Drapes: A Holistic Approach to Predictive Heating for Smart Home Applications

---

A Thesis

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Master of Science (Computer Science)

by

Andrew Frye

December 2014

Approval Sheet

This thesis is submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

Andrew Frye

This thesis has been read and approved by the Examining Committee:

Kamin Whitehouse, Adviser

Jack Stankovic, Committee Chair

Worthy Martin

Kevin Sullivan

Accepted for the School of Engineering and Applied Science:

James H. Aylor, Dean, School of Engineering and Applied Science

December 2014

Contents

List of Tables

List of Figures

Abstract

Home air conditioning is the cause for a substantial portion of total energy usage in many countries. Many home conditioning systems spend much of their energy heating, cooling, or maintaining comfortable temperatures when unoccupied. Allied with an accurate occupancy prediction system, smart conditioning systems could save homeowners a significant amount of money by heating only when there are occupants present. A model to predict occupancy must determine which features of a home are highly correlated with occupancy and which are not. Previous models have used features from a given room or zone, but ignore the relationship between rooms in a home. In this paper, we introduce Drapes, a predictive conditioning system that makes use of a holistic view of a residence to learn occupancy patterns and create heating schedules. We explore Drapes by analyzing seven home occupancy data sets ranging in size from twelve days to eighty. Our experiments and analysis show that Drapes is able to infer zone level occupancy and condition the home accordingly such that, when compared to state of the art heating algorithms without reactive heating components, occupants' discomfort is reduced by approximately 40%, and energy waste is reduced by 15% on average.

Chapter 1

Introduction

As of 2011, 22% of the total U.S. energy consumption belonged to residential buildings [1]. At $35.22 per MWh for natural gas and $118.83 per MWh for electricity, 22% of the total energy consumption translates to excessive spending [2]. As one of the leading sources of energy consumption, there is high incentive for technologies that aid in its reduction. In 2009, 48% of the energy used in U.S. homes went directly to space heating and air conditioning [1]. Our aim is to significantly reduce this total.

One development that could reduce the total energy consumed by residential heating and cooling is a smart HVAC system. This system could save energy by predictively heating only the rooms or zones which will be occupied.

Inherent with this system is a tradeoff between energy savings and occupant comfort. Clearly, a system which predicts that a room will always be empty can save a lot of energy by never heating or cooling that room; but, as this strategy maximizes miss time (the amount of time occupants spend in an unconditioned room), the occupants will rarely be comfortable. Alternatively, a system which predicts that a room will always be occupied can always keep its occupants comfortable by maintaining a temperature setpoint; but, as this strategy maximizes waste time (the amount of time the system spends conditioning a room when no one is present), the occupants will most likely disapprove of that system because of its high cost.

To address this tradeoff, the system must not only learn occupancy patterns in some way, but it must also be modifiable to suit individual homes and occupant preferences. Some may prefer a more aggressive system to increase comfort, while others may prefer a less aggressive system to conserve energy and reduce energy bills.

The first piece of the puzzle is to sense and learn occupancy patterns. Sensing occupancy is a difficult problem. In this work, we make use of a variety of occupancy sensing techniques and analyze the results of

each. Some occupancy sensing systems are naive and assume occupancy whenever motion is detected by off the shelf motion sensors. Others are a bit more complex and use RFID tags and readers placed by doorways to sense entry and exit events. Although some systems have proven to be more accurate then networks of motion sensors [10], at the present time, there is no reliable way to gather ground truth occupancy data. The best we can do is to learn patterns of `observed occupancy` as reported by the occupancy detection system. In following sections, we offer some comparisons of different occupancy observation systems, and discuss insights pertaining to each.

After gathering observed occupancy, we must learn occupancy habits in some fashion. Other works have attempted to address this issue in various ways [3, 4, 5, 6, 7, 11], but they fail to address the home as a set of related components. We hypothesize that the home is a collection of related components (rooms/zones) that can be observed individually and used collectively to learn patterns of occupancy. A model that is to learn such patterns to control heating must consider the home from a holistic perspective.

The focus of this work, is Drapes, a predictive conditioning system that uses a holistic view of the home to learn occupancy patterns and create heating schedules. Whereas state of the art approaches use information only from the zone for which a prediction is being made [7], our solution uses a holistic approach in the sense that we also make use of data from other rooms and zones in the home when making predictions for any individual zone. This approach is shown to yield greater benefits when making zone level predictions as opposed to whole house level predictions, but still shows promise for the latter and will be the subject of future research. Our contributions include the following:

- Evidence to support the hypothesis that homes are holistic, and we must take advantage of the relationships among rooms in order to accurately predict future events
- A new holistic model for predicting future conditioning events in a home that is robust and adjustable to incorporate future features found to be indicative of room occupancy
- New residential occupancy data sets from four studies in Charlottesville, VA.

Using these data sets, we find that Drapes is able to reduce the total miss time by an average of approximately 33% under that of current approaches, increasing user comfort, and is able to reduce waste time by an average of approximately 15%, resulting in less wasted energy.

A home conditioning system can be spoken of in terms of both heating the home during cooler months, and cooling the home during warmer months. For simplicity, we will speak in terms of the former for the duration of the paper.

Chapter 2

Related Work

There are many approaches to controlling home heating systems to save energy. For home heating systems that utilize occupancy prediction algorithms, there are two main types. The first is exemplified by algorithms that attempt to provide home level occupancy prediction (whether occupants will be home or away from home)[3, 4, 5]. The second type is demonstrated by algorithms that attempt to perform finer grained occupancy prediction by predicting room level or zoned occupancy[6, 7]. These algorithms typically involve some method of detecting occupancy in each zone (motion sensors, cameras, etc.), building a model of occupancy patterns, and making predictions of future occupancy events accordingly. Our solution exemplifies the latter of the two approaches.

One traditional approach is to use a setback schedule. With this approach, occupants provide the heating system with a schedule in which they will be home or away from home. When occupants are scheduled to be home, the system heats the house accordingly. Conversely, when occupants are scheduled to be away from home, the system maintains a cooler setback temperature to save energy. The system then knows when to begin heating the home to prepare for the occupants return. This approach has some serious flaws. For example, it has been shown that occupants are often poor judges of their own schedules and often set setback schedules for when they will be home, and vice versa[3]. This results in frustrated users who end up leaving the system on all the time, wasting more energy than those who don't use such thermostats.

Another approach is known as the Smart Thermostat [4]. This approach uses a Hidden Markov Model to predict whole house level occupancy. Yet this approach was not evaluated for a zone level heating system, and because of the nature of Hidden Markov Models, the Smart Thermostat would require a significant amount of training data to be able to accurately model observable states for zone level conditioning.

Krumm and Brush [3] developed new algorithms using GPS data which involve probabilistic scheduling

and drive time algorithms to predict when a person will be at home or away from home. Using the GPS data, they build a schedule that gives the probability of an occupant being home given the time of day and the day of week. Combined with this, they use GPS data to determine how far occupants are from home and how long it could potentially take for them to arrive at home. This "drive time algorithm" improves the performance of the probabilistic schedule and allows them to achieve accuracy as high as 70% when predicting whole house occupancy. However, this attempt to predict occupancy and heat the home accordingly will not take full advantage of systems that have the capability to control temperatures at the zone level. This will result in energy waste when zones within a home are heated unnecessarily.

Mozer et al. [5] use a neural network in an approach they call Neurothermostat. This approach uses a hybrid occupancy predictor that leverages a daily schedule and a neural network. Again, this approach does not focus on zone level occupancy prediction and therefore will result in wasted energy when unoccupied zones are unnecessarily heated. Furthermore, the Neurothermostat required five months of training data to accurately predict occupancy. Ideally, an occupancy prediction approach will be able to attain accuracy within only a few days or weeks of deployment. This makes the Neurothermostat infeasible for a smart HVAC system.

Another approach by Erickson and Cerpa [6] uses Markov Chains to predict zone occupancy. With this model, each state of the Markov Chain represents the current states of all rooms. However, this model takes into account the maximum occupancy of each room, and the current state of each room is defined by how many occupants are in a given room. Therefore, the number of states required by this model grows exponentially as the number of rooms increases. To compensate for this, they reduce the Markov Chain only to observed states. But as the prediction accuracy improves only as more and more states are observed, it could take a very long time for this model to be effective.

The most recent approach to using occupancy prediction to control home heating and reduce energy is PreHeat [7]. This system senses occupancy by placing RFID tags on the house keys of all occupants to collect data about whether or not occupants are home. They also place motion sensors in rooms to collect room level occupancy data. Occupancy for each zone is represented as a bit vector where each element represents a fifteen minute time window with a one indicating an occupancy event during that time interval, and a zero indicating no occupancy. Such bit vectors are built for each day that the system is installed. To predict future room occupancy, PreHeat uses a K-Nearest Neighbor algorithm (with value K = 5). It takes the elements of the bitvector created for the current day, and searches a library of bitvectors for the K most similar past days where similarity is defined by the hamming distance of the bit vectors from the first element through the element representing the current fifteen minute time window. Once the K nearest days are found, they are used to predict future occupancy based upon the mean of the binary values for the desired time

frame. However, PreHeat fails to consider the relationships that are inherent among rooms. By considering the state of only the room for which it is making a prediction, PreHeat sacrifices accuracy. PreHeat has shown improvement upon baseline control algorithms, such as a scheduled system, and will be our baseline for further analysis.

Drapes improves on these approaches by taking into account relationships that exist among different rooms or zones of a house, and by providing the capability to condition levels at the room or zone level, instead of only at the whole house level. It has also been shown to be able to reduce both miss time and waste time with only a short training period (several of our data sets contain only 12 days of data).

Chapter 3

Preliminary Work

Before developing Drapes and the current home conditioning system, results from early work which applied the same concepts proved promising.

Using the same intuition that the home is composed of many related zones, we developed and analyzed Circulo, a "just-in-time" hot water re-circulation system [8]. By using a similar approach as Drapes, and providing faucet usage data to machine learning algorithms from various faucets in a home, we successfully learned faucet usage patterns and recirculated hot water in pipes resulting in 30% less energy usage and minimal impact to user convenience.

Circulo uses a Naive Bayes model that incorporates 5 features:

- time of day
- day of week
- the amount of time that hot water was used in the past 15 minutes
- the amount of time that hot water was used in that past 60 minutes
- the amount of time that hot water was used in the past 120 minutes

These features demonstrate the need for a model that approaches the home from the perspective that each faucet is not independent of one another. Rather, through its use or non-use, each faucet is indicative of future water usage and the necessity of hot water re-circulation. These were encouraging results, reinforcing the idea that homes are holistic, and providing the driving force behind our exploration into an application of the same ideas to home occupancy prediction and air conditioning control.

Another early work, Percolator, aided in our realization of the power of decision trees. Percolator is a technique that predicts room occupancy by choosing among several occupancy models which range in

granularity. Percolator chooses which model to use based on the amount and type of training data collected. Hence, in early stages of deployment, when a limited amount of training data has been collected, Percolator uses a course grained model to make a prediction. Then, as training time increases, and Percolator collects more training data, it uses a finer grained model providing higher prediction accuracy.

Originally, Percolator used a simple threshold algorithm that predicted occupancy based on the total number of times it had seen future occupancy over the total number of times it had seen the home in its current state. Choosing finer granularity models based upon the total number of times a state had been observed appeared to perform better than using any of the models individually. We hypothesized that improving the performance of the individual models would also improve the performance of the system as a whole. We did this by modifying each model to use decision trees.

However, adapting each model to use a decision tree instead of a more naive threshold technique did not produce the expected results. While the performance of the individual models did, in fact, improve, the performance of Percolator did not. This was due to the fact that decision trees already utilize techniques (pruning and information gain) that allow them to select from the features provided them, a subset that performs well.

Thus, our attempts to manually select a model of appropriate granularity was already implicitly occurring via our use of decision trees. These results provided reassurance in our use of the decision tree machine learning technique.

The reader may wish to pursue a more detailed description of Percolator and its results. To do so, please reference Appendix A.

Chapter 4

Approach

People are creatures of habit and tend to follow regular occupancy patterns. For this reason, we hypothesize that such patterns can be modeled by features that encapsulate the entire home and exploit a home's interconnectedness. As an example of this, consider the relationship between a kitchen and dining room. When a kitchen is occupied as someone is preparing a meal, the dining room tends to be occupied shortly thereafter when they eat. We now outline the following features of a home that we explore:

- Time of Day (tod) - This feature represents the time of day in minutes. Because there are only 1440 minutes in a single day, this feature can take any integer value between 1 and 1440. PreHeat uses this feature inherently as each element in the PreHeat bit vectors map to a specific 15 minute time window during the day.

- Day of Week (dow) - This feature represents the specific day of the week (Sun, Mon, Tue, etc.). As such, it can assume any integer value between 1 and 7. PreHeat uses this idea at a coarser granularity as it uses observed weekends to predict Saturdays and Sundays, and observed weekdays to predict Mondays through Fridays.

- Room Occupied (roomOcc) - This is a binary value where 1 represents that the room is occupied, and 0 represents that the room is unoccupied. PreHeat uses this feature only for the zone for which it is trying to predict occupancy. Drapes, on the other hand, includes a duplicate of this feature for every zone in the house resulting in the cardinality of the feature space for this feature being equal to $2^{numberOfRooms}$. By doing this, we incorporate a holistic approach that allows Drapes to learn relationships among rooms that are commonly used at the same time.

- Whole House Occupied (wholeHouseOcc) - This is also a binary value where 1 represents that the house is occupied, and 0 represents that the house is unoccupied. This feature is derived from the Room State features. If the state of any room in the house is currently occupied, then this value is set to 1. Otherwise, it is set to 0.

- Duration of Room State (roomDur) - This feature represents the total amount of time in minutes that a room or zone has been in its current state (occupied or unoccupied). Knowing the duration of the current state allows the system to learn slightly more complex relationships. While Room State provides a snapshot at a specific time, Duration of Room State provides a way to see what has happened. For example, with only Room State, we can know that the Kitchen is currently occupied. However, with Duration of Room State, not only can we know that the Kitchen is occupied, but we also can know that it has been occupied for the past 30 minutes, and the model can infer, for example, that the meal preparation is nearly finished and the dining room is likely to be used soon. The value of this feature can be any positive integer. However, to simplify later calculations, we will assume the cardinality of the feature space for this feature is not greater than $5000^{numberOfRooms}$.

- Room Occupied by Person N / Whole House Occupied by Person N (roomOccByPN/ wholeHouseOccByPN) - Three of our data sets used the RFDoormat system to sense occupancy. For these data sets, the occupancy sensing system did not only provide information about room occupancy, but it also provided information about which occupants were present. Using this data, we are able to generate these features which, like Room Occupied, are binary features with 1 representing that the room is occupied by a particular occupant, and 0 representing that the room is not occupied by a particular occupant. Similar to Room Occupancy, Drapes includes a duplicate of this feature for every zone in the house, and for every occupant. Therefore, the cardinality of the feature space for these features are $2^{numberOfRooms*numberOfOccupants}$ and $2^{numberOfOccupants}$ respectively.

For our experiments with Drapes, we chose a subset of the aforementioned features such that we were able to explore the effect of a holistic model of home occupancy while at the same time preventing the model from becoming too complex. To do so, we used the features Time of Day, Day of Week, Room Occupied, and Whole House Occupied in our implementation of the Drapes algorithm. These features result in a relatively small cardinality of feature space while still providing a feature set comparable to PreHeat. While not using features such as Duration of Room State in order to reduce the cardinality of the feature space of the model and its complexity, these features still incorporate the holistic nature of the home.

4.1 Customized Models

One problem that arises after data has been observed is that of model generality. In other words, a set of features that accurately predict room occupancy in one home may not do so in another. Similarly, features may vary within the same home across rooms. Some rooms, such as the kitchen or living room, will be occupied much more frequently than others, such as the laundry room or bathroom. An occupancy model will gather data about such rooms much faster, and a finer grained model will be required to distinguish among the many instances while a coarser grained model will be needed to attain the same level of accuracy for a less frequented room.

To address this issue, we make use of decision trees: a machine learning technique that uses a tree like structure in which leaves represent target values we are trying to predict, and nodes represent various attributes in determining that target value. By their very nature, decision trees rank the attributes that provide the most information and place them higher in the decision tree structure. Then, they remove branches and features which are unnecessary through a method known as pruning. Combined, these two features of decision trees address the issue of choosing which attributes to select when making an occupancy inference.

Drapes uses many decision trees. Each is specific to a zone and a *prediction horizon* (the future time at which we are attempting to predictively condition).

4.2 Formal Definition of Drapes

At this point we provide a formal outline of the Drapes algorithm in Matlab notation as discussed in this document. It is broken down into two main components:

- Parsing Raw Data
- Learning from the Data and Making Predictions

4.2.1 Parsing Raw Data

To begin, we parse raw data files into matrix form representing the state of the home at any given minute. This file is known as the `stateMatrix`. It is defined as follows: `stateMatrix = [timestamp, roomXoccupied, roomXoccupiedByPN, wholeHouseOccupied, wholeHouseOccupiedByPN]` where `timestamp`, `roomXoccupied`, `wholeHouseOccupied`, `roomXOccupiedByPN`, and `wholeHouseOccupiedByPN` are each an `m x 1` vector where `m` is equal to the number of minutes in the study. Each cell of `timestamp` contains the Matlab datenum for the

respective minute of the study. The vector `roomXoccupied` exists for each room in the study, and `X` is replaced with the room number e.g., `room1occupied, room2occupied`, etc. Similarly, the vector `roomXoccupiedByPN` exists for each room and for each occupant where `X` is replaced by the room number, and `N` is replaced by the occupant ID. e.g., `room1occupiedByP1, room1occupiedByP2`, etc. The vectors `wholeHouseOccupied` and `wholeHouseOccupiedByPN` are the home level counterparts to `roomXoccupied` and `roomXoccupiedByPN`.

```
roomXoccupied(t) = 1 if occupancy is reported from the detection system at time
        t and 0 otherwise
roomXoccupiedByPN(t) = 1 if the detection system reports occupant N in room X at
        time t and 0 otherwise.
wholeHouseOccupied(t) = any(roomXoccupied(t))
wholeHouseOccupiedByPN(t) = any(roomXoccupiedByPN(t))
```

4.2.2 Learning from the Data and Making Predictions

Once the `stateMatrix` has been built, we can begin generating testing and training instances from it to build a classifier and an occupancy model. First, we must define several parameters that influence the construction of the model. The first variable, `predHoriz` is the number of minutes in the future for which we want to learn a pattern and make a prediction. The next two variables, `windowSize` and `percentOfWindow` are used to define heating events. The former is the size of the block of time, in minutes for which we will eventually be making predictions, and the latter is the fraction of `windowSize` that must be reported as occupied by the occupancy sensing system in order to classify the window as occupied for heating.

Finally, `featureSet` is any subset of the following list of features that may be used to train the machine learning model. All features are gathered from `stateMatrix`

- `dow`: Stands for 'Day of Week' and is an `m x 1` vector where `m` is the number of minutes in the study. Each cell contains the day of the week (e.g. 0 = "sun", 1 = "mon",2 = "tue",etc.) for the respective minute of the study.

  ```
  dow(t) = find(strcmp(datestr(stateMatrix(t,1),'dddd'),...
          {'Sunday','Monday','Tuesday',...}))
  ```

- `tod`: Stands for 'Time of Day' and is an `m x 1` vector where `m` is the number of minutes in the study. Each cell contains the time of day (in minutes) for the respective minute of the study. This value ranges from 0-1439.

```
tod(t) = str2num(datestr(stateMatrix(t,1),'MM')) +...
         str2num(datestr(stateMatrix(t,1),'HH'))*60
```

- `roomOcc`: Stands for 'Room Occupied' and is an `m x n` matrix where `m` is the number of minutes in the study and `n` is the number of rooms in the house. This value is taken directly from `roomXoccupied` vectors in `stateMatrix`.

```
roomOcc(t,:) = stateMatrix(t,[2:(number of rooms + 1)])
```

- `roomOccByPN`: Stands for 'Room Occupied By Person N' and is an `m x n` matrix where `m` is the number of minutes in the study and `n = (number of rooms)*(number of participants)`. This value is taken directly from `roomXoccupiedByPN` vectors in `stateMatrix`.

```
firstColumn = numberOfRooms + 2;
lastColumn = firstColumn + (numberOfRooms*numberOfParticipants) - 1
roomOccByPN(t,:) = stateMatrix(t,firstColumn:lastColumn)
```

- `roomDur`: Stands for 'Room Duration' and is an `m x n` matrix where `m` = the number of minutes in the study and `n` = the number of rooms. Each cell represents the duration (in minutes) of the current occupancy state for the corresponding room at that time.

```
prevState(t) = stateMatrix(t-1,2:numberOfRooms+1)
sameStateAsPreviousState(t) = prevState(t) == stateMatrix(t,2:numberOfRooms+1)
roomDur(t,find(sameStateAsPreviousState(t)) = roomDur(t-1,...
        find(sameStateAsPreviousState(t)) + 1;
roomDur(t,find(~sameStateAsPreviousState(t)) = 0
```

- `wholeHouseOcc`: Stands for 'Whole House Occupied' and is an `m x 1` vector where `m` = the number of minutes in the study. This value is taken directly from the `wholeHouseOccupied` vector in `stateMatrix`.

```
wholeHouseOccIdx = numberOfRooms + 2 + (numberOfRooms*numberOfParticipants)
wholeHouseOcc(t) = stateMatrix(t,wholeHouseOccIdx)
```

- `wholeHouseOccByPN`: Stands for 'Whole House Occupied By Person N' and is an `m x n` vector where `m` = the number of minutes in the study and `n` = the number of participants. This value is taken directly from the `wholeHouseOccupiedByPN` vectors in `stateMatrix`.

```
firstWholeHousePNIdx = numberOfRooms + 2 ...
        + (numberOfRooms*numberOfParticipants)-1 + 2
lastWholeHousePNIdx = firstWholeHousePNIdx + numberOfOccupants - 1
wholeHouseOccByPN(t) = stateMatrix(t,firstWholeHousePNIdx:lastWholeHousePNIdx)
```

With the above parameters defined and set, we are able to construct training and testing instances and build the machine learning model that will eventually make heating predictions. We do that by first defining `observedOccupancy`. This is an `m x 1` vector where `m` is the number of minutes in the study and where a value of 1 means occupancy was observed, and a value of 0 means occupancy was not observed. This vector is taken directly from the `wholeHouseOcc` component of the `stateMatrix` when we wish to predict whole house occupancy. It is a copy of a `roomXoccupied` vector if we wish to instead predict room level occupancy.

With `observedOccupancy` defined, we are free to begin building testing and training instances for the machine learning model. Testing and training instances are placed in the `testingTrainingInstances` matrix, an `m x n` matrix where `m` is equal to the number of minutes in the study and `n` is equal to the number of feature columns defined by `featureSet`. The value of `testTrainInstances` varies depending on the feature set specified, but is always the simple concatenation of the specified features. For example, if the specified features are `dow, tod, wholeHouseOccByPN`, then

```
testTrainInstances(t) = [dow(t),tod(t),wholeHouseOccByPN(t,:)]
```

To accompany the testing and training instances, we must also define the associated labels. This is done via the `testingTrainingLabels` vector, an `m x 1` vector where `m` is equal to the number of minutes in the study. Each cell is a classification label that matches the corresponding testing/training instance row in `testingTrainingInstances`,and a value of 1 means the window is occupied, and a value of 0 means the window is unoccupied. In Matlab notation:

```
testTrainLabels(t) = length(find(observedOccupancy(t+...
        predHoriz:t+predHoriz+windowSize-1))) >= windowSize*percentOfWindow
```

Finally, we are able to send all training instances that belong to previous days off to the the machine learning model to construct the classification tree. Once returned, the testing instance created from the current state of the home is passed to the model yielding a prediction for a future window of time. The predictions are placed in the `predictedHeatingEvents` vector, an `m x 1` matrix where `m` is equal to the number of minutes in the study, and each cell represents whether or not the predictive component of the heating algorithm calls for a heating event for the corresponding minute. A value of 1 means the minute is

part of a window that will be predictively heated, and 0 means the minute is part of a window that will not be predictively heated. In Matlab notation:

```
trainIndices = 1:t-(t%1440)
testIndex = t
decTree = ClassificationTree.fit(testingTrainingInstances(trainIndices,:),...
        labels(trainIndices))
predictedHeatingEvents(t+predHoriz:t+predHoriz+windowSize) = predict(decTree,...
        testingTrainingInstances(testIndex))
```

Chapter 5

Experimental Setup

Data for these experiments was collected from homes in the Charlottesville, VA area. We also used two additional datasets provided by Washington State University's CASAS Smart Home Project [9]. Information outlining these datasets is displayed in Table 5.1.

# Residents	# Rooms	# Motion Sensors	# Days
2	9	RFID	12
2	9	RFID	12
2	9	RFID	12
2	7	14	78
1	4	36	49
1 + dog	7	16	66

Table 5.1: Data set information

For dataset 4, motion sensors were deployed in all rooms of each home, and data was collected over a period of several months. To ensure the validity of the data, we selected a period of sixty days for which we are confident the motion sensors were functioning properly. The motion sensors we used were X10 sensors. For datasets 1-3, we deployed and used the RFDoormat system for occupancy sensing, and used no motion sensors. For the remainder of the data sets downloaded from the CASAS Smart Home Project and from Microsoft Research, the types of motion sensors, and the number of motion sensors deployed in each room varied. The validity of this data is assumed over the entire time range for which the data was downloaded.

Room 1	Room 2	Room 3	..	Time
1	0	0	..	:00
1	1	0	..	:01
..	..	..	..	..
0	0	1	..	n

Table 5.2: Sample room occupancy matrix built from raw data. Each row represents the state of the home at a given time of day

5.1 Mapping sensors to zones

The most difficult barrier we encountered when using the additional data sets was the lack of clarity with regards to the motion sensor layouts. Few data sets had sufficient documentation to determine exactly what sensors were placed in what rooms, but most had an image file of the floorplan with sensor layouts. An example of such a layout is shown in figure 5.1. Looking at the figure, it becomes obvious that some sensors (such as sensor M026) are ambiguously placed. We are unsure of their direction or the room for which they sense movement. In such scenarios we were forced to ignore data supplied by those sensors. This results in fewer sensors per room, and therefore a decrease in our ability to reliably infer room occupancy. Reliably sensing room occupancy is key to the ability to learn occupancy patterns. As the RFDoormat system has been shown to produce higher validity data in terms of detecting occupancy [10], Drapes is able to perform better on such data sets as seen in the Results and Analysis sections.

5.2 Algorithm Implementations

Drapes and PreHeat were both implemented entirely in Matlab using its built-in support for decision trees (for Drapes) and the K-Nearest Neighbor algorithm (for PreHeat). To implement both algorithms, we begin by parsing the raw data to build state matrices where each row in the matrix represents the state of the home at a given minute of the day, and each column except the last (which contains the time) represents a zone or room. Table 5.2 provides an illustration of a room occupancy matrix.

From this matrix we can build training and testing instances used to construct the decision trees for Drapes. At each row (for every minute of every day) we can look at the current state to gather features such as Room Occupancy, Whole House Occupancy, Time of Day, and Day of Week. We can also look at previous states to gather information about the Duration of Room Occupancy. In similar fashion, we can look ahead at the states that were observed and gather the target values for the training instances (whether or not the zone was occupied at the specified prediction horizon in reference to the specific training instance). A more detailed description of this algorithm can be found in Section 4.2.

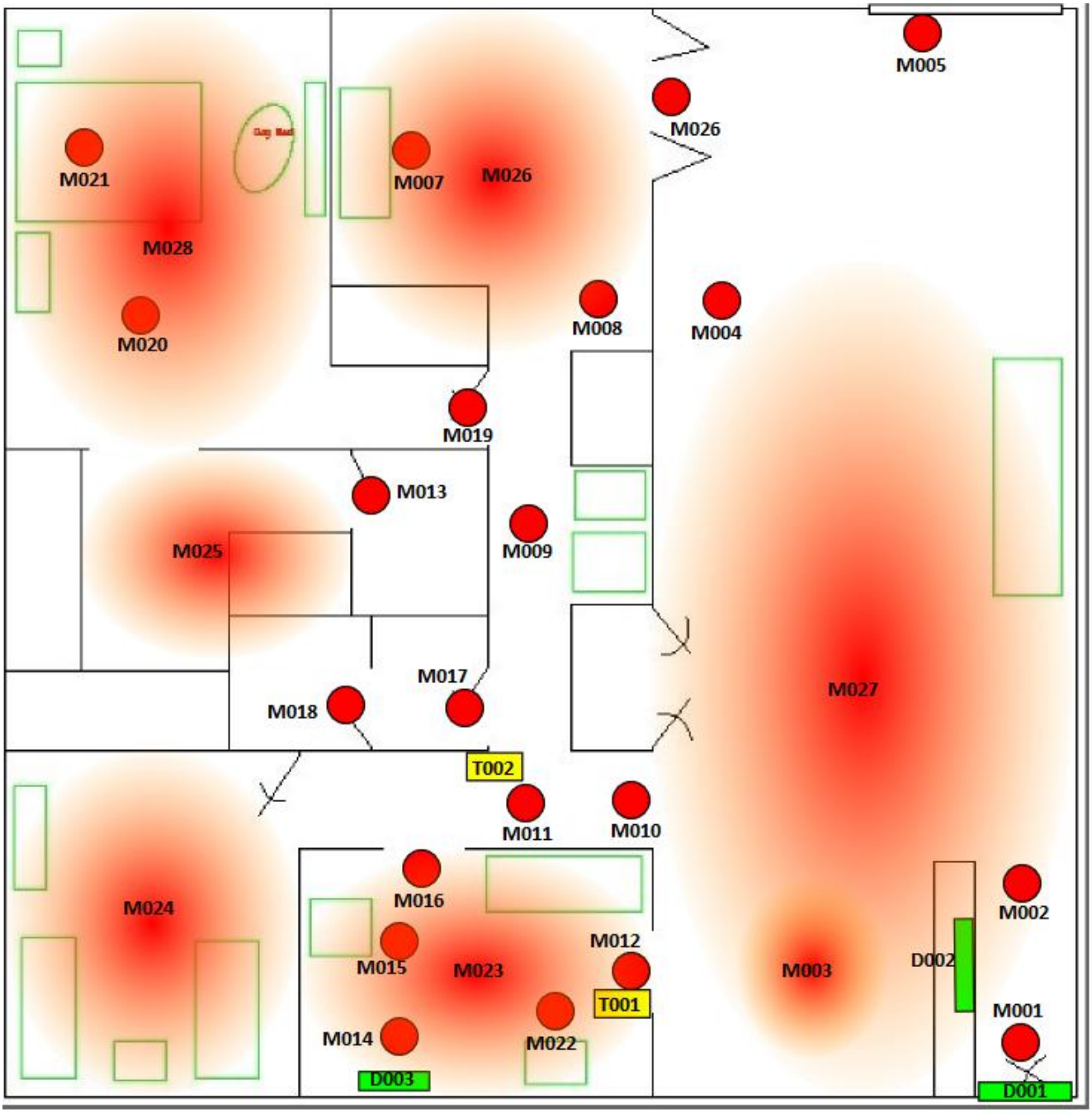


Figure 5.1: Floorplans and motion sensor layouts made it difficult to assign motion sensors to specific zones

5.3 Heating Events vs Occupancy

One of the main challenges when designing an occupancy prediction system, is that many rooms are much more difficult to predict than others. This may be true for many reasons, but it has been observed that it is particularly true for rooms for which there are fewer occupancy events or seemingly random occupancy events [7]. Examples of this include hallways or transition areas that are used to get from one zone to another, and

rooms such as bathrooms which are used mainly when occupants need to use the restroom (which can be unpredictable).

To solve this problem, PreHeat makes a prediction at the whole house level, and uses that outcome to condition such rooms. However, this can result in a significant amount of wasted energy as rooms that may only be occupied for minutes a day are continuously heated.

We have applied a different approach by defining what we have called *heating events*. Instead of attempting to predict occupancy for each minute of the day, we make predictions for continuous blocks of time. By doing so, we can filter out short periods of activity (e.g. when someone passes through a zone to get to another) and only condition at times when it is worth the cost.

To do this, we introduce two variables: *window size* and *percent of window*. Window size is the size of the continuous block of time for which we are attempting to predict conditioning events. The greater the window, the longer the conditioning system will remain on. Percent of Window is a variable that determines exactly how much total occupancy must have been observed in the window in order to classify it as occupied. For example, with a window size of 60 minutes and a percent of window value of 50%, Drapes would require at least 30 minutes of occupancy to be observed in the window in order to classify it as occupied.

Note that this technique can only be done after occupancy has already been observed, and can therefore not be used to classify the current state as occupied or not. This is simply a definition of a heating event from which we can observe and learn patterns of occupancy that are significant enough to heat.

PreHeat implicitly uses this technique. By separating the day into 15 minute windows, and classifying them as occupied if any motion was observed, they are essentially defining their heating events to have a window size of 15 minutes and a percent of window value of 6% (at least one minute of occupancy must have been observed to classify as occupied).

Figures 5.2 and 5.3 show the tradeoffs in terms of miss times and waste times when varying window size and percent of window size. These charts were generated by running the heating simulation several times for each parameter setting on the first 10 days of each dataset. In the first set of graphs, which are averages of results of a sensitivity analysis of all studies, it is clear that a shorter `windowSize` and smaller `percentOfWindowSize` yield lower miss time and higher waste time while a longer `windowSize` and higher `percentOfWindowSize` result in lower waste time and higher miss time. This is a fairly intuitive result as occupants are less likely to remain in the same room 100% of the time as the `windowSize` increases implying waste if continuously conditioned. Thus there is the ability for users to fine tune to their individual preferences depending on whether or not they would prefer to be more comfortable at the expense of using a bit more energy, or vice versa. For the remainder of our study, we chose a parameter setting with `windowSize` equal

to 30 minutes and `percentOfWindowSize` equal to .4. We found these parameters to be a reasonable middle ground on which to conduct the remainder of our experiments and analyses.

5.4 Implementing PreHeat

We use the same room occupancy matrices as introduced above to test our implementation of the PreHeat algorithm. From the room occupancy matrix for each home, we build binary bit vectors for each day for each room as specified in [7]. In their paper, the designers of PreHeat mention two improvements to their algorithm. The first is to include whether a given vector represents a weekday or weekend. The second improvement is to pad each bit vector with four hours of data from the previous and following day. It should be noted that both of these improvements were included in our implementation of the PreHeat algorithm.

5.5 Heating Simulation

To evaluate both algorithms, we initially began by using ten-fold cross validation. However, after some consideration, we abandoned this approach, because in a real deployment, days come in consecutive order, and training on days out of this natural order does not make sense. In fact, many cross-validation tools use what is known as stratified cross validation in which training instances are randomized to produce more accurate results. However, with time-series data, this approach can yield false improvements.

The best approach to evaluating both algorithms is therefore a real deployment. Because of time constraints and costs, we were unable to test an actual deployment of these algorithms. Instead, we ran them through a heating simulation and leave the deployment to future work.

The simulation gives day one to both algorithms for training, and tests on day two. Then it gives days one and two to both algorithms for training and tests on day three. It continues in this manner for every day in each study. In this way, both algorithms produce predictions for heating events for all days in the study (except the first day).

These predictions are used to set the predictive heating schedules for both algorithms as previously defined. If the algorithm makes a prediction that a heating event will occur at a given time frame, the system is set to turn on such that the zone will be properly conditioned by that point.

A reactive component is then added to both algorithms. It is defined such that if any occupancy is observed at any point and the system is not already on due to a predicted heating event, the system will be turned on. Then, the reactive system is set to turn off after n minutes of vacancy have been observed where n is a parameter which we vary and explore the effects of doing so in Section 6.3.3. Thus, the conditioning

system will be set to turn on whenever either the algorithm predicts a heating event, or the reactive system detects one. By this definition, predicted heating events take precedence over reactive heating events. The reactive component cannot cause the system to activate if it has already been turned on by the predictive component, and it will not turn off the system if it has been activated as a result of a predicted heating event.

Finally, we take the heating schedule produced by each algorithm for each data set and compare it with the observed occupancy for that data set. We then evaluate each in terms of miss time (the total number of minutes the schedule says not to heat, but for which occupancy is observed) and waste time (the total number of minutes the schedule says to heat, but for which occupancy is not observed).

5.5.1 Formal Definition of Heating Control

Once a predicted heating schedule has been realized, it is time to incorporate the reactive heating component in the heating simulation. Note that the following is an outline of the algorithms used to simulate a reactive component similar to one that could accompany an installation of Drapes.

The reactive component for both the Drapes and the PreHeat algorithms was simulated via the `reactiveHeatingEvents` matrix, an `m x 1` vector where `m` is equal to the number of minutes in the study and each cell represents whether or not the reactive component calls for a heating event for the corresponding minute. The reactive component turns on as soon as occupancy is observed and the system is currently off. It turns off after a period of vacancy has been observed for `vacancyLimit` minutes. In other words, `vacancyLimit` is a parameter that defines the minimum amount of time (in minutes) that the reactive heating component must detect vacancy in order to turn off the system under the assumption that the occupants are not returning.

```
if(reactiveHeatingEvents(t-1))
    if(reactiveHeatingEvents(t-shortCyclingLimit)
        reactiveHeatingEvents(t) = any(observedOccupancy(t:t-vacancyLimit))
    else
        reactiveHeatingEvents(t) = 1
    end
else
    reactiveHeatingEvents(t) = observedOccupancy(t)
end
```

Once both the predictive and reactive components have been determined, we can view a full schedule via the `drapesHeatingSchedule` vector where

```
drapesHeatingSchedule(t) = predictedHeatingEvents(t) or reactiveHeatingEvents(t)
```

Finally, we must include variables that are specific to each house and conditioning system via the `drapesHeatedActual` vector. It is defined as

```
drapesHeatedActual(t) = drapesHeatingSchedule(t-heatingTime) or ...
        drapesHeatingSchedule(t-coolingTime)
```

where `shortCyclingLimit` is equal to the minimum number of minutes the heating system must be left on in order to prevent damage to the system, `heatingTime` is the amount of time it takes to heat the house from an uncomfortable temperature to a comfortable one, and `coolingTime` is the amount of time it takes for the house to lose heat from a comfortable temperature to an uncomfortable one. These three parameters, `shortCyclingLimit, heatingTime`, and `coolingTime` are house and conditioning system specific.

We evaluate the system in terms of `miss time` and `waste time`. The former is a vector that represents whether or not the participants were present when the home was at an uncomfortable temperature.

```
missTime(t) = ~drapesHeatedActual(t) and observedOccupancy(t)
```

And `waste time` is a vector that represents whether or not the participants were not present while the home was set at a comfortable temperature.

```
wasteTime(t) = drapesHeatedActual(t) and ~observedOccupancy(t)
```

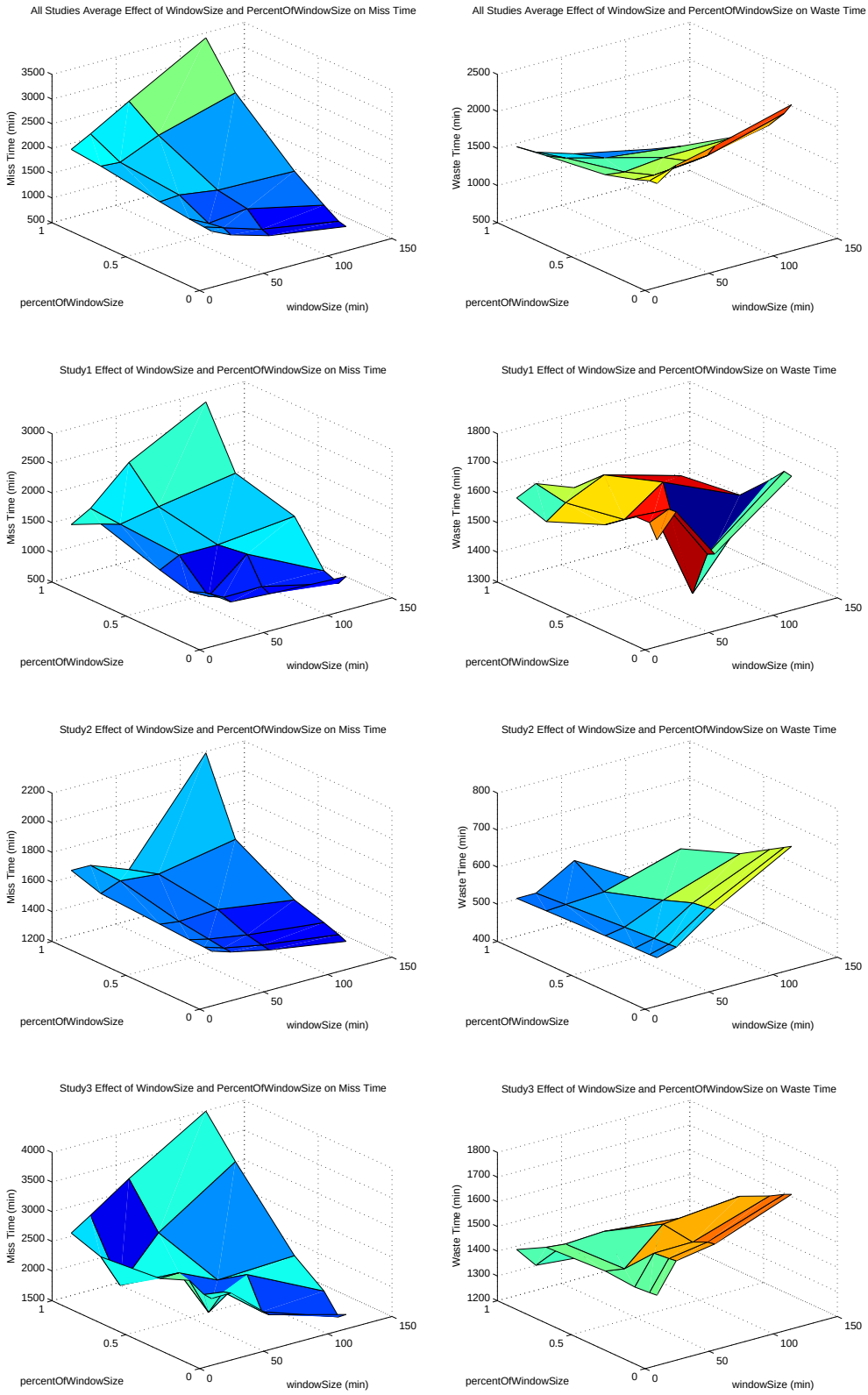

All Studies Average Effect of WindowSize and PercentOfWindowSize on Miss Time
Miss Time (min)
3500
3000
2500
2000
1500
1000
500
1
0.5
0
percentOfWindowSize
0
50
100
150
windowSize (min)
All Studies Average Effect of WindowSize and PercentOfWindowSize on Waste Time
Waste Time (min)
2500
2000
1500
1000
500
1
0.5
0
percentOfWindowSize
0
50
100
150
windowSize (min)
Study1 Effect of WindowSize and PercentOfWindowSize on Miss Time
Miss Time (min)
3000
2500
2000
1500
1000
500
1
0.5
0
percentOfWindowSize
0
50
100
150
windowSize (min)
Study1 Effect of WindowSize and PercentOfWindowSize on Waste Time
Waste Time (min)
1800
1700
1600
1500
1400
1300
1
0.5
0
percentOfWindowSize
0
50
100
150
windowSize (min)
Study2 Effect of WindowSize and PercentOfWindowSize on Miss Time
Miss Time (min)
2200
2000
1800
1600
1400
1200
1
0.5
0
percentOfWindowSize
0
50
100
150
windowSize (min)
Study2 Effect of WindowSize and PercentOfWindowSize on Waste Time
Waste Time (min)
800
700
600
500
400
1
0.5
0
percentOfWindowSize
0
50
100
150
windowSize (min)
Study3 Effect of WindowSize and PercentOfWindowSize on Miss Time
Miss Time (min)
4000
3500
3000
2500
2000
1500
1
0.5
0
percentOfWindowSize
0
50
100
150
windowSize (min)
Study3 Effect of WindowSize and PercentOfWindowSize on Waste Time
Waste Time (min)
1800
1700
1600
1500
1400
1300
1200
1
0.5
0
percentOfWindowSize
0
50
100
150
windowSize (min)

Figure 5.2

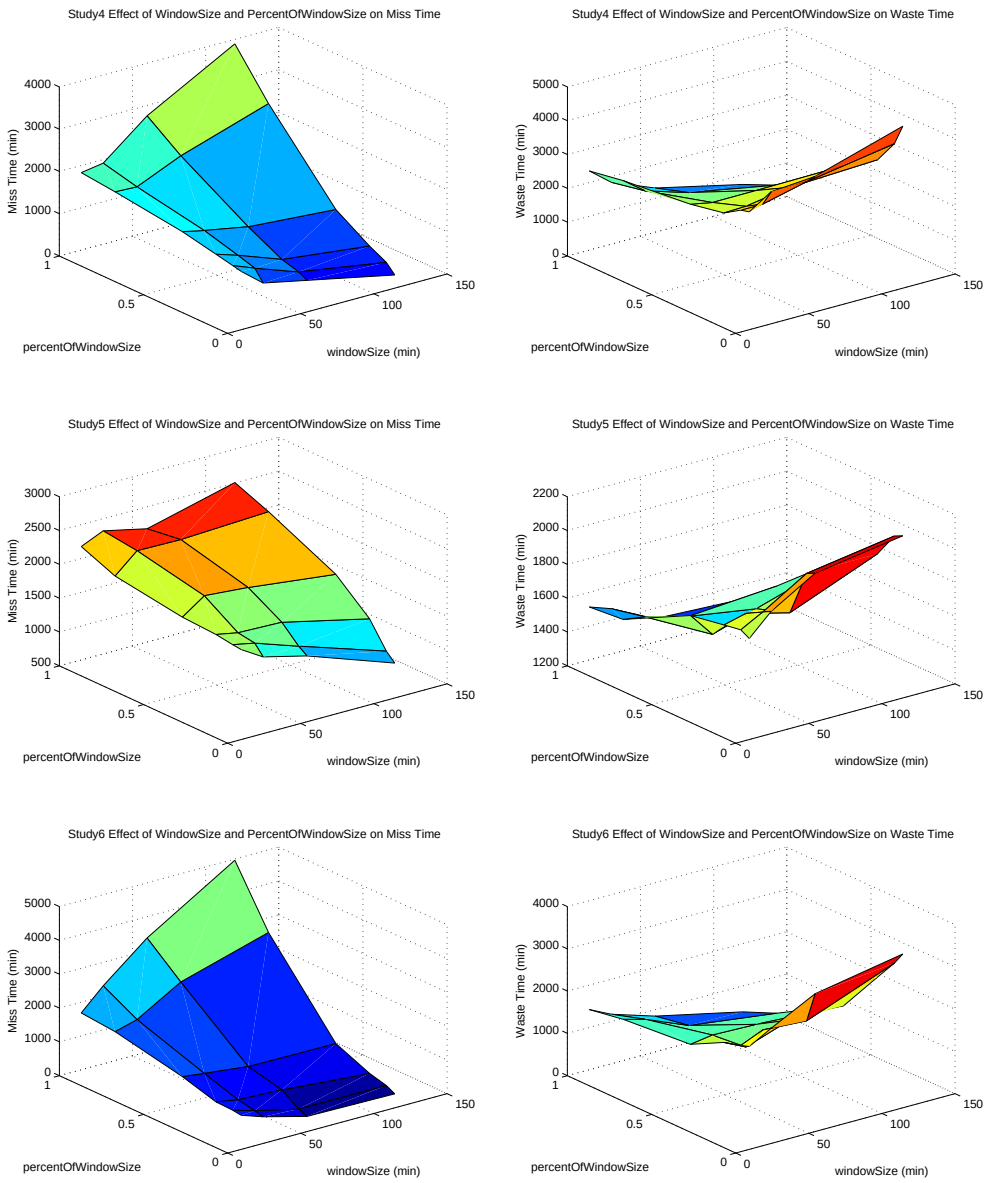


Figure 5.3: There is a clear tradeoff between Miss Time and Waste Time. As one increases, the other decreases. These parameters can be tuned to fit user preferences. For our experiments, we chose a windowSize of 30 minutes with a percentOfWindowSize of .4.

Chapter 6

Results

After running each data set through the heating simulation, and evaluating each in terms of miss time and waste time, we produced the results in Figures 6.1, 6.2, 6.3, and 6.4 where each line represents one of the seven studies, and shows the changes in miss time and waste time when using PreHeat vs. Drapes.

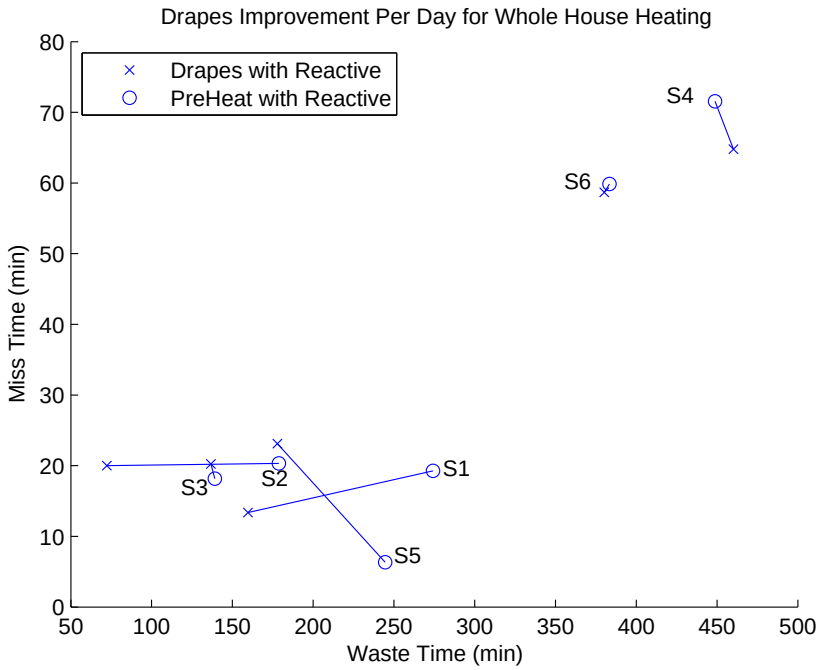


Figure 6.1: A few homes show significant improvement when using Drapes instead of PreHeat. However, the reactive component has a large impact on both algorithms and reduces visible benefits.

Figure 6.1 displays the improvements of the Drapes algorithm over PreHeat when a reactive component is incorporated. The reader will note that the performance benefits are not spectacular. However, in several of the studies we do see a significant reduction in waste time with little to no impact on miss time. This is due

to the fact that the reactive component which is the same for both algorithms is an equalizer. Both systems are limited in their ability to accurately predict future heating events, and the reactive component aids both in actively making up for missed predictions.

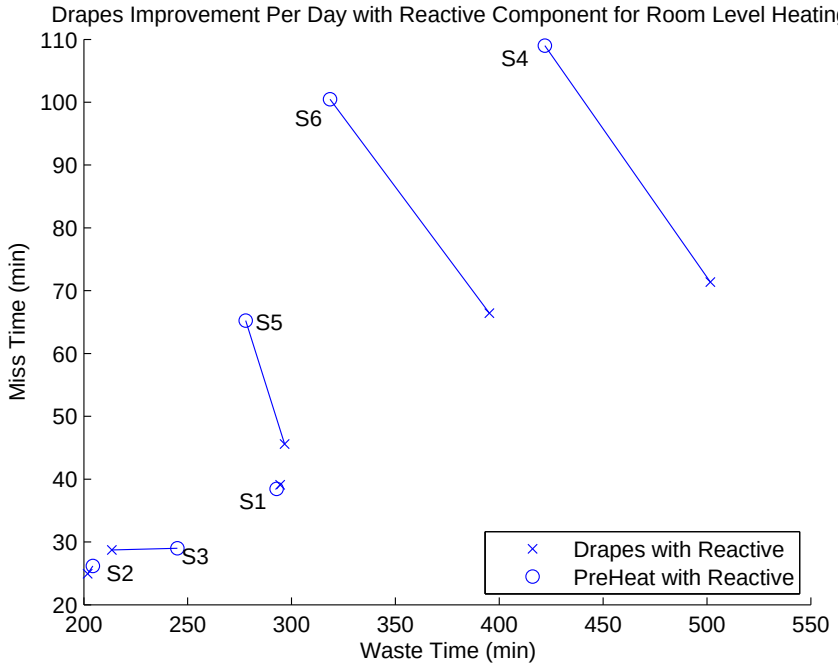


Figure 6.2: Both algorithms are capable of conditioning on a room level, but doing so is more difficult.

Figure 6.2 shows Drapes' improvements over PreHeat when predicting room level events and a reactive component is utilized. These improvements are less notable than when predicting at the whole house level. This is because, as noted earlier, individual rooms are very difficult to predict. Particularly, those with short durations of occupancy, and those which are used infrequently such as hallways and bathrooms. It is known that PreHeat performs poorly in such situations; the solution proposed by the authors of PreHeat is to condition such rooms based on whole house occupancy. This will effectively reduce miss time associated with those rooms (assuming the whole house prediction component is accurate), and increase waste time. In this exploration, we allowed both Drapes and PreHeat to condition those rooms independently of the whole house prediction. This allows us to directly compare Drapes per room conditioning capabilities to those of PreHeat. In this case, we see a fairly significant reduction in miss time for several studies, and don't see quite the reduction in waste time as compared to whole house predictions.

To really see how well the Drapes prediction algorithm improves over PreHeat's prediction algorithm, we must remove the equalizing reactive heating component. In Figure 6.3, it is apparent that Drapes yields significant reductions in both miss time and waste time for several houses. In particular, we can see more

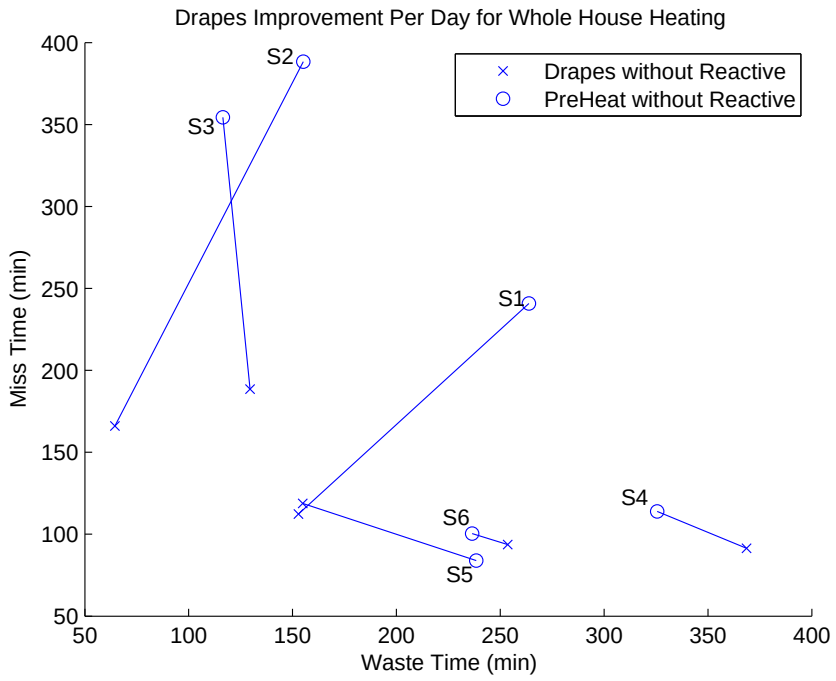


Figure 6.3: Drapes prediction reduces both miss time and waste time for many houses over PreHeat prediction, and has a significant reduction in one with minimal cost to the other for other houses.

significant improvements in data sets one, two, and three, while in others, it displays a tradeoff as it produces a reduction in terms of one metric, while sacrificing a bit of the other. The differences between these data sets is the method by which occupancy was observed. For data sets one, two, and three, the RFDoormat system was used resulting in observed occupancy with higher validity. Motion sensors fall subject to periods of inactivity, when, for example, occupants may sit still long enough for motion sensors to stop detecting them. The RFDoormat system does not have this limitation and can therefore provide us with higher fidelity data. With all data sets considered, Drapes' predictive heating component reduces miss time by an average of approximately 33% over PreHeat's, and reduces waste time by an average of approximately 15% in whole house level conditioning.

Figure 6.3 shows the improvement when predicting whole house level heating events, but Drapes is also capable of predicting room or zone level events. Figure 6.4 shows that Drapes is also capable of predicting room level heating events, although this is more difficult to do as patterns tend to be more complex at finer granularities. Still, we can see that Drapes generally makes improvements in terms of either miss time or waste time, and in some cases both. And again, we can see a tradeoff between sacrificing one for the other.

The reader may note that Studies 1-3 in Figure 6.3 show the most significant improvements in Drapes over PreHeat. Then, in Figure 6.6, Studies 1-3 show the smallest improvement over a reactive approach. This seems counter-intuitive as we may make the assumption that if Drapes showed the potential to have

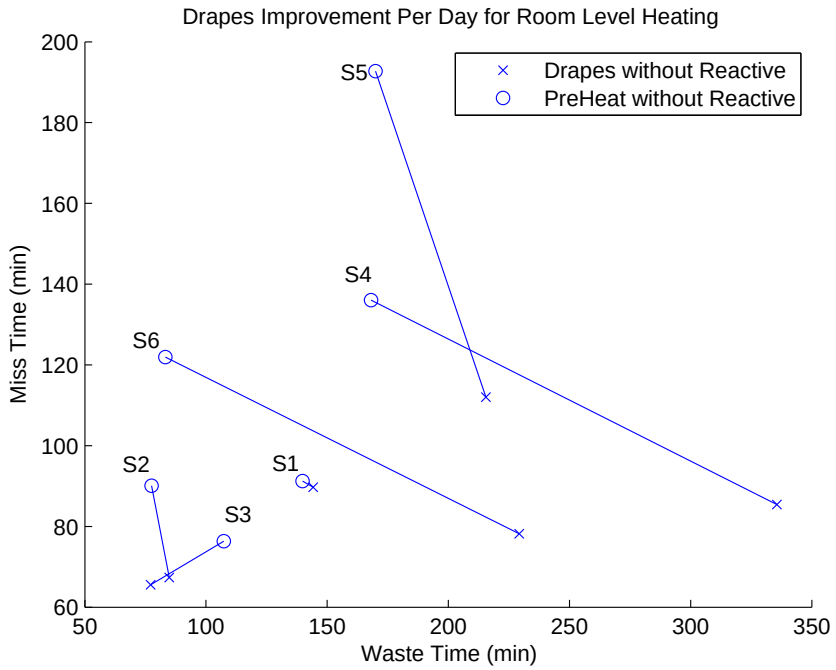


Figure 6.4: Drapes prediction does not yield quite the performance boost for room level heating as for whole house level, but some improvements are still notable.

highest prediction accuracy over PreHEat in Studies 1-3, it should also show the most improvement over a reactive strategy in the same studies. However, given further thought, we can reason that this is due to a more reliable occupancy sensing system for Studies 1-3. As noted earlier, motion sensor data is subject to false negatives if the occupants remain motionless for an extended period of time. Doing so would result in more, non-contiguous, observed occupancy events. That being the case, the reactive component would turn on and off more frequently resulting in increased miss time and waste time. This is opposed to the RF Doormat system which observes continuous occupancy events as long as the occupants remain in the same room, regardless of their level of activity. This allows a reactive component to more accurately condition a room without turning on and off multiple times during the same occupancy event.

At this point, the reader may have reasoned that these comparisons are unfair. After all, PreHeat uses a decision threshold used to fine tune the tradeoff between miss time and waste time, and how can we be sure that we have not simply created an approach that works better than PreHeat's default parameters?

To address these concerns, consider Figure 6.5. These graphs display the tradeoff between miss time and waste time as PreHeat's decision threshold is varied between 0 and 1. As the decision threshold is lowered, the PreHeat algorithm becomes more aggressive, heating more frequently, resulting in higher waste time and lower miss time. As it is increased, the PreHeat algorithim becomes less aggressive, heating more infrequently, resulting in higher miss time and lower waste time. Included on these graphs is Drapes' performance for a

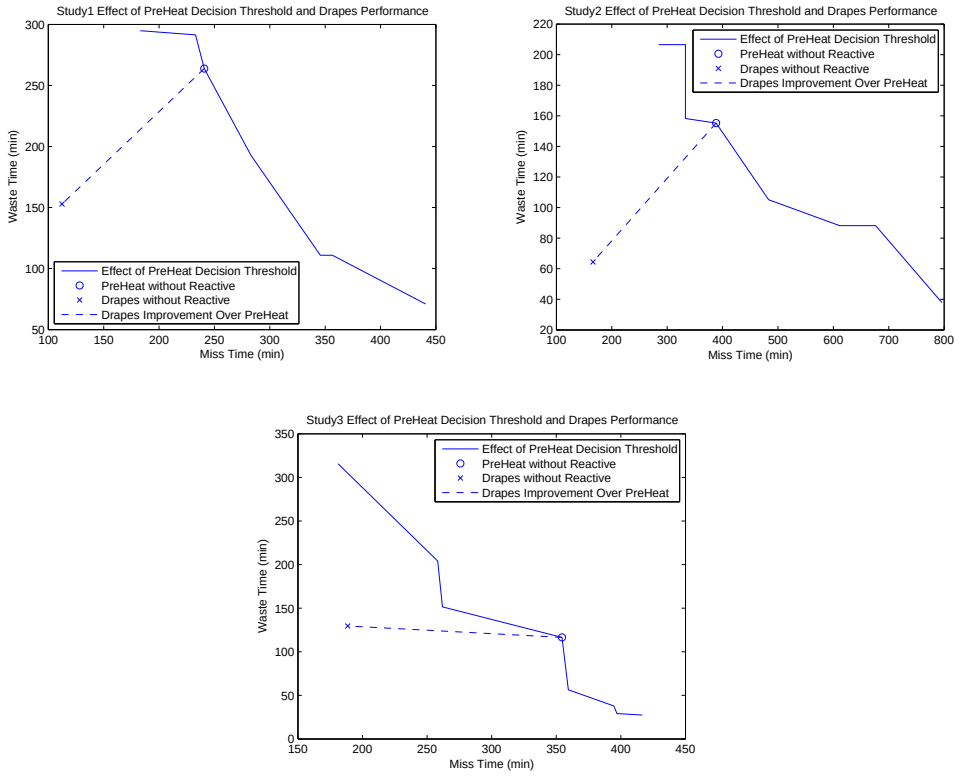


Figure 6.5: As PreHeat's Decision Threshold varies, we see the tradeoff between Miss Time and Waste Time. No matter the threshold, Drapes shows improvement.

single setting of its comparable parameters: `windowSize` and `percentOfWindowSize`. The performance of Drapes lies well beneath the PreHeat curve at all points, regardless of the decision threshold. What we see is a point that lies on a completely separate ROC curve. By using the Drapes algorithm, we are not simply choosing a different point on the same ROC curve as PreHeat, we are providing a better tradeoff altogether. This supplies confidence to the fact that we are not simply reducing miss time and waste time over PreHeat's default decision threshold of .5, but rather we are improving over the PreHeat approach as a whole.

6.1 Power of Prediction

In order to see how important the predictive component is to the Drapes heating algorithm, we compared it to a purely reactive heating schedule with no predictive component incorporated. The results of this experiment can be seen in Figure 6.6. As evidenced by the figure, adding the predictive component of Drapes to a purely reactive heating algorithm reduces miss time while using a bit more energy to do so. Naturally, the improvement seen in terms of miss time will be a function of the amount of time it takes to heat a zone from a setback temperature to a setpoint temperature. This tradeoff is explored in Section 6.3.3.

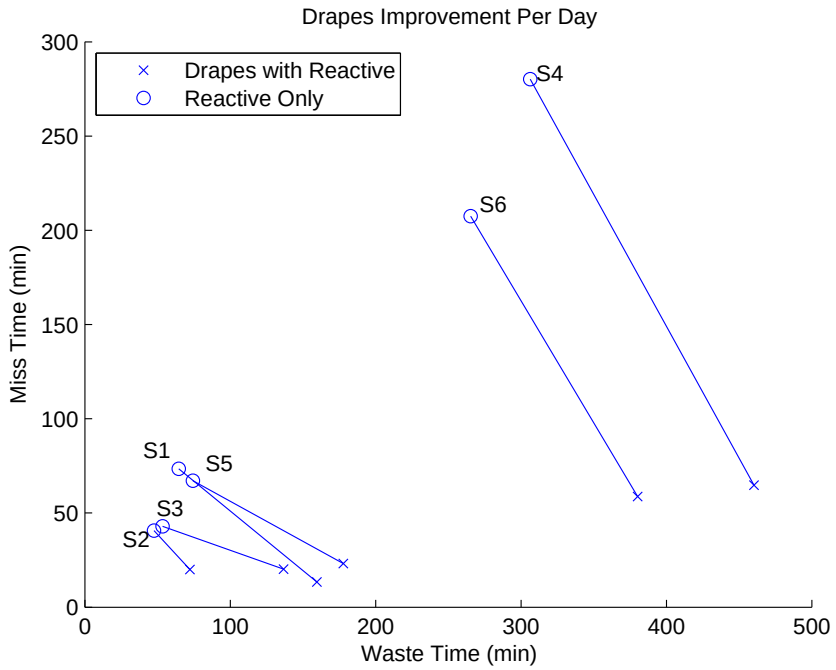


Figure 6.6: Adding the predictive component of Drapes to a purely reactive heating algorithm reduces miss time in all cases while sacrificing waste time.

6.2 Predictive vs. Reactive Heat and Energy Savings

Each figure in Figure 6.11 represents the importance of predictive heating and reactive heating in both Drapes and PreHeat. In all cases, the predictive component of Drapes is responsible for a greater percentage of the heating events than is the reactive component. PreHeat, on the other hand tends to rely more heavily on its reactive component. This is an important distinction as many homes provide the capability for multi-stage heating [4]. Essentially the HVAC system in these homes can operate in one of three settings: stage 1, used to maintain the current temperature, stage 2, used to gradually reach a setpoint temperature, and stage 3, used to quickly reach a setpoint temperature. Stage 1 uses the least amount of energy followed by stages 2 and 3 which use increasing amounts of energy respectively. Thus, a heavier reliance on reactive heating can result in greater energy waste as doing so takes more energy than heating a room slowly over time. As drapes is shown to use its predictive component for approximately 15% more of its heating events on average, Drapes will save its users more energy over time.

These energy savings can be seen in Figure 6.12. Let us assume that it costs 30,000 BTU per hour to maintain a setpoint temperature, and it costs 50,000 BTU to heat a room from a setback to a setpoint temperature in one hour. Assuming each home in the data set is equipped with an HVAC system capable of multi-stage heating as described above, and the efficiency factors of stage 1, stage 2, and stage 3 heating are

1.46, 1.65, and 1 respectively (these values are borrowed from analysis in [4]), Drapes shows reduced energy usage per day in all studies with an average per day savings of approximately 6 kWh. This translates to a $.20 per day savings for those using natural gas, and a $.76 per day savings for those using electric heat.

6.3 Analysis

There are many variables of interest given any occupancy prediction technique. We focus on several of those variables here.

6.3.1 Prediction Horizon Analysis

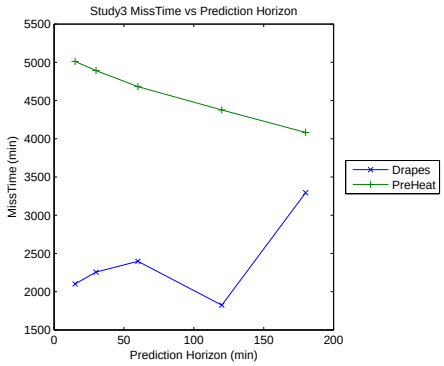
Figures 6.7a and 6.7b show the effects of varying the prediction horizon. PreHeat trends when varying the prediction horizon showed that PreHeat tends to be a bit more aggressive as the variable is increased. The figures show a decrease in miss time and an increase in waste time indicating that PreHeat is heating more frequently. The trends for Drapes, however, are not as easy to interpret. Typically showing a fairly steady increase in waste time, the effect of prediction horizon on miss time is not as clear. Seemingly sporadic jumps and dips make it difficult to draw conclusions. This can be attributed to the use of decision trees. Such a highly variable machine learning technique will inevitably display some unusual and perhaps even confusing trends when evaluated. It appears that these trends may be, in part, specific to the machine learning algorithm used. Other works have shown a clear decrease in accuracy as the prediction horizon is increased [7]. Further work is necessary to fully understand the effect that the prediction horizon has on Drapes.

Nevertheless, it takes different amounts of time to condition different zones and different homes. This is a function of the type of home, the environment, the weather, how the home is built, etc. It is important, then, that Drapes be able to predict at varying prediction horizons. For our simulations, we assume a home in which predicting 60 minutes in advance of a heating event is sufficient.

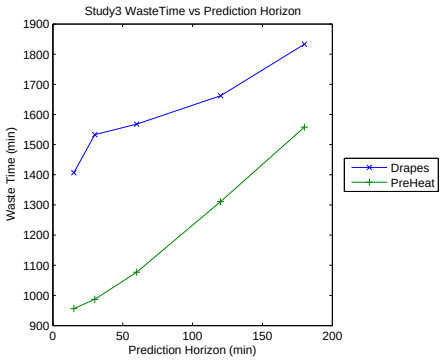
6.3.2 Feature Analysis

Perhaps the most interesting of the variables discussed in this section is that of feature selection. Figure 6.8 shows the effects on miss time and waste time for Study One as we look at each feature individually, then as we begin to combine them.

The first seven feature sets shown in Figure 6.8 are individual features as previously described, some of which have lower miss times and waste times than others indicating that they are better predictors of zone



(a) Drapes shows lower miss time and waste time regardless of the Prediction Horizon



(b) Drapes shows lower miss time and waste time regardless of the Prediction Horizon

occupancy. In this case, we can see that there are two sets of features that appear to be excellent predictors of whole house occupancy: feature 3: Room Occupancy, and feature 6: whole House Occupancy. This indicates that our hypothesis is correct, and that there are relationships inherent among rooms that are indicative of future occupancy. However, beginning with feature set eight, as we start to combine features, an interesting trend emerges. We begin to see that miss time and waste time gradually increase as features are added. This trend is most likely due to the amount of training data being insufficient to cover the growing feature space. This trend can be seen in Figure 6.9.

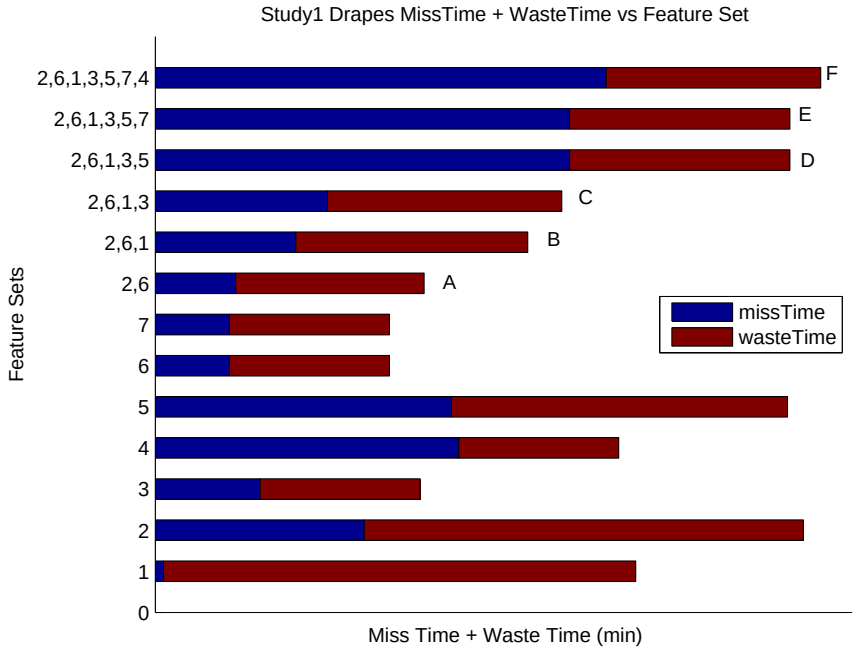


Figure 6.8: Some features have much lower miss times and waste times than others, indicating they are better predictors of heating events. Feature Sets 8-13 begin to combine individual features found in Feature Sets 1-7.

Table 6.1: Figure 6.8 Feature Set Lookup Table

#	Feature	Size of Feature Space
1	Day of Week	7
2	Time of Day	1440
3	Room Occupancy	$2^{numRooms}$
4	Room Occupancy By Occupant	$2^{(numRooms*numOccupants)}$
5	Duration of Room Occupancy	$5000^{numRooms}$
6	Whole House Occupancy	2
7	Whole House Occupancy By Occupant	$2^{numOccupants}$

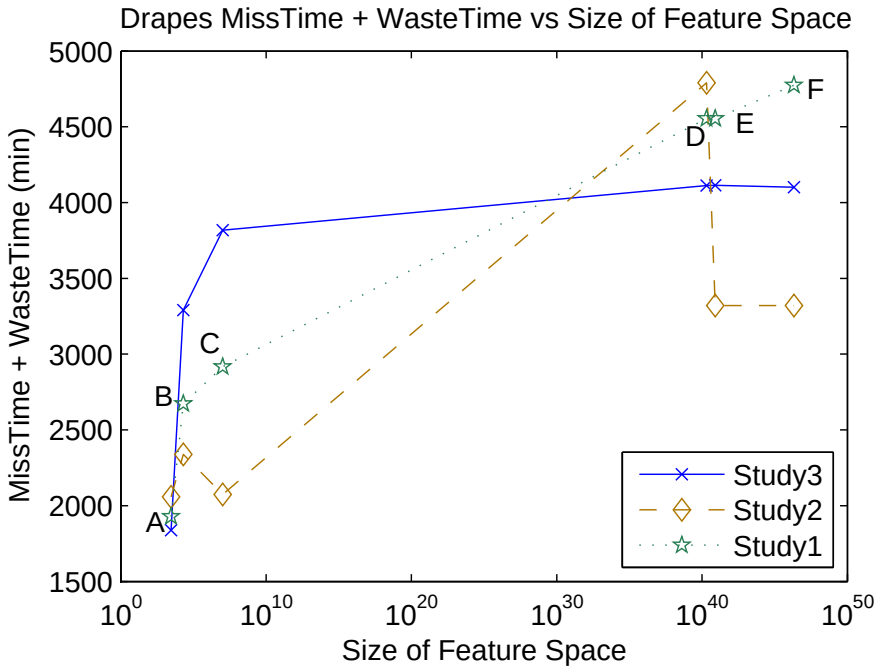


Figure 6.9: As the size of the feature space increases, insufficient training data causes increased miss time and waste time.

6.3.3 Simulation Parameters

All houses are very different. They gain and lose heat at different rates, they have different conditioning systems which must be controlled in various ways to optimize their energy consumption, etc. To address some of these concerns, we incorporate several parameters into our heating simulation to help understand how some of these varying scenarios will respond to a Drapes installation. These parameters are analyzed in Figure 6.10.

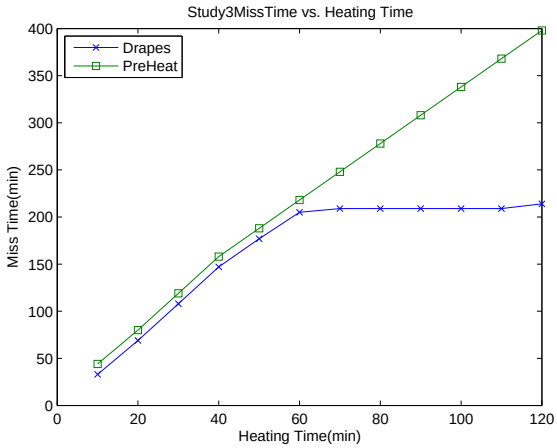
The first parameter we explore is `heatingTime` in Figure 6.10a. As the amount of time it takes for a conditioning system to get a room to a comfortable temperature increases, so does the miss time. This is an intuitive result, as the penalty is incurred mainly by the reactive component. When a room is reactively heated, every minute it takes to reach the comfortable temperature is likely to result in occupant discomfort

as they wait for the heat to 'kick in'.

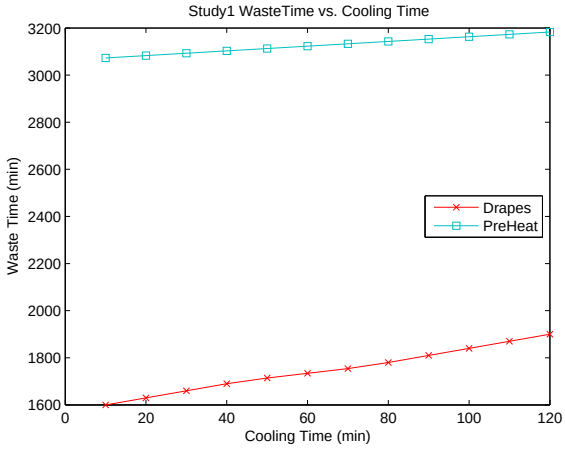
The second parameter, `coolingTime` is similar to the first. As the amount of time it takes for a room to lose its heat increases, so does the waste time as evident in Figure 6.10b. If the cooling time of a home were known in advance, this parameter could be used to reduce waste time. Knowing the `coolingTime` in advance would allow us to turn off the system at the most opportune time to result in minimal wasted energy at the end of a heating event. In essence, the system could be tuned such that just as occupants are predicted to be leaving a zone, the temperature would just be reaching a more energy effecient setback temperature at that moment.

Finally, we discuss the `vacancyLimit` parameter: the minimum amount of time the reactive component must observe vacancy before it turns off. Intuitively, there is a direct correlation between this value and the amount of waste time at the tail end of a heating event. The longer the system must wait before making the assumption that the occupants will not return, the longer it must maintain a comfortable temperature believing the occupants may return. This is exactly the trend we see in Figure 6.10c.

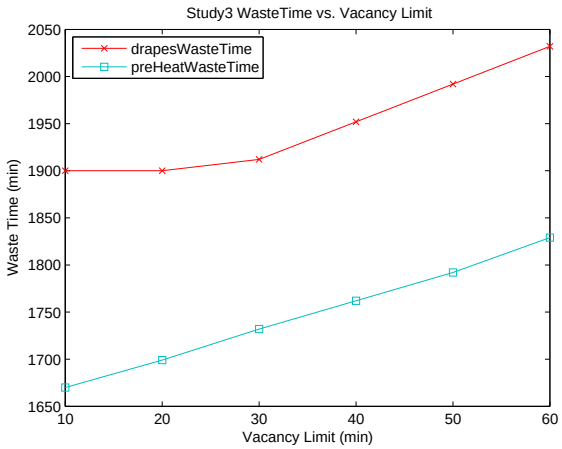
Individually, none of these simulation parameters are inherently interesting. However, it is important to understand the effect they have on the heating algorithms in order to have a more comprehensive understanding of how these algorithms may perform in various homes and environments.



(a) The amount of time it takes to heat a room at the start of a reactive heating event directly influences the amount of miss time.



(b) The amount of time it takes to cool a room at the end of a reactive heating event directly influences the amount of waste time.



(c) The amount of time that no occupancy must be observed before the reactive component turns off the heating system directly influences the amount of waste time.

Figure 6.10

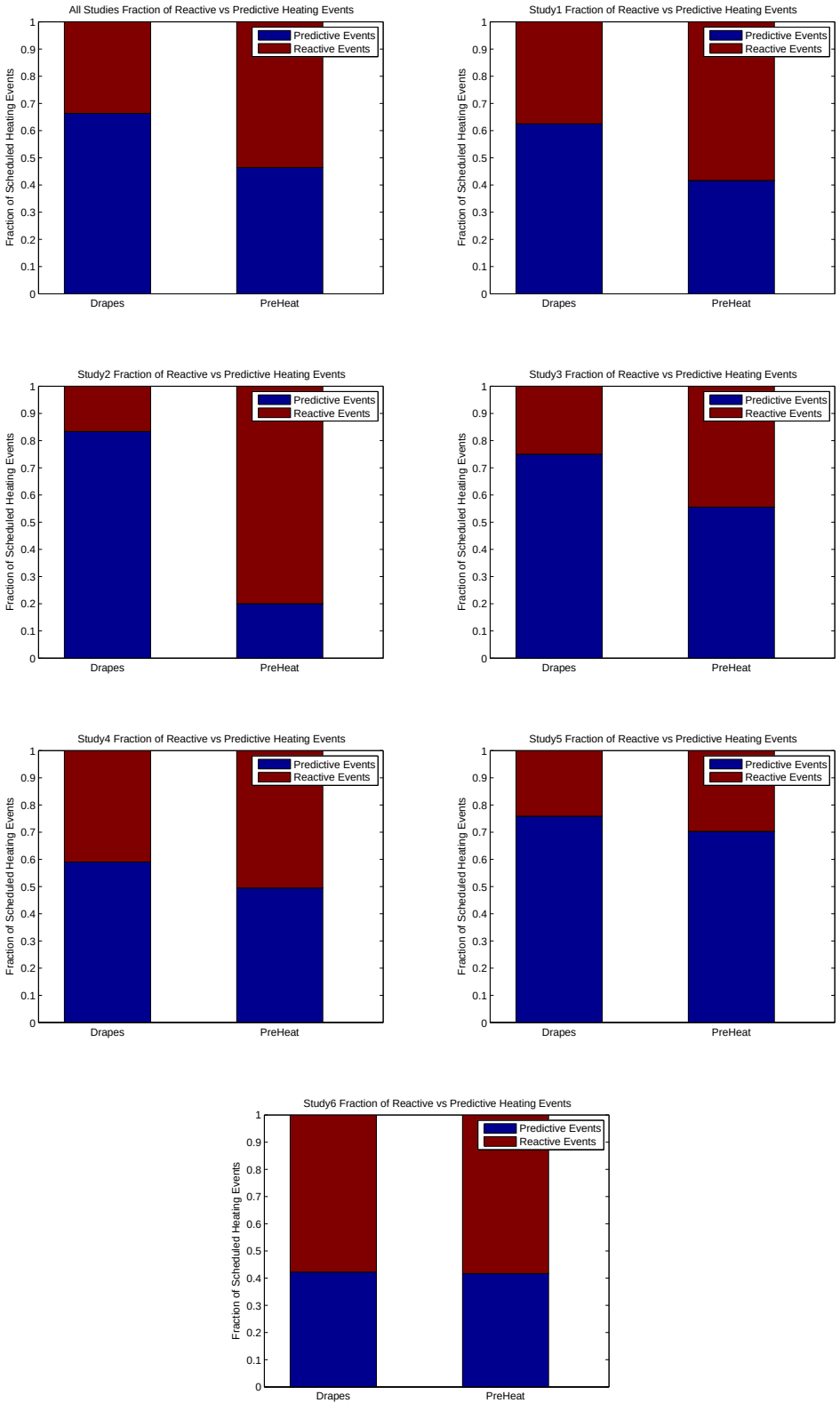


Figure 6.11: Drapes utilizes its prediction component more frequently than PreHeat resulting in more efficient energy usage.

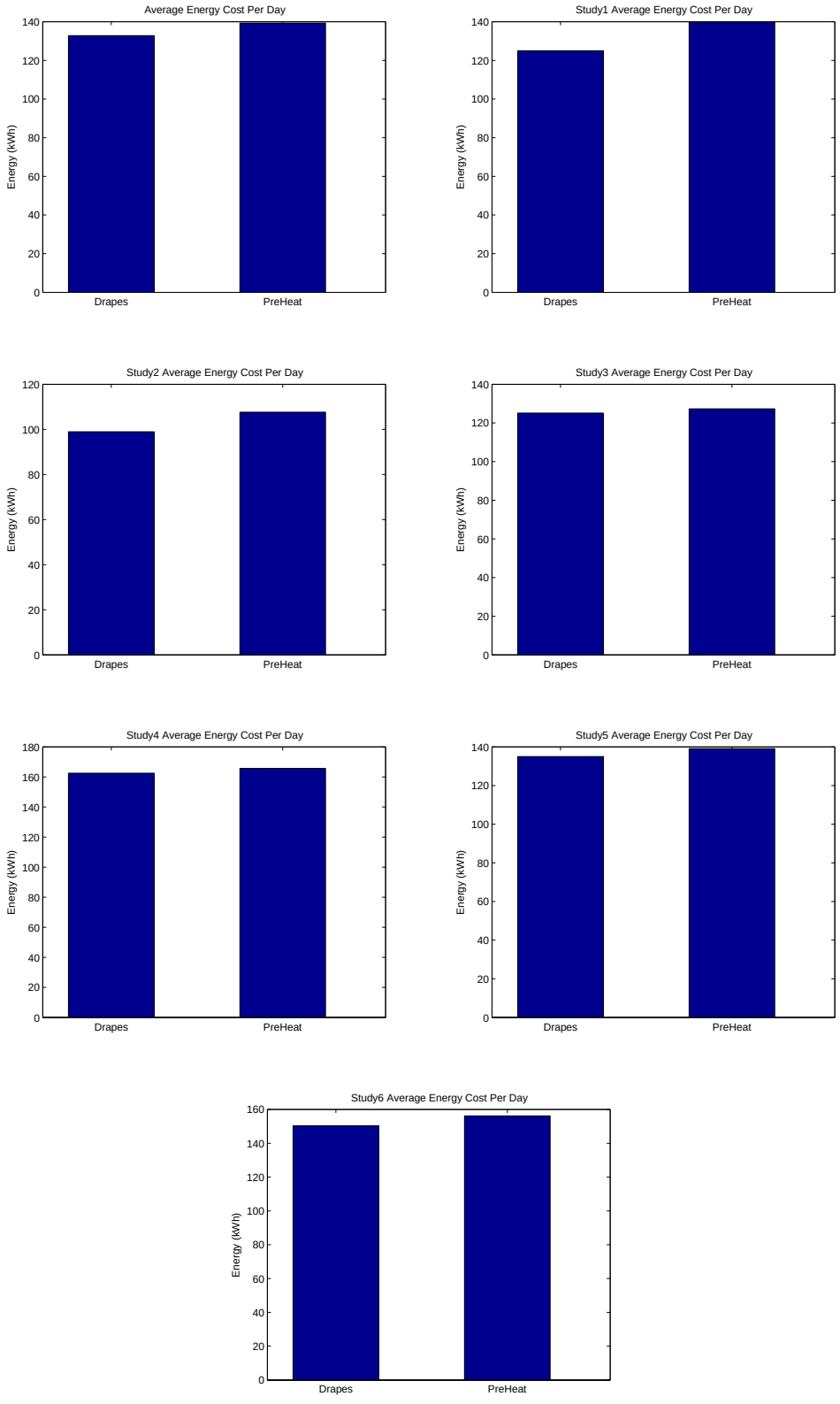


Figure 6.12: In homes with multi-stage heating, Drapes shows reduced energy usage resulting in less energy waste and smaller energy bills.

Chapter 7

Conclusion

In this paper, we have attempted to address the excessive energy use with regards to residential heating and cooling systems by introducing Drapes, predictive heating system that takes advantage of the holistic nature of homes. Drapes can be used for various smart home applications that may require knowledge of future occupancy, but it is most readily applicable to a smart heating system in which predictions of future occupancy are used to appropriately heat or cool zones in a home. The novelty of our approach lies in the realization that rooms in homes are not independent, but rather they are inherently related. Therefore, any valid occupancy prediction system must take a holistic approach when modeling occupancy patterns within a home.

Using homes from the Charlottesville, VA area and four additional data sets from Washing State Universities CASAS Smart Home Project and Microsoft Research, we compared Drapes to the state of the art system, PreHeat, and showed that Drapes is able to waste less energy and provide higher levels of comfort to occupants. Finally, we provided useful insight into which features of a home sufficiently model occupancy patterns by analyzing the improvements evident across a series of occupancy models.

7.1 Limitations and Future Work

There are several important limitations to this work. First, because of its choice of machine learning algorithm, PreHeat has an innate abilitly to modify the probability decisions made by the algorithm. In other words, it is more easily adjustible to suit occupant needs and wants. If, for example, the user would prefer to save energy and spend less money, then PreHeat can increase its probability threshold, requiring a greater number of the 5 similar days to agree upon an occupancy prediction, thereby making it more likely that the system will predict unoccupied and refrain from heating a room. On the other hand, should the occupants

desire to be comfortable more often at the expense if using a bit more energy and money, they can reduce the probability threshold, making it more likely that the system will predict occupancy and heat the room more often. Drapes also has this ability, however it is somewhat more cumbersome to change. Mentioned previously, the percent of window variable used to determine the amount of occupancy that must be observed to classify a window as a heating event can be adjusted to meet this need. As that variable is lessened, the Drapes algorithm becomes more aggressive finding more events it can classify as heating events. And as the percent of window required is increased, the algorithm finds fewer events that meet the criteria, making it a less aggressive approach. However, this parameter makes fine tuning the algorithm a bit more difficult than setting a simple probability threshold.

Another drawback is that the energy savings calculations found in this paper are simplified. We acknowledge that there are many more complexities when calculating energy usage, including the fact that the rate at which energy is lost differs for each room and house. It is not simply lost, but is transferred to other parts of the house. Similarly, heating systems are much more complicated than addressed here, and a control algorithm must be robust to work with them.

Finally, finding long term motion sensor deployment data sets is difficult. Those found online are limited, and to execute our own deployments is costly in terms of both time and money. Because of this, it is difficult to be able to generalize and draw conclusions from such a limited amount of data. However, we believe that our data is various enough, that the conclusions drawn are reasonable.

Aside from the benefits we have shown to exist for residential heating and cooling control, there exist many other potential applications for our work.

Although the solution is not yet complete, we envision a future in which home occupancy monitoring and prediction is commonplace. Applications and benefits of such systems could be limitless. One such application is monitoring of the elderly and those with other mental or physical disabilities. To illustrate, a future version of Drapes may be able to learn occupancy patterns with such accuracy, that it may observe an elderly occupant still in bed when predicted to be active in the home. Or perhaps the occupant has been in the same room for a longer period of time than is typical. In such cases, it may be wise to notify an immediate family member or caretaker to let them know that something is unusual, and it might be a good idea to check in.

Appendices

Appendix A

Percolator

In this section, we introduce Percolator, a technique that predicts room occupancy by choosing among several occupancy models which range in granularity. Percolator chooses which model to use based on the amount and type of training data collected. Hence, in early stages of deployment, when a limited amount of training data has been collected, Percolator uses a course grained model to make a prediction. Then, as training time increases, and Percolator collects more training data, it uses a finer grained model providing higher prediction accuracy. To analyze this, we collect data in four homes for 60 days each. We focus on several features of each home including time, room state, and duration of current room state among others. Although our results suggest that applying the Percolator technique to decision trees does not increase their performance, we still are able to explore some interesting results. Mainly, we show that models which use only a subset of features can predict as well as more complex, finer grained models at short prediction horizons. And for longer prediction horizons, although the simpler model does not perform as well, its accuracy is till about 70% in most cases. We show that, depending on your needs and budget, a simpler, less expensive model may prove to be a better alternative to a finer grained, more expensive model.

A.1 Approach

Percolator is composed of multiple models which range in granularity of features. Based on how much training data has been collected, Percolator chooses which model to use. To understand how this might be effective, consider a system that has been deployed for one week only. In such a scenario, there may be data collected such that we see that every time the kitchen is occupied at 5:00, the dining room is subsequently occupied 30 minutes later. As more data is collected on subsequent days, there may be data to suggest that when the kitchen is occupied at 5:00, and the living room has been occupied for some period of time, then the living

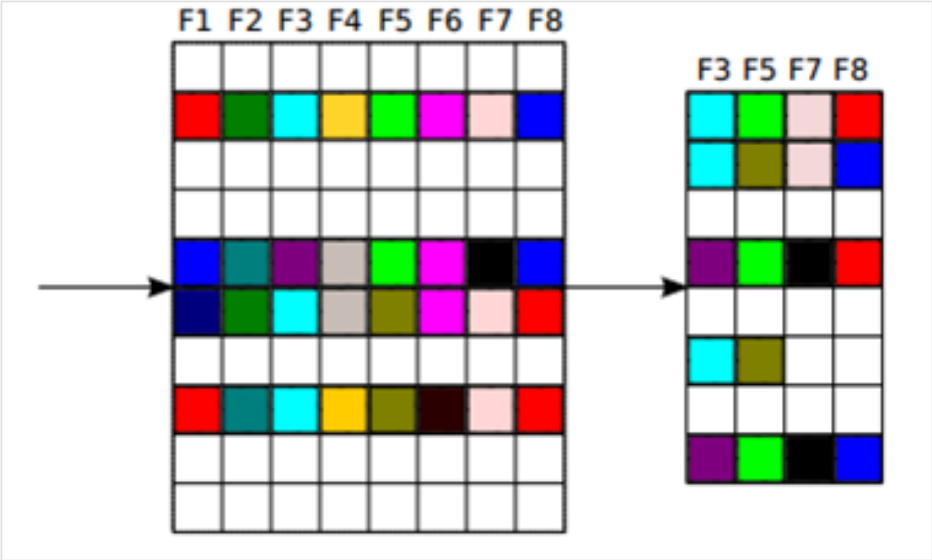


Figure A.1: An illustration of Percolator with a depth of 2. Each row represents a given state of a home, with each column being different features such as time, room state, etc. When a given state has been observed enough times, we may use the fine-grained model. Otherwise, we abstract away from some features, combining rows with similar features to form the coarse-grained model resulting in more observations per state.

room, instead of the dining room, will be occupied 30 minutes later. A course grained model, such as the one used in the first few days of deployment is not able to distinguish between the two events. Instead, a finer grained model is able to make a more accurate prediction.

To illustrate this more clearly, consider Figure A.1. It illustrates a Percolator composed of two models. Each column (F1-F8) represents a feature of the room, such as time, room state, duration of room state, neighboring room states, etc. Each row represents a state of the home, and is associated with several values: the total number of times we have seen that state, and the total number of times each room has been occupied at a given time in the future. If there has been enough data collected to make a prediction given a full set of features (if we have seen the room state enough times), then we can use the finer grained model. If however, there has not been enough data collected (if we have not seen the room state enough times), we abstract away from certain features and look at a coarser grained model. When we abstract away from given features, rows are combined based on common features yielding a greater number of observances of a given state of the home from which we can hopefully make more accurate predictions.

Each model is implemented via a decision tree. In other words, for each room, prediction horizon, and model, there is a decision tree trained on data collected. Percolator's objective is then to determine which decision tree to use for prediction. Given an observed state of the home, if observed enough times, it can make a prediction using a decision tree trained on a full feature set. If not observed enough times, it can make a prediction using a decision tree trained on some subset of the full feature set.

A.2 Experimental Setup

Data collected from four residences over periods ranging from two to six months was used to evaluate different feature combinations. Although data was collected over such a long range of time, we used only sixty days worth of data for each home in our analysis to ensure that we were making reasonable comparisons. These deployments included both single and multi-person homes with varying occupancy patterns. The residents ranged from students and working professionals to stay at home parents and residents who worked at home. Table A.1 summarizes the information about the homes.

In our studies, we focused on six main attributes: time, exterior door state change, room state, neighboring room state, non-neighboring room state, and duration of current room state. The cost of gathering such information is costly. It requires motion sensors and door sensors to be placed throughout the home in every room. Time, room state, and duration of current room state, however, can all be gathered from a single sensor. Neighboring room state, non-neighboring room state, and exterior door state change require placing different sensors in other rooms, and on exterior doors respectively. This incurs greater costs not only for the hardware, but for installation, computation, and maintenance as well.

A.3 Early Percolator Results

Before using decision trees to predict occupancy at room level, Percolator used a simple threshold technique to make occupancy predictions. In this case, probability of occupancy is given by the following equation: $P_O = \frac{C_R}{C_T}$ where C_R is the total number of times a room has been occupied at the prediction horizon of interest, and C_T is the total number of times we have seen the given room state. If this probability lies above some set threshold, then Percolator predicts that the room will be occupied at the given prediction horizon. Results of using this technique can be seen in Figure A.2. As evidenced by the figure, no model predicts with accuracy above 70% at any given prediction horizon. However, when models are combined via the Percolator technique, there is a dramatic increase in accuracy. This is due to the fact that Percolator, by its very nature, selects the model that has recorded the most observations of a given room state, and thus has a higher probability of predicting occupancy correctly.

Given these results, we hypothesize that if we can improve the prediction accuracy of the given models, then the prediction accuracy of Percolator would also show improvement. To do this, we move away from simple threshold techniques and use a different machine learning algorithm.

Figure A.4 illustrates our exploration of different machine learning techniques. Figure A.4a and figure A.4b show the energy and comfort penalties respectively that are associated with each explored machine

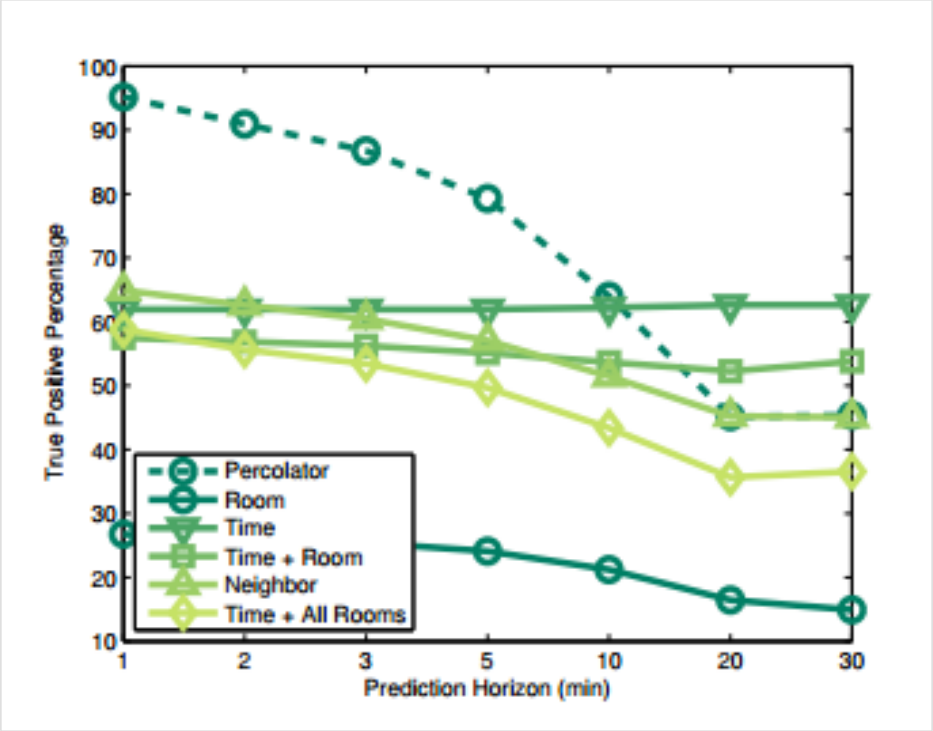


Figure A.2: Preliminary results using the Percolator technique. No individual model predicts with accuracy above 70%, but Percolator, which uses a combinataion of all models, shows a dramatic increase in prediction accuracy.

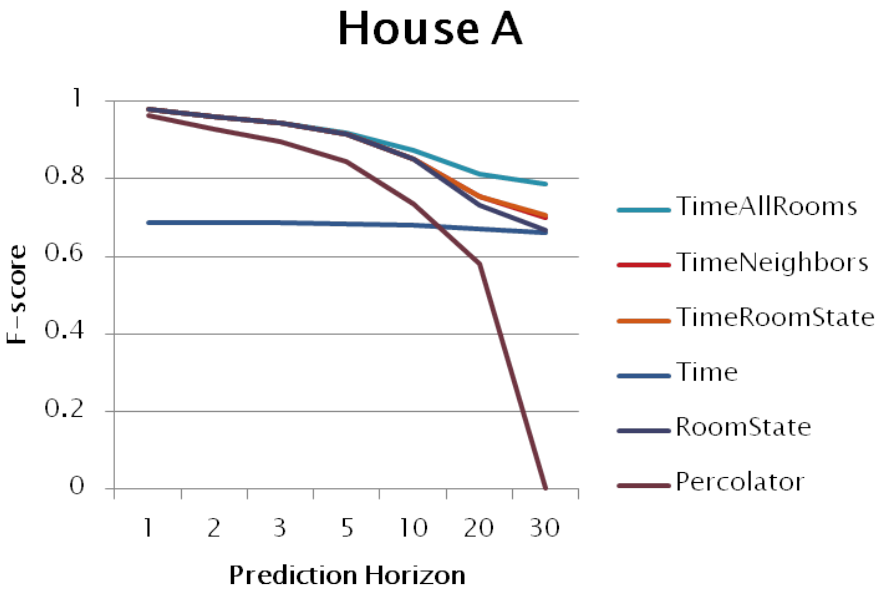
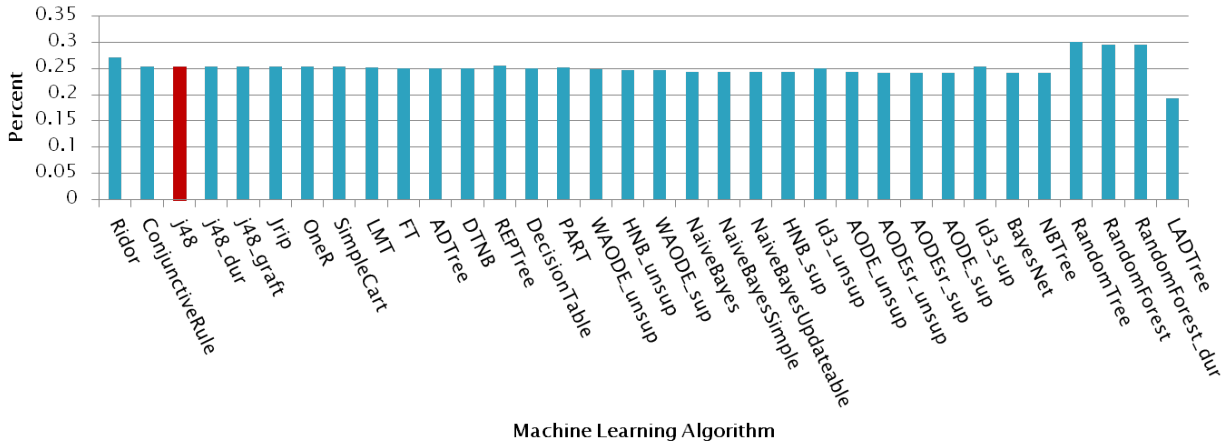
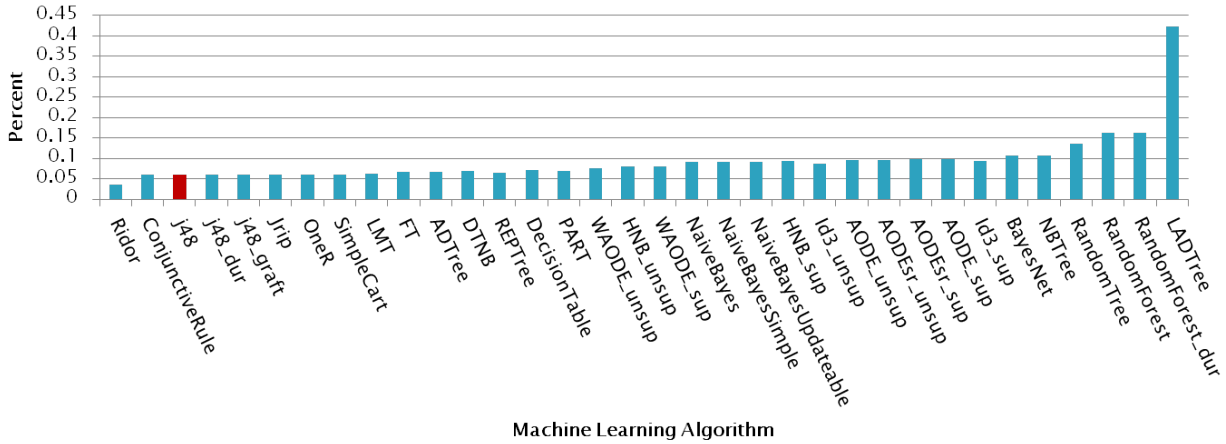


Figure A.3: When individual decision tree models were combined via the Percolator technique, the result was a decrease in prediction accuracy



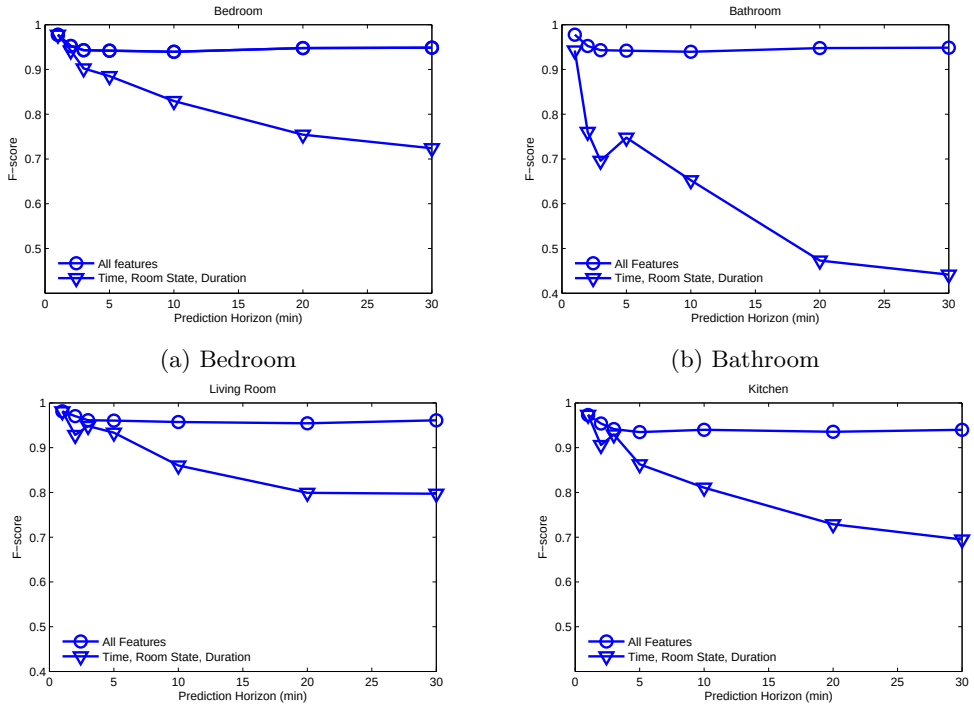
(a) Energy Penalty



(b) Comfort Penalty

Figure A.4: The j48 (C4.5) algorithm pays lower energy and comfort penalties on average than other machine learning algorithms

learning technique. To briefly explain the difference, an energy penalty is paid any time our prediction system incorrectly predicts that a room will be occupied resulting in heating a room when no occupants are present. And a comfort penalty is paid any time our prediction system incorrectly predicts that a room will not be occupied resulting in unhappy occupants in an unheated room. The ideal machine learning technique will have minimal energy and comfort penalties. Preferably, the technique should be a familiar one to allow for an easier comprehension of the approach and to more readily lend itself to comparisons. Based on these attributes, we selected the j48 decision tree algorithm which is equivalent to the more well known C4.5 algorithm.



(a) Bedroom (b) Bathroom

(c) Living Room (d) Kitchen

Figure A.5: Charts (a), (c), and (d) above show that a simpler, less expensive model performs as well as a more costly model at short prediction horizons. However, even at longer horizons, the simpler model still performs above 70% in most cases.

House	#Residents	#Rooms	#Motion Sensors	#Weeks of Data
A	2	7	14	12
B	1	7	8	24
E	2	10	10	8
F	3	7	7	12

Table A.1: Summary of house details, and length of data collection

A.4 Results and Analysis

We evaluated our models for each room, house, and prediction horizon using 10 fold cross validation on the 60 days of data collected. Due to space constraints, because we found similar results across all homes, we will discuss results in term of house A alone. Figure A.3 charts the results for house A. As evidenced by the figure, we saw a dramatic increase in performance among all models compared to those shown by preliminary results. However, when Percolator attempted to use those models, its prediction accuracy was not as high. And as the prediction horizon increased, its performance became worse. Why are these results so much worse than our preliminary results? By its construction, Percolator attempts to determine how many features it should use to predict future room occupancy. It does this by selecting finer or coarser grained models based

on how much data it has collected. However, it is limited in its selection of features by the models that have been predefined for its use. Decision trees, on the other hand, already have a method of feature selection built into their construction. When given a full set of features, decision trees use information gain, or some other measure of attribute importance, to select which features yield the most information on the target value. Once the tree is constructed, it prunes away unnecessary branches, potentially removing some features from its decision structure altogether. For this reason, Percolator is not applicable to decision tree occupancy models.

Because decision trees alone outperformed our preliminary results, we began to explore some other interesting findings. We turned to explore which features are important for room prediction. This was done by building several decision trees using different subsets of features, and comparing their prediction accuracy across homes. To compare prediction accuracy, we averaged our results across all common rooms to determine for which type of room and prediction horizon a simpler model would perform as well as a more complex model. Common rooms across homes include only four types: Bedrooms, Bathrooms, Living Rooms, and Kitchens. Other rooms not common to all homes include those such as nurseries, mud-rooms, offices, etc.

As is evidenced by Figure A.5, bedrooms, living rooms, and kitchens can be accurately predicted using a smaller occupancy model for shorter prediction horizons. Even as the prediction horizon increases, the simpler model still performs relatively well. Naturally, there is a trade off here. You can use a more specific model, encompassing more features of a home and receive greater accuracy at long prediction horizons while incurring greater cost, or you can use a simpler model, which performs just as well for shorter prediction horizons with lesser cost, but does not perform quite as well at longer prediction horizons. Depending on the application that is using the occupancy prediction model, one may be more practical than the other. For example, should we desire to integrate with a home heating system, certainly a more accurate occupancy prediction system is preferable. Otherwise, we may incur greater comfort penalties (failing to heat a room when occupied) and greater energy penalties (heating a room when not occupied). On the other hand, if we desire to integrate with another system, such as one that only requires short term predictions, then we can reduce costs and attain comparable accuracy by using the simpler model. Obviously, however, the simpler model may not perform as well for all rooms. Considering, for example, the bathroom in Figure A.5, the simpler model showed a decrease in accuracy almost immediately.

On another note, it is likely that there are other features besides the six that we consider in this paper that may provide more insight and higher accuracy when predicting room level occupancy. More research is needed in this area, however, to determine which features of a home may perform better.

Bibliography

[1] Energy Information Administration. http://www.eia.gov/. Accessed: 2013-10-02.

[2] International Energy Agency. Key world energy statisitcs. http://www.iea.org/publications/freepublications/publication/KeyWorld2013.pdf, 9 rue de la federation, 75739 Paris Cedex 15, France, 2013.

[3] John Krumm and A.J.Bernheim Brush. Learning time-based presence probabilities. In Kent Lyons, Jeffrey Hightower, and ElaineM. Huang, editors, *Pervasive Computing*, Lecture Notes in Computer Science. 2011.

[4] Jiakang Lu, Tamim Sookoor, Vijay Srinivasan, Ge Gao, Brian Holben, John Stankovic, Eric Field, and Kamin Whitehouse. The smart thermostat: Using occupancy sensors to save energy in homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 211–224, New York, NY, USA, 2010. ACM.

[5] Michael C Mozer, Lucky Vidmar, Robert H Dodier, et al. The Neurothermostat: Predictive optimal control of residential heating systems. *Advances in Neural Information Processing Systems*, 1997.

[6] V.L. Erickson, M.A. Carreira-Perpinan, and A.E. Cerpa. Observe: Occupancy-based system for efficient reduction of hvac energy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 258–269, April 2011.

[7] James Scott, AJ Bernheim Brush, John Krumm, Brian Meyers, Michael Hazas, Stephen Hodges, and Nicolas Villar. Preheat: controlling home heating using occupancy prediction. In *Proceedings of UbiComp*, 2011.

[8] Andrew Frye, Michel Goraczko, Jie Liu, Anindya Prodhan, and Kamin Whitehouse. Circulo: Saving energy with just-in-time hot water recirculation. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, pages 16:1–16:8, New York, NY, USA, 2013. ACM.

[9] WSU CASAS smart home project. http://ailab.wsu.edu/casas/.

[10] Juhi Ranjan, Yu Yao, and Kamin Whitehouse. An rf doormat for tracking people's room locations. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 797–800, New York, NY, USA, 2013. ACM.

[11] Manu Gupta, StephenS. Intille, and Kent Larson. Adding gps-control to traditional thermostats: An exploration of potential energy savings and design challenges. In Hideyuki Tokuda, Michael Beigl, Adrian Friday, A.J.Bernheim Brush, and Yoshito Tobe, editors, *Pervasive Computing*, volume 5538 of *Lecture Notes in Computer Science*. 2009.