price assignment enables the algorithm to control the demand better, albeit only slightly. The best performing configuration was one without a minimum amount of cheap electricity per day and minimum length of intervals in which the cheap electricity is approportioned.

Comparison of the root mean squared errors of the presented configurations is presented in Table 3.2.

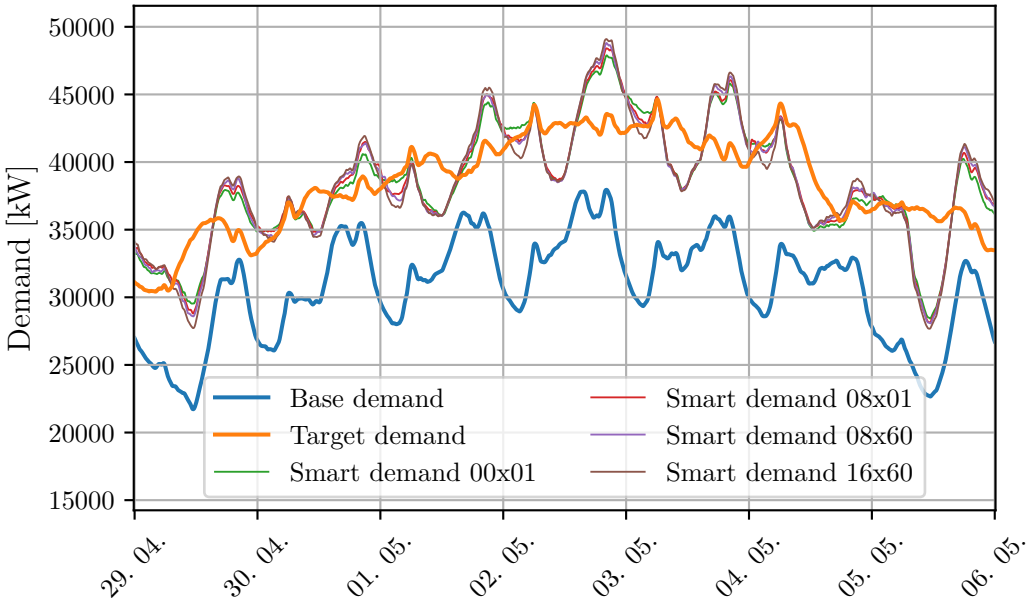

Figure 3.7: Comparison of electricity price settings

Table 3.2: Comparison of root mean square errors of different price configurations

Configuration	RMSE [kW]
00x01	2634.2
08x01	2822.1
08x60	2912.1
16x60	3124.2

3.3 Storage air conditioning

The largest deviations from the target demand happen in the summer months, as that is when most people are using their air conditioning units, which typically have a low accumulative capacity and therefore can't shift their energy consumption by much. We decided to test if utilizing air conditioning units with thermal storage would bring the electricity demand closer to the target.

We simulated the Texas power grid as in previous scenarios, but we changed the air conditioning units to utilize energy storage with a mean capacity of 20 kWh, to imitate commercially available ice storage air conditioning solutions [18]. The results of the simulation of two weeks in August 2018 are presented in Figure 3.8. We compare the demand on a grid with traditional air conditioning units (as inferred from the Dataport data), the demand on a grid with storage air conditioning units, and the demand that would happen in an uncontrolled scenario with traditional air conditioning units.

Utilizing energy storage in air conditioning units allows the power demand of the units to be controlled better, as they cool down their storage when electricity is cheaper, typically during the valleys in the grid-wide power demand, and then distribute the stored cold during the day when it is required, not needing to use large amounts of electricity when it is expensive. This greatly benefits the grid, as it allows the total demand to reach the target much more accurately than with traditional air conditioning units. Comparing the root mean square

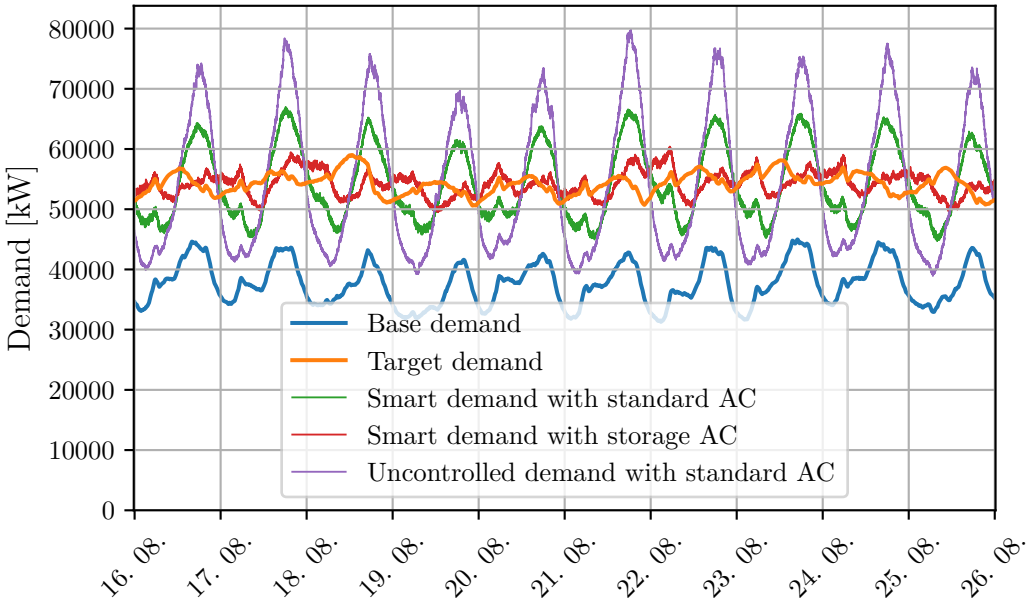

Figure 3.8: Results of the simulation with ice storage air conditioning

errors of the three scenarios further supports this hypothesis, as seen in Table 3.3.

Table 3.3: Comparison of root mean square errors of different air conditioning systems

Configuration	RMSE [kW]
Standard ACs	6354.9
Storage ACs	2781.4
Uncontrolled	11634.2

3.4 Customer motivation

To see if our demand control system would motivate subscribers to cooperate with the needs of the utility distributor, we also calculated the total amount the households would pay for their electricity supply for both scenarios: if they adjusted their appliance usage according to the variable electricity prices, and if they did not adjust their appliance usage and did not conform to the electricity pricing. The lower and higher electricity prices in the simulation were set the same as previously, 0.05 USD/kWh and 0.15 USD/kWh. The average electricity costs for one household for the simulated intervals described in Section 3.1 are shown in Table 3.4.

In the winter and spring months, when the main source of power demand is the charging of electric cars, households that adjust their electricity usage according to the prices set by our algorithm can achieve a meaningful electricity expenses reduction compared to households that do not, the savings can be up to half of the total cost.

In summer, as air conditioning starts to be utilized heavily, the electricity expenses rises rapidly, and as the demand from standard air conditioning units can be shifted in time only slightly, the households that adjust their usage have

Table 3.4: Average electricity costs for cooperating and non-cooperating subscribers

Interval	Average cost for cooperating subscribers [USD]	Average cost for non-cooperating subscribers [USD]
January 19 to January 27	8.5 $	16.7 $
February 15 to February 24	9.1 $	18.5 $
April 29 to May 6	8.9 $	16.3 $
August 16 to August 29	47.4 $	62.9 $
September 7 to September 16	17.8 $	28.5 $

their electricity expenses reduced less than in winter or spring, but the savings can still be over a quarter of the total cost. This serves to show that the motivation of the households to adjust their electricity usage to suit the distributor is quite significant, as it offers an opportunity to achieve substantial savings for the subscribers.

To evaluate if the utilization of storage air conditioning units would enable to drive the expenses even lower, we also calculated the total amount the households would pay for their electricity supply in the scenario described in Section 3.3. The lower and higher electricity prices in the simulation were set the same as previously, 0.05 USD/kWh and 0.15 USD/kWh. The average electricity costs for one household for the interval between August 16, 2018 and August 26, 2018 are shown in Table 3.5.

Table 3.5: Electricity costs based on the type of air conditioning units

Subscriber type	Electricity cost for August 16 to August 26 [USD]
Cooperating with standard ACs	36.6 $
Cooperating with storage ACs	25.3 $
Non-cooperating	49.0 $

Utilizing storage air conditioning units can bring further savings in the summer months, as reduction of the electricity expenses can be another quarter of the cost for the interval in which air conditioning is used heavily. This shows that storage air conditioning units offer substantial benefits not only to the utility distributor, but to the subsribers as well.

Conclusion

In this thesis we sought to develop an algorithm for controlling the power demand of households connected to the power grid, in order to reduce peaks and valleys in the total grid consumption, and to match the power demand with its production. We have set three key goals: to protect the privacy of the subscribers; to allow the system to control multiple appliance types; and to make cooperation with the utility distributor beneficial to the subscribers.

The algorithm we propose enables the power distributor to set a target demand profile which it would like to achieve, and based on the base demand prediction and historical subscriber behaviour it generates variable electricity prices for the households in order to influence them to use electricity at desired times. This serves to protect the subscribers privacy compared to centralized control algorithms, not requiring them to divulge information about their appliance usage to the distributor, allows for simultaneous control of multiple appliance types, and enables the distributor to incentivize the subscribers to cooperate with the distributor.

We have also presented a smart grid simulator enabling us to evaluate the performance of different demand control algorithms, simulating them on grids with households containing appliances based on real world data collected in various studies and projects.

Using the simulator, we have evaluated the performance of the presented demand control algorithm on a scale model of the Texas power grid, achieving a considerable reduction of the deviations in the total grid consumption compared to an uncontrolled household demand, therefore potentially reducing the reliance of the distribution grid on energy storage systems, and in turn reducing the operational costs. The simulation results also show that if the electricity pricing scheme employed by the presented algorithm would be adopted, it would strongly motivate subscribers to use electricity at times desirable by the utility distributor, as they could potentially save over a third on their electricity bills.

Electric vehicles have shown themselves to be the most controllable of the simulated appliances, as they have comparatively weak requirements for their charging as opposed to other power-hungry appliances such as air conditioning or space heating. This could be alleviated by incentivizing subscribers to adopt appliances with a high energy storage capacity, such as heat banks or ice storage air conditioning. We have simulated a modified grid where the households have adopted these appliances, and found that this allows the demand to be balanced significantly better still, and the savings for the households adopting these appliances could be even more significant still.

Future work

To evaluate the concept behind the presented algorithm, more research needs to be done. A few social issues present themselves, such as some subscribers being possibly unwilling to accept a varying electricity price tariff, hence the distributor would have to find ways to encourage its customers to participate in the demand control program, perhaps by offering incentives in form of a lower electricity price

overall, compared to regular customers.

The algorithm relies heavily on predictions of subscriber behaviour, which include their driving habits and household appliance usage. The projects collecting these statistics are still relatively scarce, therefore the available predictions might not be accurate in some locations and conditions. Additionally, in our simulations we control the charging of EVs only when they are at home, whereas in reality their owners may charge them at their workplace or other locations as well, offering further options for load-balancing.

Furthermore, the algorithm also relies on predictions of the base demand of the grid, which may be inacurrate at times, leading to less than ideal results. In our simulation, we require that all subscribers know their electricity price for at least one full day ahead. Instead, an approach with multiple levels could be implemented, where some subscribers would agree to know their prices only for a shorter period in the future, and if the base demand predictions would change over time, their prices would be set so their demand would balance the problems caused by the inacurrate predictions. caused by the inacurracies in the predictions.

Since some subscribers might be willing to give up even more control of their appliances, if incentivized properly, our algorithm could be used in conjunction with a different, more direct and centralized one, where the smaller, more consenting group of subscribers would have their appliances controlled directly by the distributor to accomodate for the issues in the demand of the rest of the subscribers.

Further research should also be done in collaboration with utility providers. We have only researched the numerical part of the system, but collaboration with the utility providers would offer additional insight into the technical challenges that could arise with its implementation in the real world, like the distribution of the electricity prices to households. Moreover, the providers have access to data about the grid not available to the public, and they can supply the ideal target demand profile for their grid, based on the characteristics of the power plants they operate.

Recently, a substantial amount of research has targeted the possibility of integrating electric vehicles deeper into the power grid, with vehicle-to-grid systems proposing to utilize the batteries in electric vehicles as load-balancing batteries for the whole grid, drawing power when the total grid demand is low, and selling it back when demand is high, to supplement the power plants on the grid, all while ensuring that the batteries of the vehicles will be fully charged when the vehicles are needed for usage. Our algorithm could be well suited for these scenarios, as households get high electricity prices during intervals when the power demand on the grid is expected to be high, to discourage them from using power unnecessarily, and during intervals when the power demand is expected to be low they get lower prices, to encourage them to use power to fill the valleys in the demand. This would work hand in hand with the vehicle-to-grid systems, encouraging EVs to charge their batteries during lower price intervals, further filling the valleys, and then selling the stored electricity back to the grid for a higher price when the grid needs a higher supply. Further modifications to the simulator can be done in the future to enable it to model vehicle-to-grid systems.

Bibliography

[1] E. Ungar and K. Fell. Plug in, turn on, and load up. *IEEE Power and Energy Magazine*, 8(3):30–35, May 2010. ISSN 1540-7977. doi: 10.1109/MPE.2010.936354.

[2] K. Clement-Nyns, E. Haesen, and J. Driesen. The impact of charging plug-in hybrid electric vehicles on a residential distribution grid. *IEEE Transactions on Power Systems*, 25(1):371–380, Feb 2010. ISSN 0885-8950. doi: 10.1109/TPWRS.2009.2036481.

[3] K. Schneider, C. Gerkensmeyer, M. Kintner-Meyer, and R. Fletcher. Impact assessment of plug-in hybrid vehicles on pacific northwest distribution systems. In *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, pages 1–6, July 2008. doi: 10.1109/PES.2008.4596392.

[4] A. G. Boulanger, A. C. Chu, S. Maxx, and D. L. Waltz. Vehicle electrification: Status and issues. *Proceedings of the IEEE*, 99(6):1116–1138, June 2011. ISSN 0018-9219. doi: 10.1109/JPROC.2011.2112750.

[5] Federal Highway Administration U.S. Department of Transportation. National household travel survey, 2017. URL `https://nhts.ornl.gov`.

[6] Pecan Street Inc. Dataport, 2019. URL `https://dataport.pecanstreet.org/`.

[7] U.S. Department of Energy. Grid energy storage, 2013. URL `https://www.energy.gov/sites/prod/files/2014/09/f18/GridEnergyStorageDecember2013.pdf`.

[8] D. Wu, D. Aliprantis, and L. Ying. Load scheduling and dispatch for aggregators of plug-in electric vehicles. *IEEE Trans. Smart Grid*, 3:368–376, 03 2012. doi: 10.1109/TSG.2011.2163174.

[9] M. G. Vayá and G. Andersson. Centralized and decentralized approaches to smart charging of plug-in vehicles. In *2012 IEEE Power and Energy Society General Meeting*, pages 1–8, July 2012. doi: 10.1109/PESGM.2012.6344902.

[10] E. L. Karfopoulos and N. D. Hatziargyriou. A multi-agent system for controlled charging of a large population of electric vehicles. *IEEE Transactions on Power Systems*, 28(2):1196–1204, May 2013. ISSN 0885-8950. doi: 10.1109/TPWRS.2012.2211624.

[11] C. Ahn, C. Li, and H. Peng. Decentralized charging algorithm for electrified vehicles connected to smart grid. In *Proceedings of the 2011 American Control Conference*, pages 3924–3929, June 2011. doi: 10.1109/ACC.2011.5990895.

[12] L. Gan, U. Topcu, and S. H. Low. Optimal decentralized protocol for electric vehicle charging. *IEEE Transactions on Power Systems*, 28(2):940–951, May 2013. ISSN 0885-8950. doi: 10.1109/TPWRS.2012.2210288.

[13] Qiao Li, Tao Cui, Rohit Negi, Franz Franchetti, and Marija D. Ilic. On-line decentralized charging of plug-in electric vehicles in power systems. *CoRR*, abs/1106.5063, 2011.

[14] E. S. Rigas, S. D. Ramchurn, and N. Bassiliades. Managing electric vehicles in the smart grid using artificial intelligence: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):1619–1635, Aug 2015. ISSN 1524-9050. doi: 10.1109/TITS.2014.2376873.

[15] J. Fink and J.L. Hurink. A greedy algorithm for local heating. Submitted 2017.

[16] Electric reliability council of texas. URL `http://www.ercot.com/`.

[17] U.S. Energy Information Administration. Household energy use in texas, 2009. URL `https://www.eia.gov/consumption/residential/reports/2009/state_briefs/pdf/tx.pdf`.

[18] Ice energy. URL `https://www.ice-energy.com/`.