iProgram: Inferring Smart Schedules for Dumb Thermostats

Srinivasan Iyengar, Sandeep Kalra, Anushree Ghosh,
David Irwin, Prashant Shenoy, Benjamin Marlin
University of Massachusetts Amherst

ABSTRACT

Heating, ventilation, and air conditioning (HVAC) accounts for over 50% of a typical home's energy usage. A thermostat generally controls HVAC usage in a home to ensure user comfort. In this paper, we focus on making existing "dumb" programmable thermostats smart by applying energy analytics on smart meter data to infer home occupancy patterns and compute an optimized thermostat schedule. Utilities with smart meter deployments are capable of immediately applying our approach, called iProgram, to homes across their customer base. iProgram addresses new challenges in inferring home occupancy from smart meter data where i) training data is not available and ii) the thermostat schedule may be misaligned with occupancy, frequently resulting in high power usage during unoccupied periods. iProgram translates occupancy patterns inferred from opaque smart meter data into a custom schedule for existing types of programmable thermostats, e.g., 1-day, 7-day, etc. We implement iProgram as a web service and show that it reduces the mismatch time between the occupancy pattern and the thermostat schedule by a median value of 44.28 minutes (out of 100 homes) when compared to a default 8am-6pm weekday schedule, with a median deviation of 30.76 minutes off the optimal schedule. Further, iProgram yields a daily energy savings of 0.42kWh on average across the 100 homes. Utilities may use iProgram to recommend thermostat schedules to customers and provide them estimates of potential energy savings in their energy bills.

Categories and Subject Descriptors

J.7 [**Computer Applications**]: Computers in Other Systems—*Consumer products*

General Terms

Design, Experimentation, Measurement

Keywords

Energy, Electricity, HVAC, Grid

BuildSys'15, November 4–5, 2015, Seoul, South Korea..

ACM 978-1-4503-3981-0/15/11 ...$15.00.
DOI: http://dx.doi.org/10.1145/2821650.2821653 .

1. INTRODUCTION

Buildings account for over 40% of the energy and 75% of the electricity usage in the U.S. [15] and other developed countries. Heating, cooling and ventilation (HVAC) alone accounts for 54% of the energy usage in residential buildings. Thermostats typically regulate use of central HVAC systems by enabling users to specify a desired setpoint temperature, and then cycling the system on and off to maintain the temperature within a fixed range of the setpoint.

The simplest type of thermostat requires users to manually switch the HVAC system on and place it into heating or cooling mode, as well as specify the desired temperature. Manual thermostats, while simple and inexpensive, are i) prone to human error, since users may forget to turn them off when leaving for an extended period, and ii) reduce user comfort, since users cannot activate them in advance of their arrival, causing an initial period of discomfort on entry as a home (or room) heats up or cools down to the setpoint. To address these drawbacks, most modern thermostats are programmable. A programmable thermostat enables a user to manually program a thermostat schedule, which specifies the time each day the HVAC system should be on and its corresponding setpoint temperature. Users derive a schedule based on when they expect to be home and away. If a user's expectations are correct, then a programmable thermostat can reduce human error by automatically turning off (or adjusting the setpoint) when a user leaves. To increase comfort, users may also enter schedules that pre-cool or pre-heat a home (or room) in advance of their expected arrival.

Programmable thermostats are simple and cheap devices present in tens of millions of homes. While programmable thermostats are marketed for their potential energy savings, prior research has shown they do not save energy in practice, largely because users often program them incorrectly or not at all [21, 23]. Unfortunately, the interface to many programmable thermostats is difficult to navigate and tedious to program, which often discourages users from ever programming them. Even when taking the time to program a schedule, users are left with the complex task of determining a static schedule that optimizes their energy savings and comfort given their occupancy pattern, which may be dynamic and difficult to predict. Further, since daily activities and, thus, occupancy patterns inevitably change over time, e.g., summer activities differ from winter activities, users must periodically re-program their thermostat as their optimal schedule changes, or risk losing any of its energy-saving benefits. As a result of these challenges, many users avoid programming thermostats, which defeats their purpose.

Smart thermostats, which sense occupancy and automatically program a schedule without user intervention, were introduced to address the drawbacks of programmable thermostats. There has been a large amount of prior work on smart thermostats that monitor and predict occupancy patterns using various techniques and

sensors, e.g., motion or GPS [1, 10, 11, 18, 24], and numerous commercial smart thermostats are now available, including NEST [20], Lyric [19], Ecobee [7], etc. However, smart thermostats are still niche devices for energy-efficiency enthusiasts, largely due to their high cost and the overhead of installing them. For example, smart thermostats currently cost $>10\times$ more than a programmable thermostat, e.g., $250 for the NEST versus $15 or less for an entry-level programmable thermostat. While the cost of smart thermostats may decrease over time, tens of millions of "dumb" programmable thermostats will remain in homes for many years to come.

Thus, we focus on making existing "dumb" programmable thermostats smart without requiring new investments to upgrade to an expensive smart thermostat or install additional sensors. Our goal is to address the problems that prevent existing programmable thermostats from fully realizing their energy savings potential. Our key insight is that electricity usage data from smart meters indirectly reveals occupancy patterns that can be used to automatically derive custom thermostat schedules for each home. We argue that schedules automatically derived from smart meter data are better aligned with occupancy than schedules manually entered by users, enabling users to save energy and making it easier for them to program their thermostat. Importantly, our approach uses an existing sensor that is already widely deployed in homes: as of 2014, 50 million smart meters had been installed in the U.S., which covers more than 43% of all meters [14]. Further, smart meter installations are growing rapidly: in the past five years from 2009 to 2014 the number of deployments increased by nearly a factor of four (from 13 million in 2009 to 50 million in 2014 [14]) with 75% coverage expected by 2016 [12]. In addition, utilities that collect smart meter data have an interest in energy-efficiency, e.g., as part of government-mandated energy-efficiency programs, and direct access to a large set of customers and data. Thus, utilities are in the best position to suggest thermostat schedules to users, to provide incentives for users to adopt them, and to verify their adoption.

Our system, called iProgram, analyzes data from a home's smart electricity meter to infer occupancy patterns and derive a custom thermostat schedule. iProgram extends recent research [4, 16] in inferring occupancy from smart meter data in important ways, which are driven by its application context. For example, while prior work uses training data to build a model that correlates occupancy with certain features of smart meter data [16], our application does not permit gathering training data, since it would require installing temporary occupancy sensors in tens of millions of homes. In addition, the models in prior work typically use a home's power usage as primary feature to infer occupancy. Unfortunately, homes with misprogrammed thermostats often exhibit high power usage when unoccupied, since the HVAC system is often a home's largest load. Thus, prior techniques often perform poorly on exactly the homes with the most to gain from iProgram. Finally, we evaluate iProgram, not only using occupancy-centric accuracy metrics, but also by its ability to derive effective thermostat schedules in real homes.

Our hypothesis is that iProgram can improve HVAC efficiency *en masse* by analyzing smart meter data to infer long-term occupancy patterns and compute a custom thermostat schedule. In evaluating our hypothesis, this paper makes the following contributions.

- **Targeted Occupancy Detection.** We develop a targeted occupancy detection technique that performs well on energy-inefficient homes with misprogrammed thermostats, and is applicable to utility-scale datasets where training data is not available. Rather than apply a general machine learning approach, we craft a domain-specific technique that leverages time-series component analysis to isolate the "burstiness" of interactive loads that directly correlate with occupancy.
- **Deriving Thermostat Schedules.** We show how to translate the inferred long-term occupancy patterns into a probability distribution of occupancy to derive a custom static thermostat schedule. We present extensions to support scheduling different types of thermostats, e.g., 5-2-day, 7-day, etc., identifying sleeping periods, and changing occupancy patterns.
- **Open Cloud Service.** We implement iProgram as an open cloud service where users may upload their smart meter data and receive suggested thermostat-specific schedules. The service also exposes a web services API to enable third-party devices, such as a basic WiFi-enabled (but non-learning) programmable thermostat, to access its schedules.
- **Evaluation.** We evaluate the accuracy of iProgram's targeted occupancy detection technique and its derived thermostat schedules, as well as its ability to infer thermostat schedules in real homes. For the former, we use data from over 100 homes from the ECO [3, 16] and Pecan Street datasets [22], which include one-minute power data for up to six months. For the latter, we conduct an anonymous user study on 8 homes where we analyze their smart meter data to suggest a thermostat schedule based on their inferred occupancy pattern.

2. BACKGROUND AND MOTIVATION

iProgram assumes a home is equipped with a networked power meter—a *smart meter*—that reports aggregate electricity usage at fine-grained intervals, e.g., every one to fifteen minutes. Most residential HVAC systems are partially or fully electric: nearly all space cooling, i.e., air conditioning, is electric and 38.1% of U.S. homes use some form of electric space heating [8]. Even when using oil- or natural gas-based heating, the mechanical systems that circulate the hot air or water are electric. In addition, the type of HVAC system in a home is typically a matter of public record, and available to iProgram (and utilities). Thus, given an address, iProgram can identify the homes and seasons where HVAC usage is included in smart meter data, and appropriately configure itself.

iProgram specifically targets homes that use programmable thermostats to regulate HVAC operation. Only 7% of U.S. homes have central heating but no thermostat, and only 1% of homes have central cooling but no thermostat [9]. In contrast, 37% of U.S. homes with central heating have a programmable thermostat, and 29% of homes with central cooling have a programmable thermostat [9]. Importantly, iProgram does not dictate a specific type of programmable thermostat or its configuration. There are a wide range of thermostats available, including 1-day programmable, 7-day programmable, and 5-2-day programmable. In addition, some programmable thermostats are now networked, enabling users (or third-party software) to remotely program and control them. Note that network-enabled programmable thermostats differ from high-end smart thermostats in that they do not automatically program a schedule by learning home occupancy patterns via sensors.[1]

For homes with only manual thermostats (or no thermostat at all), iProgram can still compute the misalignment between the HVAC system's operation and the occupancy pattern, and then estimate the potential energy savings from upgrading to a programmable (or smart) thermostat.

2.1 Problem Statement

Our goal is to analyze a home's smart meter data to derive a thermostat schedule that accurately reflect its pattern of occupancy. Formally, we represent a schedule as a function $S(t)$ that returns a desired thermostat setpoint temperature at each time t. In essence,

[1]See `http://wifithermostat.com` for examples.

the thermostat schedule defined by $S(t)$ consists of a series of variable-length intervals that specify different setpoint and setback temperatures, which denote the desired temperature when users are present and away, respectively. A thermostat schedule must specify either a setpoint or setback temperature for each interval. In addition, for programmable thermostats, the thermostat schedule repeats every interval T, which limits t's range. For example, for a 1-day programmable thermostat $T = 24$ hours, while, for a 7-day programmable thermostat, $T = 168$ hours.

iProgram derives the schedule $S(t)$ by analyzing the time-series $P(t)$ of power readings generated by a smart meter to infer when a home is occupied. As in prior work, we represent occupancy as a binary function $O(t)$, where zero is an unoccupied home and one is an occupied home. Occupancy detection then requires inferring $O(t)$ from $P(t)$. Prior work on inferring occupancy from smart meter data leverages the insight that power usage that is high and variable often correlates with occupancy, since occupants use interactive devices that consume energy when home. Prior work evaluates many approaches for inferring occupancy based on this intuition, ranging from employing simple thresholds on power's mean, variance, and range [4] to advanced machine learning techniques using Hidden Markov Models (HMMs), Support Vector Machines (SVMs), and k-Nearest Neighbor (k-NN) classifiers [16]. While accuracy depends on the correlation between a home's occupancy and electricity usage, it generally ranges between 75% and 95%.

2.2 Research Challenges

Unfortunately, applying prior work to iProgram's application of deriving optimal thermostat schedules *en masse* from smart meter data is problematic for multiple reasons. As we describe below, iProgram i) does not have access to training data, ii) has a particular emphasis on reducing the energy use of energy-inefficient homes with misprogrammed thermostats, and iii) focuses on deriving thermostat schedules and not just detecting occupancy.

No Training Data. Prior techniques for inferring occupancy from smart meter data rely on an initial set of training data to build a model that correlates occupancy with power data. Gathering training data requires instrumenting the home with occupancy sensors or directly tracking occupants' location, e.g., via their smartphone. While gathering training data is feasible in a few homes, it does not scale to utility-sized datasets. Further, many users will not likely consent to such instrumentation due to privacy concerns. While improving home thermostat schedules across a large number of homes has the potential to significantly reduce energy use, the cost to privacy may not be worth the benefit in energy savings for many users. While privacy concerns also exist with smart meters [5], consumers must already trust utilities with their smart meter data.

Of course, one way to address the lack of training data using the techniques above is to simply apply models built from training data from some sample test homes to a much larger set of homes. Other energy data analytics, such as Non-Intrusive Load Monitoring (NILM), often take this approach [17]. However, NILM's goal is to extract common, and often repetitive, device power signatures from smart meter data. Importantly, these signatures are often similar across homes. For example, every home likely has a refrigerator with a similar repeating pattern of energy consumption. In contrast, occupancy patterns, as well as how occupants interact with electrical devices, are highly user-dependent, and typically vary significantly across homes. As a result, applying training data from test homes to a wider set of homes without training data is not as useful.

Misprogrammed Thermostats. Prior techniques for inferring occupancy from smart meter data work well when occupancy patterns align with power usage. However, when a thermostat is misprogrammed, it will turn on the HVAC system when occupants are not home, or turn off the HVAC system when they are home. This results in an inherent mismatch between the the HVAC system's power usage, which is a large component of a home's overall power usage, and its occupancy. In this case, using power usage to derive occupancy is more error-prone, especially during the misaligned intervals when the HVAC system is on and the home is unoccupied (or vice versa). In fact, as shown in Figure 1, previous occupancy techniques that use the magnitude of power consumption as a metric to infer occupancy often yield incorrect results during misaligned periods. To be effective, iProgram must derive more accurate occupancy patterns despite periods of misalignment.

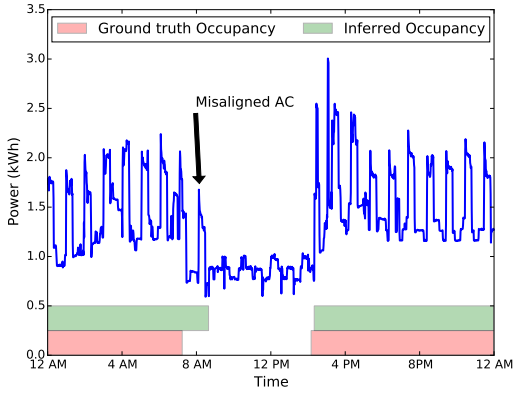


Figure 1: Any approach that correlates high power usage with occupancy is error-prone for misprogrammed thermostats.

More generally, the more energy-efficient a home, the more the simple features from prior work, e.g., power's mean, variance, and range, will correlate with occupancy—since an energy-efficient home is one that uses a minimal amount of power when occupants are away. Conversely, the less energy-efficient a home, the less these features will correlate with occupancy. This relationship between energy-efficiency and the accuracy of energy data analytics (and privacy) has also been identified in computer systems [6]. Thus, to accurately infer occupancy in energy-inefficient homes, iProgram requires a more targeted approach than prior work. As we discuss, we craft an approach specifically tailored to the inherent characteristics of power data that signify occupancy.

Thermostat Scheduling. There is substantial work on deriving thermostat schedules from occupancy patterns. For example, smart thermostats actively sense occupancy using motion or GPS sensors [18, 24] and dynamically alter a thermostat's schedule in real time based on the current occupancy. The scheduling problem for iProgram differs from smart thermostats, since it computes a static schedule based on long-term occupancy patterns for a programmable thermostat, rather than directly sensing and dynamically adjusting to real-time occupancy. Work by Gao and Whitehouse [13], which computes a thermostat schedule based on long-term occupancy statistics gathered from sensors, is most similar to iProgram's scheduling problem. However, their approach is limited to computing the time and duration of a single unoccupied period each day, while our approach is general and computes the optimal thermostat schedule for a given long-term pattern of occupancy.

2.3 Evaluation Metrics

Since iProgram ultimately focuses on translating occupancy patterns into a thermostat schedule, we use evaluation metrics relevant to thermostat scheduling, rather than simply quantifying the accuracy of our new binary classifier for occupancy detection. In par-

ticular, we use *miss time* (MT) and *waste time* (WT) to quantify the performance of a thermostat schedule. Intuitively, the miss time is the amount of time a home is occupied but the HVAC system is not on, i.e., where its temperature deviates by more than X$^\circ$ from the setpoint. Likewise, the waste time is the amount of time a home is unoccupied and the HVAC system is on, i.e., where the temperature deviates by less than X° from the setpoint. Formally, we define the miss time and waste time in terms of the *conditioning period* (CP) below for T time periods with occupancy $O(t) \in 0, 1$.

$$CP(t) = \begin{cases} 1, & \text{if the home is conditioned at time t,} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Given CP(t), we define the average daily miss time and waste time over an N day period, as shown below.

$$MT = \frac{\sum_t (O(t) - CP(t))}{N} \quad \forall\ t \text{ where } O(t) = 1 \quad (2)$$

$$WT = \frac{\sum_t (CP(t) - O(t))}{N} \quad \forall\ t \text{ where } CP(t) = 1 \quad (3)$$

Our definition of miss time differs from prior work [13], which defined the metric by assuming only a single contiguous unoccupied period during the day, e.g., from the time occupants leave in the morning to when they return in the afternoon/evening. In general, our metric reflects that there may be multiple non-contiguous unoccupied periods during the day, e.g., if someone leaves for work, comes home for lunch, and then leaves again. Our definition of miss time is applicable to such non-contiguous schedules.

Of course, a simple way to achieve a miss time of zero is to never alter the thermostat setpoint, even when a home is not occupied; likewise, a simple way to achieve a waste time of zero is to never turn on the HVAC system, even when the home is occupied. There is a tradeoff between miss time and waste time, which corresponds to a tradeoff between user comfort and HVAC energy usage: a low miss time signifies high user comfort but results in higher HVAC energy usage, while a low waste time signifies low HVAC energy usage but results in lower user comfort. To resolve this tradeoff, we also define the *mismatch time* for a particular schedule as the sum of its miss time and waste time. The mismatch time essentially places an equal value on both energy-efficiency and comfort.

Note that iProgram computes a thermostat schedule $S(t)$ for homes that specifies when the thermostat should be at a setpoint or a setback temperature. However, it does not determine the setpoint and setback temperature for the user. These values should be defined by a home's occupants, since they depend on occupants' subjective notion of comfort, i.e., some people may like their home warmer or colder than others. iProgram could aid in setting setback temperatures based on the regularity and length of a home's occupancy pattern. For example, the more regular and predictable a home's schedule, the deeper the setback that is possible without causing discomfort on arrival (since the schedule will be able to reliably pre-heat the home before the expected arrival). In addition, setting the depth of the setback also depends on a home's size and insulation, since these attributes affect the time it takes to change the temperature back to the setpoint temperature. We leave setting the setback temperature's depth as future work: in our current prototype, iProgram computes the schedule, while the user sets their desired setpoint and setback temperature for different periods.

In addition to scheduling-centric evaluation metrics, we also quantify the performance of our occupancy detection technique using standard metrics for binary classifiers, which are defined based on values in a confusion matrix. Specifically, we compute the percentage of time our occupancy detector yields a true positive (TP), true negative (TN), false positive (FP) and false negative (FN). Given these values, we compute accuracy as shown below.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

Since different homes may be occupied for different durations, accuracy is not an ideal evaluation metric. For example, if a home is occupied 90% of the time, then an approach that simply assumes the home is always occupied will perform well. Thus, a better single measure of a binary classifier's performance is the Matthews Correlation Coefficient (MCC), shown below, which ranges from -1.0 to 1.0 where a 0.0 is equivalent to random guessing, a 1.0 is equivalent to perfectly predicting occupancy, and a -1.0 is equivalent to being perfectly wrong about every prediction. The MCC is generally a better single measure for a binary classifier than Precision, Recall, or F-Score, since it is robust to different values of the ground truth data and does not depend on the arbitrary labeling of values in the confusion matrix.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (5)$$

3. IPROGRAM DESIGN

iProgram's basic workflow, depicted in Figure 2, is to i) apply an occupancy detection technique to infer occupancy from smart meter data, ii) use the inferred long-term occupancy pattern to compute a histogram of the probability of occupancy at any time each day, and iii) generate a static schedule that minimizes the mismatch time (or optimizes a user-specified tradeoff between miss time and waste time). Since prior general occupancy detection techniques [4, 16] do not work well for iProgram, as discussed in Section 2, we present a new targeted technique designed specifically to work without training data on energy-inefficient homes with misprogrammed thermostats. We then discuss translating the inferred occupancy patterns into non-contiguous schedules for programmable thermostats. Our techniques combine several statistical methods, including time-series decomposition and probability-based occupancy modeling, to derive custom thermostat schedules. In addition, we present multiple extensions to our basic technique.

3.1 Detecting Occupancy

Our approach derives from the notion of "burstiness" in network protocols, such as TCP [26]. A TCP flow consists of a sequence of packets from a source to a destination server. Intuitively, a burst is then a group of consecutive packets with shorter inter-arrival times than packets occurring before or after the group. In addition, a group of bursts with shorter inter-arrival times than bursts occurring before or after the group is referred to as a family of bursts.

In our context, packets are analogous to significant power events, i.e., increases or decreases in power. We apply burstiness to occupancy detection, in part, because we view it as the most innate characteristic of power usage that directly correlates with occupancy: when occupants are home they cause bursts of power events by turning devices on and off. Other characteristics of home power usage that correlate with occupancy, including power's mean, variance, and range, only indirectly derive from burstiness. For exam-

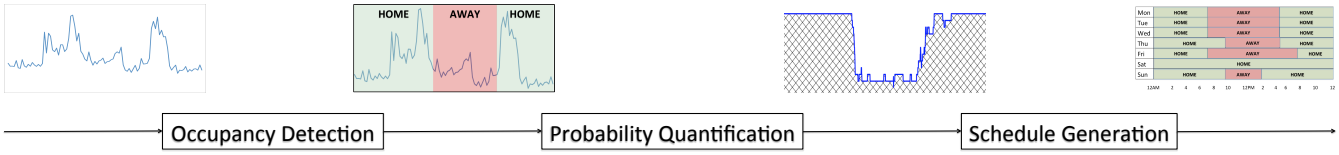


Figure 2: iProgram's basic workflow

ple, power's mean tends to increase as occupants turn devices on. Likewise, power's variance (or standard deviation) is essentially a dampened form of burstiness, while power's absolute range simply captures the largest burst within some period.

Since the characteristics above are not the most direct reflection of occupant presence in smart meter data, their correlation with occupancy is generally less strong than burstiness. As discussed in Section 2, any technique that correlates occupancy with a high average power will incorrectly detect occupancy if the HVAC system turns on in an unoccupied home.

Of course, burstiness, as with the other metrics, may not always perfectly correlate with occupancy, as large background loads like the HVAC system also cause bursts of power events. To isolate these bursts, we use basic time-series decomposition to deconstruct the power time series (P) into the seasonal(P_s), trend(P_t), and noise (P_n) components. The seasonal component of a time series captures patterns that tend to repeat semi-regularly over known, fixed periods of time (of any duration). Thus, the seasonal component only includes the non-interactive background loads, which are typically periodic and cause bursts of power usage in an unoccupied home. In contrast, the trend component captures the general non-repetitive trend in the data, e.g., increasing or decreasing over time, by filtering out medium and high frequency fluctuations. Thus, in our algorithm below, we only correlate bursts in the power usage of the trend component with occupancy. Finally, the noise component represents random fluctuations that are neither repetitive nor follow a trend. We extract the noise component from the trend component to eliminate random bursts that are less likely to be associated with human activity, which are generally part of a trend.

Below, we present our occupancy detection algorithm that combines time-series decomposition and burstiness to detect occupancy from smart meter data. For each day, from time $dStart$ to time $dEnd$, we isolate the periodic background loads using time-series decomposition. Specifically, we compute the seasonal component of the daily time-series for all window sizes $[1 \ldots \nu_{max}]$, and then select the value ν_{opt} that maximizes the seasonal component's average power consumption over the day. Here, the time-series decomposition will extract any repeating patterns of usage that are near the specified window size. Our goal is to isolate the HVAC system's operation from the power usage; since the HVAC system is generally the largest load in the home, computing the seasonal component at its window size, i.e., its periodic interval, will result in the largest energy consumption over the day. Since we do not know the HVAC system's duty cycle and periodicity in advance, we perform an exhaustive search over all possible window sizes.

After isolating the seasonal component with the window size that maximizes daily energy consumption, we consider the magnitude of the change in the trend component over time. In particular, if the change in the power usage of the trend component from any time $t-1$ to time t is greater than the average reading-to-reading change in power, we flag time t as a power event. If the time period between two events is within a threshold δ_{day}, we label the time period between the events as a burst. Similarly, if the time period between two bursts is within a threshold δ_{day}, we label the time period between the bursts as a family of bursts. We set $\delta_{day} = 2$ hours in our current prototype. We then interpolate between families of bursts and label periods between them as being occupied, while all other periods are labeled as non-occupied. Note that the algorithm above only considers daytime detection from 6am to 11pm.

Algorithm 1 Occupancy Detection

procedure OCCUPANCYDETECTION(P)
Initialize: $O[i] \leftarrow 0 \quad \forall\, i \in 1 \ldots length(P)$
  $P_{day} \leftarrow P[dStart : dEnd]$
  $P_s[\nu], P_t[\nu], , P_n[\nu] \leftarrow Decompose(P_{day}, \nu) \quad \forall\, \nu \in 1 \ldots \nu_{max}$
  $\nu^{opt} \leftarrow \underset{\nu}{argmax} |\sigma(P_s[\nu])|$
  $P_s^{opt}, P_t^{opt}, P_n^{opt} \leftarrow Decompose(P_{day}, \nu^{opt})$
  $O_{day} \leftarrow \mathbb{1}(\nabla_1[P_t^{opt}]| > mean(|\nabla_1[P_t^{opt}]|))$
  $O_{day}[i : i+1] \leftarrow 1 \quad \forall\, T(i+1) - T(i) <= \delta_{day}$
  $O_{night}[i] \leftarrow O_{day}[i] \quad \forall\, i \in T[nStart : nEnd]$
  $O_{night}[i : i+1] \leftarrow 1 \quad \forall\, T(i+1) - T(i) <= \delta_{night}$
  $O[i] \leftarrow 1 \quad \forall\, i \in O_{day}[i] = 1$
  $O[i] \leftarrow 1 \quad \forall\, i \in O_{night}[i] = 1$
  return O
end procedure

We detect nighttime occupancy separately from daytime occupancy, since nighttime occupancy is not strongly correlated with burstiness in the power usage. Here, we focus on evening and morning bursts that occur between 7pm-11pm and 6am-9am, respectively. If we do not detect a burst in either period, we label the nighttime period as unoccupied, otherwise we compute occupancy as above. This strategy works well in practice, although it cannot differentiate between occupants going to sleep and then waking up in the middle of the night, and occupants going out late at night and then coming home in the early morning hours. Pseudocode for our algorithm above is shown in Algorithm 1.

Note that burst detection above is based on the magnitude of each change in power relative to the average reading-to-reading change in power for a home. Thus, the algorithm requires no training data of ground truth occupancy to learn a model *a priori*. We select the average change in power as the threshold because we have found that changes in home power are bimodal with many small changes in power (near zero) that stem from natural variations in a device's power usage, and a few large changes in power that stem from human activity. In computing the mean change in power, we are able to detect the latter, while eliminating the former.

3.2 Generating Schedules

The algorithm above infers periods of occupancy and non-occupancy each day. Over many days, we can then compute a histogram that shows the probability of occupancy for each sampling interval, e.g., every minute in our dataset. The histogram captures the regularity of a home's occupancy pattern. Figure 3 shows an example of the histogram for Home B in the UMass Smart* dataset over a three week period. The figure shows the probability the

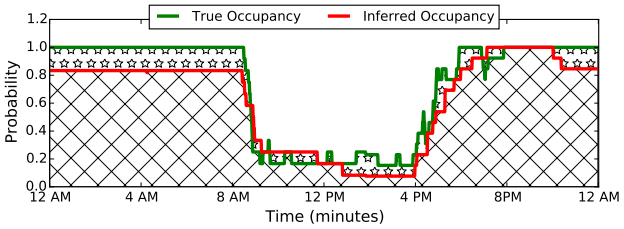


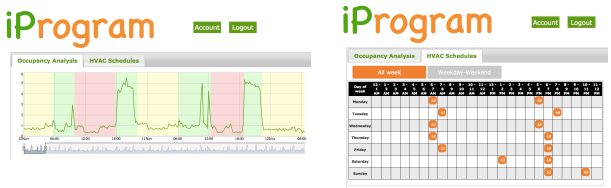
Figure 3: An example histogram of the probability of occupancy (inferred and ground truth) each minute of the day over a three week period for Home B in the Smart* dataset [2].

home is occupied each minute of the day for both the ground truth occupancy and our inferred occupancy. For this home, our occupancy detection technique works well, and the home has a highly regular pattern of usage, making it a good candidate for a static thermostat schedule. Using this histogram, we can directly compute a schedule that optimizes for a specific miss time, waste time, or mismatch time. For example, we can specify a miss or waste time, and compute the schedule that minimizes the waste and miss time, respectively. We can also compute the schedule that minimizes mismatch time, i.e., the sum of miss and waste time.

The basic algorithm sorts the histogram by the probability of occupancy for each given time period. We may then specify a target average daily miss time and compute the schedule that minimizes the waste time. We start with the period with the lowest probability of occupancy and compute average miss time over this period assuming that we turn off the HVAC system (or shift it to the setback temperature). We can compute the average daily miss time directly from the histogram by simply multiplying the length of the time period by the probability of occupancy. We then compute the average daily miss time over the period with the next smallest probability of occupancy, assuming that the HVAC system is off during this period, and increment our total average daily miss time. We continue iterating in sorted order until the total accumulated miss time reaches our target. Since the HVAC system is off during these periods, it contributes no waste time, as waste time only accrues when the HVAC system is on (and the home is unoccupied).

Computing the minimum miss time for a target waste time follows the same procedure, but starts with the period with the largest probability of occupancy and iteratively computes the per-period waste time (and increments the total waste time) assuming the HVAC system is on. Computing the optimal mismatch time works in the same way, but computes the mismatch time for each period, and stops when the mismatch time increases after any iteration. Computing the mismatch time may proceed in either sorted order, i.e., largest to smallest or vice versa. If the occupancy information is accurate, then the algorithms described above are optimal with running time linear in the size of the input data, which includes a day's worth of minute-level power data. Constructing the histogram is also linear in the size of the input data, which we generally limit to the previous four months due to changes in occupancy across seasons. Pre-conditioning homes before occupants come home is necessary to maintain user comfort. iProgram can incorporate such pre-conditioning times by adjusting its schedules to account for the time required by the HVAC system to return the home from the setback to the setpoint temperature.

Note that our schedules are not limited to a single contiguous period of non-occupancy each day, as in prior work [13], and may result in multiple periods of non-occupancy. For example, if a user regularly comes home for lunch, the schedule would indicate the increase in occupancy probability during lunchtime, which would affect position of the lunchtime period in the sorted order above.



(a) (b)

Figure 4: iProgram screenshots, including the inferred occupancy (a) and generated thermostat-specific schedule (b).

3.3 Scheduling Extensions

We extend iProgram's basic algorithm in multiple ways.

Different Thermostat Types. We can use the same algorithm as above to compute schedules for different types of programmable thermostats, such as 5-2-day or 7-day, by simply considering different datasets. For a 7-day thermostat, we run the algorithm separately for datasets with all Mondays, all Tuesdays, etc. to compute a separate Monday schedule, Tuesday schedule, etc. Likewise, for a 5-2-day thermostat, we partition the data into two sets—for weekdays and weekends—and separately compute schedules for each.

Sleeping Schedules. iProgram may also separate sleeping from non-occupancy in its schedules. The sleeping period differs from non-occupancy in that homes may have a different thermostat setting when the home is occupied and sleeping versus not occupied. For example, homes may set bedroom heating or cooling zones in multi-zoned systems based only on the sleeping versus not sleeping periods, irrespective of occupancy (since bedrooms may not be occupied during the day, even when the home is occupied). iProgram assumes the sleeping period is the longest period of nighttime occupancy (between 7pm and 9am).

Dynamic Learning. Our algorithm above computes static thermostat schedules for different types of programmable thermostats. However, schedules may change over time. For example, homes' pattern of occupancy generally changes with each season due to changes in the climate or based on the academic calendar, i.e., children are out of school over the summer. We have extended our approach above to be more dynamic by recomputing schedules based on new data. In our case, we use a moving average approach that gives more weight to more recent data. To do so, we recompute the histogram above each day using a weighted measure of occupancy that discounts older occupancy data. In our prototype, we apply this discount on a monthly basis, since we are primarily focused on adapting to seasonal changes. Thus, to compute the histogram, we give full weight to occupancy data within the past month, and discount older data by a factor $\alpha < 1$. While a recomputed schedule would need to be manually (re)programmed by the user, which implies it should be infrequent and only capture large changes in the schedule, the process may be automated for newer programmable thermostats with WiFi capabilities.

4. IPROGRAM IMPLEMENTATION

We implement iProgram as an open web service that enables users to upload their power data and select their type of thermostat and then visualize their occupancy patterns, as well as generate a custom thermostat schedule. The architecture consists of six modules: a profile manager, storage engine, occupancy analyzer, schedule generator, visualization engine, and API manager.

The profile manager handles account creation, user authentication, and user meta-data. The profile manager interacts with the storage engine to store and retrieve information on user profiles, as

well as power consumption data. Users may upload power data directly as CSV files or provide a URL for a third-party meter that stores the data. We currently support eGauge power meters, but intend to add additional third-party meters in the future, such as TED. The occupancy analyzer implements occupancy detection and schedule generation algorithms from the previous section. The module stores the discretized occupancy information in the storage engine. The occupancy information is then read by the scheduler generator, which is capable of generating schedules for 1-day, 5-2-day, and 7-day programmable thermostats. The schedule generator supports the different scheduling algorithm variants, which either specify a target miss/waste time and minimizes waste/miss time, respectively, or optimizes for the mismatch time.

The visualization engine displays the occupancy information and the generated schedules to the user. Figure 4(a) shows a sample of the inferred occupancy information for a home, where green indicates the home is occupied and awake, red indicates it is unoccupied, and yellow indicates it is occupied and sleeping, and Figure 4(b) shows a sample thermostat schedule for a home. Finally, iProgram exposes an external REST API via the API manager, which provides a programmatic interface for networked thermostats to auto-program themselves using iProgram's schedules. The API exposes information in JSON format, and could be extended to offer If-This-Then-That (IFTTT) recipes to trigger automated actions for certain events, e.g., if the schedule changes.

iProgram's web service is built using Django, a popular Python-based web application framework. We use SciPy stack, which includes assortment of scientific computing libraries for Python, to process, store, and analyze power data. For time series decomposition, we use Statsmodels [25], a python library for time series analysis. We use dygraph, a Javascript graphing library for displaying occupancy data, and a sqlite3 database to store each user's profile, power data, occupancy information, and thermostat schedules.

5. EXPERIMENTAL EVALUATION

We evaluate iProgram using both data from over 100 homes across three public datasets, as well as results from a user study in 8 anonymous homes. Our datasets include the ECO dataset [3], the UMass Smart* dataset [2], and the Pecan Street dataset [22]. Each dataset includes different types of homes in different climates. The ECO dataset includes both 1Hz average power data for 6 homes in Switzerland, where 5 homes include binary occupancy data ranging from 25 to 66 days. Similarly, the UMass Smart* dataset includes 1Hz average power data for 3 homes in Massachusetts, and binary occupancy data for 1 month in two of the homes (Home A and Home B), as described in recent work [4]. Note that even though these datasets include 1Hz data, we apply our techniques to minute-level power data, since that is the highest resolution offered by utilities. Our method should also apply to even lower resolution data, as they will also capture the trend component, although the accuracy of the schedules may degrade at coarse resolutions. The Pecan Street dataset includes average power data every minute for 1,200 homes across Texas, Colorado, and California.

We have ground truth occupancy for the ECO and UMass datasets. Note that the ground truth is only used to determine the accuracy of our approach; *ground truth data is never used by the iProgram algorithm*, since iProgram does not require ground truth when computing schedules. While Pecan Street does not provide occupancy data, it does include average power data for a large number of circuits in each home. For our evaluation, we identify circuits in the dataset that only power interactive devices, e.g., lighting, televisions, microwaves, etc., and use our burstiness technique (without the initial time-series decomposition step to remove

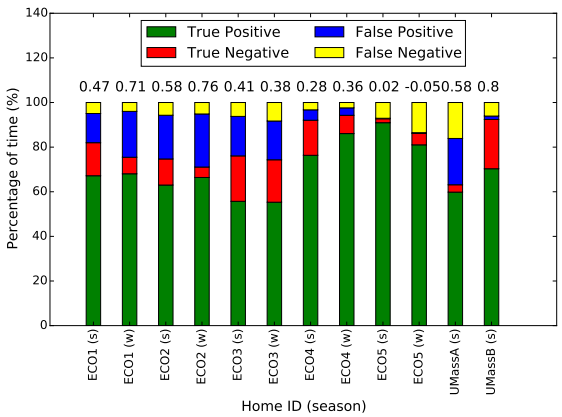


Figure 5: Graphical depiction of the confusion matrix for homes in the ECO and Smart* datasets, along with the associated MCC (atop each bar).

background loads) to infer occupancy, which we use as a proxy for ground truth occupancy. In prior work, we demonstrate that power events for circuits that only power interactive devices highly correlate with occupancy [4]. Importantly, the Pecan Street dataset also instruments the circuits related to HVAC energy usage, enabling us to identify the HVAC usage pattern and quantify its energy usage.

Since Pecan Street uses a consistent nomenclature for labeling circuits in homes, we choose circuit labels with a high probability of only powering interactive loads, and only select homes that have i) more than five dedicated interactive circuits, ii) a central HVAC system, and iii) a normal occupancy rate in the range of 60-90%. Based on this criteria, we select 100 homes from the Pecan Street dataset. While our proxy for ground truth occupancy may result in more false negatives, i.e., where occupants are home but not using any interactive devices, than the actual ground truth data, evaluating iProgram across 100 homes gives an indication of its flexibility for different types of homes with a range of occupancy patterns.

5.1 Occupancy Detection Accuracy

Figure 5 shows the performance of our occupancy detection technique on each home in the ECO and UMass Smart* datasets across their respective time periods. The figure graphically depicts each possibility in the confusion matrix, where the bottom two (red and green) portions of each bar represent the accuracy, and the number atop each bar is the MCC. The accuracy is in the range of 75%-95%, while the MCC ranges from near 0 (or akin to random guessing) to 0.76. Home 5 from the ECO dataset demonstrates why accuracy is not a good measure of performance: since the home's occupancy rate is high, our accuracy is also high (85-90%), but the MCC indicates our technique performs similar to random guessing. In contrast, Home A in the Smart* dataset yields the lowest accuracy, largely due to its low occupancy rate, but third highest MCC. Note the occupancy rate is $TP + FN$ (the green and yellow bars).

Overall, our results on the ECO dataset compare favorably with prior work [16]. In some cases, our MCC results are better than the best approach in prior work, e.g., ECO2 (winter), ECO3 (winter), ECO4 (winter), and in some cases they are slightly worse. The only notable deviation is ECO5, where our accuracy results are in line with prior work, but our MCC results are much worse than the best technique, e.g., an SVM. However, our technique uses the coarser minute-level data offered by current smart meters, rather than the second-level data used in prior work. In addition, iProgram's technique, which does not use training data, compares favorably to the best option out of multiple techniques, e.g., based on threshold-

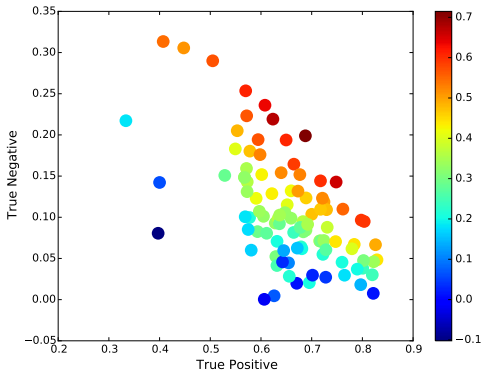


Figure 6: Scatterplot of TP and TN values for homes in the Pecan Street dataset. The color of each dot indicates the MCC.

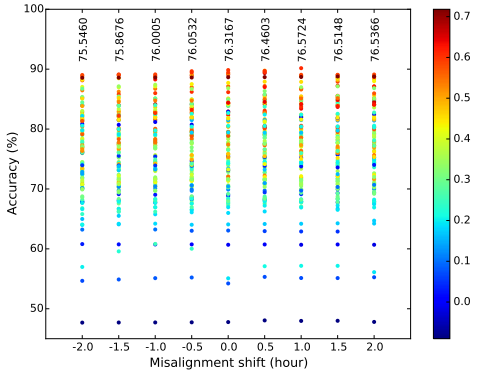


Figure 7: Accuracy of occupancy detection on HVAC systems that are misaligned with occupancy on the Pecan Street datasets. The color of each dot represents each home's MCC with the average MCC labeled for each value on the x-axis.

ing, kNN, SVM, and HMMs, that require training data. However, since there is little consistency in prior work as to which training-based technique is best across homes (or even seasons within the same home) in the ECO data, there is no way to choose the best training-based technique *a priori* without empirically comparing results from each one. Thus, we view iProgram's targeted occupancy detection approach, which does not require training data and is directly applicable to smart meter data (at smart meter data resolutions), as an advance in the state-of-the-art.

Similarly, Figure 5 shows the performance of iProgram's occupancy detection technique on each home in the Pecan Street dataset. Since there are over 100 homes, we use a scatterplot of the TP and TN values for each home, where the color indicates the MCC and the top right portion of the graph indicates the highest accuracy. The graph shows that our technique works well for many homes, although 15 of the homes are blue, indicating a low MCC. The result illustrates that occupancy detection from smart meter data is not perfect. Of course, accuracy and MCC are occupancy-centric metrics, while iProgram's goal is to improve HVAC schedules and save energy. As with Home 5 in the ECO dataset, in many cases, homes with low MCCs also exhibit high occupancy rates, e.g., >80% where thermostat scheduling is not a challenging problem.

We next show that iProgram's occupancy detection technique is robust to HVAC systems that are not aligned with occupancy. To demonstrate this, we intentionally shift the HVAC system usage in the smart meter data by different amounts up to 2 hours forward and backward in time to misalign it with occupancy, and then detect occupancy on the resulting traces. Figure 7 shows the results, where

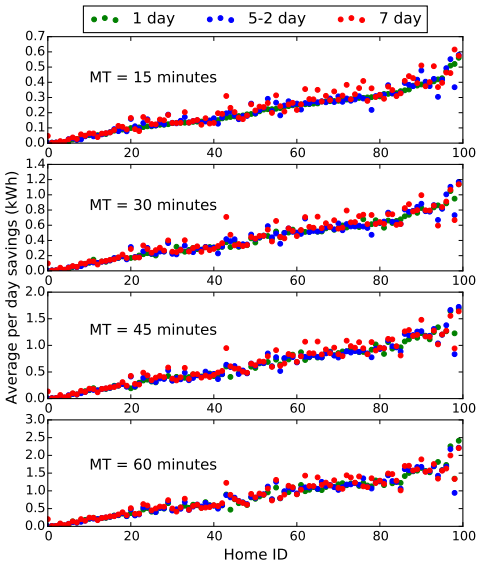


Figure 8: Estimated energy savings for different target miss times for each home in the Pecan Street dataset.

accuracy is on the y-axis, and the color of each dot represents each home's MCC with the average MCC labeled for each value on the x-axis. The graph shows that the magnitude of misalignment in the HVAC system has little effect on the accuracy of occupancy detection. We do not plot the ECO dataset, since none of its homes have a central HVAC system labeled in the data.

5.2 Thermostat Scheduling Performance

Figure 8 shows the estimated average daily energy savings each home in the Pecan Street dataset could achieve over 6 months using iProgram with different target miss times. Each point on the x-axis represents the estimated energy savings for a home, where we order homes by their energy savings when using a 1-day schedule. Here, we estimate the energy savings assuming the HVAC system uses iProgram's schedule; we then count any energy consumed by the HVAC system during an off period in the schedule towards our savings. Of course, the HVAC system may occasionally turn on during unoccupied periods to maintain the setback. In this case, we assume the setback is sufficiently deep that occupants are not away long enough for the temperature to reach it.

The graph shows that, as expected, a 7-day schedule achieves slightly more energy savings than a 5-2-day schedule, which in turn achieves slightly more savings than a 1-day schedule. However, the savings relative to a 1-day schedule, while significant in some homes, are not significant on average. Also as expected, the greater the acceptable target miss time (and the lower the tolerable comfort level), the more energy savings the homes achieve. The maximum energy savings (with a miss time of 60 minutes) is near 2.5kWh per day, which is over 10% of a typical U.S. home's average daily energy usage. Overall, the average daily energy savings is in the range of 0.25 kWh (for a 15 minute miss time) to 1.0kWh (for a 60 minute miss time), which represents a 1-5% energy reduction in an average U.S. home. However, the strength of our approach does not lie in achieving large energy savings in a small number of homes, as with smart thermostats, but rather in being immediately applicable to saving energy across a large number of homes.

We also compare iProgram's schedule to a default 8am-6pm thermostat schedule. We find that it reduces the mismatch time by a

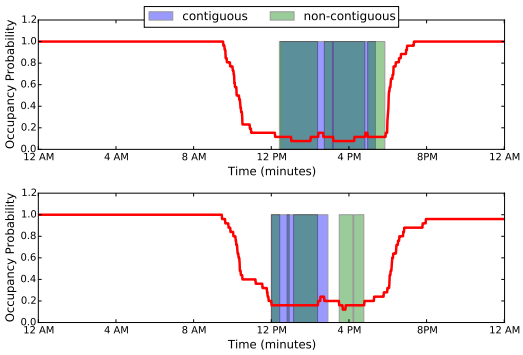


Figure 9: Illustration of the benefits of a non-contiguous thermostat schedule with multiple unoccupied periods.

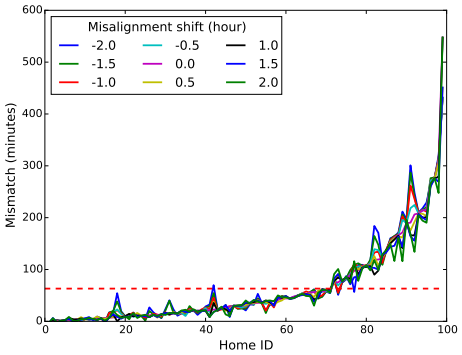


Figure 10: The mismatch time for homes in the Pecan Street dataset with different shifts in the HVAC system usage.

median value of 44.28 minutes (out of the 100 homes) for a weekday schedule, with a median deviation of 30.76 minutes off the optimal schedule (assuming perfect occupancy). In this case, iProgram would yield a daily energy savings of 0.42kWh on average across the 100 homes. Note that this estimated energy savings is conservative, since most Pecan Street homes are already highly efficient, and have little room for improvement. The Pecan Street participants have volunteered to have their homes instrumented by Pecan Street and are highly aware of their energy usage. Thus, we expect the energy savings possible in a typical U.S. home is likely to be much higher.

iProgram is more general than prior work [13] on computing static thermostat schedules based on occupancy data in that it can compute non-contiguous schedules that include multiple distinct unoccupied periods. Figure 9 illustrates such a non-contiguous schedule. Here, the red line depicts the probability of occupancy at each point during both a typical Monday (top) and Friday (bottom), the blue region depicts the unoccupied period for the optimal contiguous schedule, which is restricted to a single unoccupied period, and the green region depicts the unoccupied periods of the optimal non-contiguous schedule. On both days, there is a slight increase in the occupancy probability near the middle of the day at lunchtime, presumably resulting from the occupants coming home for lunch. Our optimal non-contiguous schedule recognizes this and turns on the HVAC system (indicated by the lack of the green regions when the probability rises), while the contiguous schedule keeps the HVAC system off (indicated by the remaining blue region when the probability rises). The optimal non-contiguous schedule reduces the overall miss time of the resulting schedule by 20 minutes across both days compared to the optimal contiguous schedule.

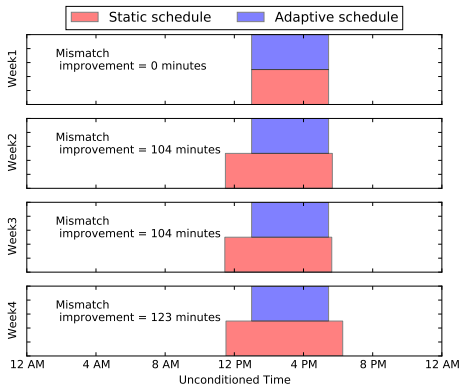


Figure 11: iProgram's schedule adapts as occupancy changes.

Figure 10 shows the optimal mismatch time for homes in the Pecan Street dataset, where we shift the HVAC system usage from 0 to 2 hours to misalign it with occupancy. Our results show that iProgram's schedules are robust to such misalignments, as the mismatch time across homes does not vary significantly with the shift in HVAC usage. The graph also shows that the average optimal mismatch time for the Pecan Street data set is near 60 minutes, although the average is affected by a few homes with very large mismatch times; 75% of the homes have a mismatch time below 60 minutes and 50% of them have a mismatch time <30 minutes.

Figure 11 illustrates iProgram's ability to adapt to changes in a the pattern of occupancy. In this case, iProgram computes a schedule using one month of data from a representative home. We then swap every Wednesday and Sunday for the next month to simulate a change in working hours. The graph then shows two schedules: static, where the schedule is generated using the first month's occupancy, and adaptive, where schedules are generated using a moving window of 1 month. In the first week, both static and adaptive approaches derive the same schedule as they are using data for the same time period. As shown, iProgram adapts quickly to the changed schedule with new data available every week.

5.3 User Study

To validate our algorithm in a real world setting, we conducted a user study, which asked participants to rate the schedules generated by iProgram for their respective homes. The objective of the study was to i) get user feedback on iProgram's schedules, ii) better understand user interactions with their thermostats, and iii) validate scheduling anomalies identified by iProgram. We worked with a local utility company in the northeast U.S. to solicit 8 anonymous users for the study. Each user consented to analysis of their smart meter data, which was anonymized to preserve the privacy of the participants. In this case, the data for each home covered a 3 month period with average power data recorded every 5 minutes. We used iProgram to derive a 7-day schedule, and then presented a customized questionnaire to each participant that asked them to rate their schedule on a scale of 1 (does not match my occupancy pattern) to 5 (perfectly matches). In addition, the questionnaire included questions about the type and usage of the home's thermostat, and the weekly work patterns of the home's occupants. We also included customized questions to corroborate any anomalies in the schedule derived by iProgram. Such anomalies included long periods of low electricity usage over multiple days, a repeating unoccupied period on one specific day of the week, etc.

Based on the results, we discovered each participant had a programmable thermostat in their home, but all the homes manually adjusted the thermostat without programming it. Most the partic-

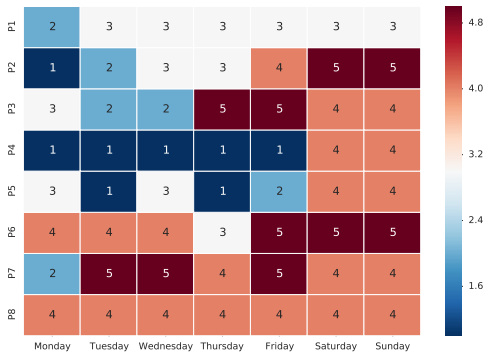


Figure 12: User ratings of iProgram's thermostat schedules for different days of the week. Over 73% of the daily schedules have good ratings of 3 or higher.

ipants work patterns followed a 5-2-day schedule, although one home included a retired couple with a much higher rate of occupancy. iProgram was also able to correctly identify a majority of the scheduling anomalies discovered during the analysis. Figure 12 shows the participants' ratings of iProgram's schedule for different days of the week. The figure shows that 73% of the schedules and derived occupancy patterns got ratings of 3 or higher. In addition, participant P4, which was the retired couple, gave a poor rating for weekdays. In this case, the occupants were frequently home on weekdays but did not consume much electricity, so iProgram identified these periods as unoccupied when generating its schedule.[2] Finally, 5 of the participants had an average rating above 3, and 3 of the participants had an average rating above 4 across all days.

6. CONCLUSION

In this paper, we present iProgram, a system for inferring smart schedules for programmable thermostats from smart meter data. iProgram cannot directly apply prior work on occupancy detection from smart meter data due to its application context, which does not provide access to training data and has a particular emphasis on inefficient homes with misaligned thermostats. Thus, we design a more targeted occupancy detection approach, which leverages time-series component analysis to isolate the burstiness of interactive loads. In addition, we show how to translate inferred patterns of occupancy into an optimal static thermostat schedule. We implement iProgram as a web service, and evaluate it across data from over 100 homes, as well as conduct a user study to verify its efficacy in real homes. As part of future work, we plan to work with a local utility to include iProgram's suggested thermostat schedules in real energy bills and as part of energy audits.

7. ACKNOWLEDGEMENTS

This research was supported by NSF grants IIP-1534080, CNS-1405826, CNS-1253063, and the Massachusetts Department of Energy Resources.

References

[1] Y. Agarwal, B. Balaji, S. Dutta, R. Gupta, and T. Weng. Duty-cycling Buildings Aggressively: The Next Frontier in HVAC Control. In *IPSN*, April 2011.

[2] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht. Smart*: An Open Data Set and Tools for Enabling Research in Sustainable Homes. In *SustKDD*, 2012.

[2]We have refined our approach to to better capture burstiness and to interpolate across detected bursty periods.

[3] C. Beckel, W. Kleiminger, R. Cicchetti, T. Staake, and S. Santini. The ECO Data Set and the Performance of Non-Intrusive Load Monitoring Algorithms. In *BuildSys*, November 2014.

[4] D. Chen, S. Barker, A. Subbaswamy, D. Irwin, and P. Shenoy. Non-Intrusive Occupancy Monitoring using Smart Meters. In *BuildSys*, November 2013.

[5] D. Chen, D. Irwin, P. Shenoy, and J. Albrecht. Combined Heat and Privacy: Preventing Occupancy Detection from Smart Meters. In *PerCom*, March 2014.

[6] S. Clark, B. Ransford, and K. Fu. Potentia est Scientia: Security and Privacy Implications of Energy-Proportional Computing. In *HotSec*, August 2012.

[7] Ecobee. `http://ecobee.com`, January 2015.

[8] U.S. Energy Information Administration, Frequently Asked Questions, Residential Energy Consumption Survey (RECS). `http://www.eia.gov/consumption/residential/data/2009/`, 2009.

[9] U.S. Energy Information Administration, Most Homes have Central Thermostats on Heating and Cooling Equipment. `http://www.eia.gov/todayinenergy/detail.cfm?id=14771`, January 28th 2014.

[10] V. Erickson, M. Carreira-Perpinan, and A. Cerpa. OBSERVE: Occupancy-based System for Efficient Reduction of HVAC Energy. In *IPSN*, April 2011.

[11] V. Erickson and A. Cerpa. Occupancy Based Demand Response HVAC Control Strategy. In *BuildSys*, 2010.

[12] K. Fehrenbacher. 75% of U.S. Electric Meters will be Smart by 2016. In *Gigaom Research*, March 5th 2012.

[13] G. Gao and K. Whitehouse. The Self-programming Thermostat: Optimizing Setback Schedules based on Home Occupancy Patterns. In *BuildSys*, November 2009.

[14] J. S. John. 50 Million U.S. Smart Meters and Counting. In *GreenTech Grid*, September 16th 2014.

[15] J. Kelso, editor. *2011 Buildings Energy Data Book*. Department of Energy, March 2012.

[16] W. Kleiminger, C. Beckel, T. Staake, and S. Santini. Occupancy Detection from Electricity Consumption Data. In *BuildSys*, November 2013.

[17] J. Kolter and M. Johnson. REDD: A Public Data Set for Energy Disaggregation Research. In *SustKDD*, August 2011.

[18] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse. The Smart Thermostat: Using Occupancy Sensors to Save Energy in Homes. In *SenSys*, November 2010.

[19] Honeywell Lyric. `http://lyric.honeywell.com`, January 2015.

[20] Nest. `http://nest.com`, January 2015.

[21] M. Nevius and S. Pigg. Programmable Thermostats that Go Berserk? Taking a Social Perspective on Space Heating in Wisconsin. *ACEEE Summer Study on Energy Efficiency in Buildings*, 8, December 2000.

[22] Pecan Street. `http://www.pecanstreet.org/`.

[23] T. Peffer, M. Pritoni, A. Meier, C. Aragon, and D. Perry. How People use Thermostats in Homes: A Review. *Building and Environment*, 46(12), December 2011.

[24] J. Scott, A. Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar. PreHeat: Controlling Home Heating Using Occupancy Prediction. In *Ubicomp*, September 2011.

[25] J. Seabold and J. Perktold. Statsmodels: Econometric and Statistical Modeling with Python. In *Scipy*, 2010.

[26] S. Shakkottai and N. Brownlee. A Study of Burstiness in TCP Flows. In *PAM*, March 2005.