Collaborative Recommender Systems for Building Automation

Michael LeMay, Jason J. Haas, and Carl A. Gunter
University of Illinois at Urbana-Champaign
{mdlemay2@cs, jjhaas2@crhc, cgunter@cs}.uiuc.edu

Abstract

Building Automation Systems (BASs) can save building owners money by reducing energy consumption while simultaneously preserving occupant comfort. There are algorithms that optimize this tradeoff, such as detecting which appliances are turned on without requiring expensive status detectors to be attached to each appliance. However, better ways are needed to determine which algorithms are best-suited to a particular building. This paper explores the idea of allowing building managers to automatically communicate among themselves and exchange ratings of individual monitoring and control algorithms in such a way that each building manager can then obtain predicted ratings for all algorithms that he has not yet tried personally. We allow individual algorithms to be replaced by using a blackboard architecture to loosen the coupling between them. We propose a recommender system that operates on a database of contributed ratings to predict ratings of untried algorithms. To explore this approach, we developed a prototype that seamlessly interacts with both emulated physical buildings and buildings simulated in software and we implemented several of the control algorithms described in previous works. We demonstrate a recommender system that selects between algorithms in various types of buildings.

1. Introduction

Collaborative recommender systems have been used to create recommendations that help individuals maximize the utility of time and money spent consuming movies, products, and reading materials. The collaborative features of these systems help them to continuously adapt themselves to changing consumer attitudes and reason about massive sets of items by integrating ratings from other individuals with similar interests or characteristics. Another domain that will potentially require consumers to select among a large number of items is the one occupied by Building Automation Systems (BASs), which can be used to conserve energy and reduce building owners' energy bills in other ways. Current buildings commonly implement simple control schemes, such as using motion detectors to determine when rooms are occupied and automatically turning off lights in unoccupied rooms. However, more sophisticated control regimes that automatically respond to changing electricity prices and adjust the settings of air conditioners and other complex devices to minimize costs while maintaining adequate comfort levels in the building are coming into existence.

The increase in overall BAS complexity creates opportunities to develop algorithms that perform well in some types of buildings and poorly in others. For example, a control algorithm that shuts down an air conditioner when a building is unoccupied may perform well in a small home that can be cooled quickly, yet perform poorly in a large home that has more significant thermal mass, creating longer cooling times. Along similar lines, consider a more advanced Heating, Ventilation, and Air-Conditioning (HVAC) controller that predicts when a building's occupants will arrive and pre-cools the building, rather than waiting for them to actually arrive and be detected. There may be a tradeoff between the accuracy and computational complexity of occupancy prediction heuristics, causing simpler and cheaper algorithms to be preferred in buildings with few occupants, since their movements are less likely to interfere with each other and confuse a simple heuristic, while more complex algorithms would be required in buildings with many occupants.

Without a detailed understanding of these subtleties, the manager of a particular building may have difficulty selecting among control algorithms, and may not even realize that the algorithms in his BAS are not performing as well as others that are available. These complexities can also make it difficult to accurately evaluate algorithms using simulations, since it can be difficult to develop a realistic simulation of all aspects of a building controlled by a sophisticated BAS. This

. In: Hawaii International Conference on System Sciences, Waikoloa, Hawaii, January 2009

motivates our system, which does not require simulations but rather deals with actual buildings. The deficiencies in BAS management workflow that we have highlighted are particularly troubling given the increasing cost of a poorly-configured BAS that wastes electricity. From 2005 to 2007, the average cost of energy in the United States rose by approximately 12.3% [1]. For comparison, the CPI inflation in the United States was 6.17% during the same period [2].

In this paper, we explain how collaborative recommender systems can be used to select among multiple building control algorithms to optimize the energy-efficiency and occupant comfort levels of a particular building. Our primary contribution is a loosely-coupled architecture for coordinating BAS control algorithms, sharing ratings of those algorithms among a group of building managers, and replacing them when the building manager decides to do so in response to ratings received from other building managers. Our secondary contributions include a prototype implementation and evaluation of a *Social Filtering* collaborative recommender system for BAS control algorithms, several specific BAS control algorithms for testing, and a system that integrates those algorithms using the recommender to demonstrate how a system based on our design could help building managers select appropriate algorithms.

Our BAS control architecture is based on the blackboard architectural pattern, which permits multiple interchangeable modules to collaborate on finding a solution to a particular problem [3]. Each module is capable of processing some events that may be added to the blackboard by other modules. A central director is responsible for receiving events that are placed on the blackboard and assigning those events to other modules for processing. This loosely-coupled architecture permits the recommender system to make recommendations for individual components within the system.

The recommender system uses a central database of control algorithm ratings contributed by participating building managers to provide recommendations as to which control algorithms should be used. When the building manager selects a new algorithm to be used, the system dynamically swaps that algorithm into the place of the algorithm currently providing the functionality of the new algorithm without interrupting the BAS' operation. The salient feature of the recommender system is that it weights control algorithm ratings from managers of similar buildings more heavily than other ratings when generating recommendations for a particular building's manager. Several aspects of the building and its environment are modeled in the system and used as a basis for such weightings.

The rest of this paper is divided as follows. In Section 2, we discuss related work and provide background on current trends in building control systems and recommender systems. In Section 3, we present a detailed design of our blackboard architecture for building automation. In Section 4, we describe a recommender system that can be used to select individual control algorithms. In Section 5, we describe our prototype implementation of this approach. In Section 6, we evaluate our approach on the basis of our implementation. Finally, we conclude and describe future work in Section 7.

2. Related Work and Background

The Neural Network House (NNH) used artificial neural networks to automatically adapt various aspects of a house to the desires of its occupants, while also helping to minimize energy consumption [4]. The primary motivation for its development was the observation that residential occupants are unwilling to program home automation devices, even those that are relatively simple, such as VCRs. Thus, the house is capable of training itself. After the system is initialized, it sets all adjustable components in the house to their lowest energy states, forcing the occupant to manually adjust those components to more comfortable settings. The neural network accepts these adjustments as input and subsequently automatically implements those settings when it detects the occupant. After some time, the neural network gradually adjusts each setting back towards a more conservative level, always pushing towards an energy-conserving equilibrium. The occupant is allowed to resist this trend when the neural network interferes with comfort or convenience.

The MavHome project is more wide-ranging than NNH in its application of technologies from artificial intelligence, machine learning, databases, mobile computing, robotics, and multimedia in creating an entire smart home that acts as an intelligent agent [5]. It uses an occupant activity prediction scheme based on Lempel-Ziv (LZ) and Prediction by Partial Match (PPM) compression methods. This scheme is an efficient implementation of a Markov predictor. In our prototype, we include an LZ-based Markov predictor inspired by the description of MavHome, but it lacks the optimizations described in that paper.

As far as we know, recommender systems have not been previously used to select among possible algorithms for controlling smart homes. In fact, we were unable to find any works that use recommender

systems to select among various algorithms at all. However, recommender systems are widely-used in other domains [6]. The purpose of a recommender system is to predict how a particular user will rate some item based upon characteristics of the item as compared to other items, ratings from other users, and/or ratings from the same user of other items. Recommender systems are typically classified into three categories. *1) Content-based recommendations*: Users are recommended items similar to those that they have consumed in the past. *2) Collaborative recommendations*: Users are recommended items that people with similar characteristics rated highly. *3) Hybrid approaches*: Hybrid recommenders combine content-based and collaborative methods. Because it can be very difficult to characterize the content of algorithms in a general manner, we rely on collaborative recommendations in our system.

An important component of our prototype is a Non-Intrusive Load Monitoring (NILM) algorithm that can analyze the energy consumption of a segment of a building and determine what appliances are in use. Several NILM algorithms have been developed, and we implement two of them in our prototype. The seminal work on NILM was performed by G.W. Hart and classifies loads into three categories based on their power consumption profiles [7]. The categories are: *1) ON/OFF*: the appliance has only two states, meaning that it can be turned on or off; *2) Finite State Machine (FSM)*: the appliance has more than two discrete states, such as a fan with three speed settings; *3) Continuously Variable*: the appliance has a large or practically infinite number of states, such as a lamp on a dimmer control. This work also presented a clustering-based algorithm for detecting transitions between discrete appliance states based on predetermined profiles of those states. We implemented that algorithm in our prototype, along with a simple brute-force algorithm that performs a constrained 0-1 Knapsack search of all possible appliance state combinations. Many other NILM algorithms have been developed, but they typically require electric meters with high sampling rates that are too expensive for use in residential applications.

Once control decisions have actually been made, they must then be implemented in the BAS. One approach for doing so is embodied in the Meter Gateway Architecture (MGA) [8]. It explicitly provides a pathway for incrementally deploying intelligence throughout a home or building with multiple loci of control that interact with each other in a complementary fashion. This is beneficial because some devices

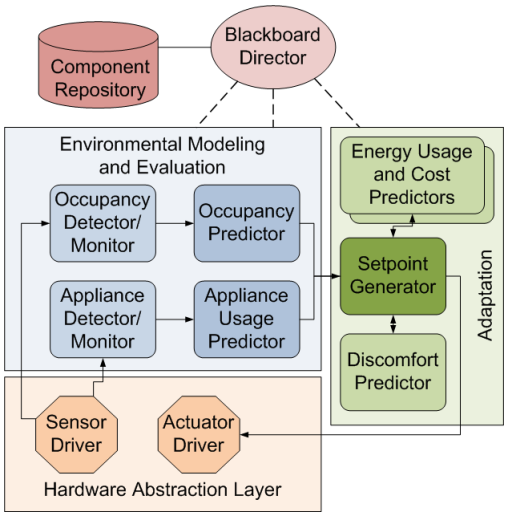

Figure 1. Blackboard architectural pattern used to coordinate the interactions between modules in the system.

are so inexpensive that it is infeasible to integrate control functionality into them, while others are more advanced and must use specialized control algorithms to attain optimal performance.

3. Blackboard BAS Architecture

The blackboard architectural pattern that was developed by the artificial intelligence community [3] can be used to loosen the coupling between control algorithms. An overview of a possible architecture is depicted in Figure 1. The blackboard is a software object that hosts several cooperating modules and coordinates communication between them. All modules have the objective of solving some "problem" that is posted on the blackboard, and they accomplish that by taking information from the blackboard and putting other information back onto the blackboard. We refer to these pieces of information as "messages," since they are treated somewhat similarly to unicast or multicast network messages with blackboard modules playing similar roles as network hosts. Ultimately, one of the modules places a solution to the problem on the blackboard. One of the primary advantages of that pattern is the loose coupling it provides between modules. None of the modules communicate directly with each other; all of the communication occurs through the blackboard. This provides a simple abstraction of computation that permits independent development of modules and also makes it possible for the blackboard to "hot-swap" modules without interrupting the computation of a solution.

Our system makes use of several types of modules to implement intelligent building control, as shown in Figure 1. We describe each of these modules below.

Sensor driver modules are responsible for monitoring the physical environment and perhaps some virtual environment based on state information obtained from the network or some other source. Sensors also generate messages in response to notable events in those environments. For example, an electric meter interface module is a software sensor that generates a message when it receives a new usage indication from the physical sensor it is monitoring. The software and hardware interfaces connecting software sensors to underlying physical sensors are undefined and may involve communication over USB or serial ports, PCI busses, IP networks, etc.

Actuator driver modules receive messages that are intended to change the state of some appliance and must manipulate a physical or virtual environment accordingly. For example, some module may generate a setpoint to turn a specific lamp on, and if an actuator module that is capable of implementing that setpoint is registered with the blackboard, it must do so and inform the blackboard that the setpoint has been implemented. Again, the hardware and software interfaces that are used to perform this process are left undefined. If no actuator exists to handle some particular setpoint, the blackboard must instruct the building manager to manually implement the setpoint.

Energy modelers receive sensor messages from any electric meters registered with the blackboard and generates building-wide views of energy consumption. Energy modelers also make an up-to-date instance of this map available on the blackboard on a periodic basis. Essentially, the energy modeler serves to aggregate and synchronize electric meter readings that are concerned with some particular Current Transformer (CT) of a building and that may arrive asynchronously.

Appliance usage detectors may translate sensor readings into indications that some appliance is in use and/or may analyze energy usage maps to infer the state of an appliance using NILM algorithms. Many BASs include interfaces to either directly query appliances to determine their states (on or off, dimmer level, thermostat setpoint, etc.) or have remote controls that can receive commands used to adjust appliance state, and thus, infer the state of the appliance. Sensor readings from any such BAS can be translated into appliance usage indications. Periodically, the appliance usage detector produces an appliance usage map that aggregates all these indications and information obtained from power analysis for use by other modules.

Appliance usage predictors analyze maps generated by appliance usage detectors and generate predictions from that information, since some modules use predictions of what appliances will be used in the future to make decisions. For example, a kitchen light controller may be interested in predictions regarding when a kitchen appliance, such as a coffee maker, will be turned on, so that it can preemptively activate the kitchen light and thus improve occupant safety or comfort.

Occupancy detectors translate sensor readings into occupancy indications and may analyze energy usage maps to infer occupant locations. Many modules use occupancy data for locations in the controlled building to make decisions. Currently, we only consider confidence levels that some location is occupied and make no distinction between various identifiable occupants within the building or the activities they are currently performing. Sensor inputs from motion detectors may be directly translated into occupancy indications. Alternately, the use of an appliance in a particular location may imply that the location is occupied. Occupancy detectors periodically make occupancy maps available to the blackboard.

Occupancy predictors are similar to appliance usage predictors, but predict the future occupancy status of locations instead of the future appliance usage.

Setpoint generators consider the maps produced by appliance usage and occupancy detectors and predictors, and possibly also the energy modeler directly. The setpoint generators use those inputs to generate updated setpoints for appliances recognized by the system. For example, if the occupancy detector indicates that a particular room is newly occupied at 9:32PM, and an internal model maintained by the setpoint generator indicates that occupants lower the temperature setpoint of an air conditioner installed in the room whenever they enter it between the hours of 8:00PM and 4:00AM, the setpoint generator may generate a setpoint to automatically perform that action. To help evaluate possible setpoints, setpoint generators can use the following three modules.

Energy usage predictors transform an appliance usage map into a prediction of how much energy will be consumed by those appliances when they occupy the states given in the map.

Energy cost predictors transform energy usage maps into corresponding predictions of how much it will cost to consume that energy at a particular time. These modules can implement complex power cost models to accommodate real-time electric pricing, subsidy thresholds, and other factors.

Discomfort predictors can model tradeoffs between occupant comfort and convenience versus energy costs associated with various setpoints and thus predict the future comfort of occupants given specific setpoints.

The blackboard internally implements an implicit publish-subscribe model for managing communications. It recognizes a set of well-defined relationships between modules, as described in the previous paragraphs, and forwards messages accordingly.

4. Recommender System

The system recommends specific modules that can be installed in the blackboard. Any of the categories described above may contain many different modules that can be more or less useful in a given system instantiation. In Section 2, we described several types of appliance usage and occupancy prediction algorithms and different appliance detection algorithms. We described the tradeoffs between those algorithms that make them more or less suitable for various circumstances. The recommender system should help building managers explore those tradeoffs and ultimately select modules that satisfy their requirements.

Recommender systems themselves support many types of algorithms, so it was necessary for us to consider the requirements of this application and design an appropriate recommender system. It is unlikely that content-based recommenders will be suitable for selecting among building control algorithms, since it is difficult to characterize algorithms. It is not possible to completely dismiss this possibility, since it may be useful to coarsely classify algorithms according to their objectives (maximum energy savings, balanced approach, maximum comfort, etc.), but we believe that there are more useful ways of representing and evaluating tradeoffs than through static categories, as we will explain below.

Different types of BAS control modules and algorithms must be rated on different aspects of their performance. Appliance usage and occupancy detector and predictor algorithms must be rated according to their accuracy, while setpoint generator algorithms must be rated according to the comfort and energy cost savings that they provide. Other aspects may be interesting to building managers and can be discovered using surveys or other human polling techniques. Sensors and actuators are ordinarily simple software modules, similar in function to Operating System (OS) device drivers. However, even in the OS device driver domain, some devices do have multiple drivers that provide distinguishing features. It is thus conceivable that some of those modules could also be rated, but unlikely. Energy modelers are very simple, so it should not be necessary to rate them, since it is inconceivable that any value-added functionality could be integrated into them.

Several types of content-independent recommender algorithms, also known as prediction techniques, can be used to aggregate ratings of algorithms from multiple users and use those ratings to predict the rating a new user would provide for a particular algorithm. One of their drawbacks is that many users may not be motivated to submit ratings to the system, since they do not directly benefit from spending their time in that way. Some successful recommender systems, such as that used by Amazon, can infer a user's rating of items based on other actions that are directly useful to the user, such as purchasing particular items. Such techniques may be applicable here, using inferences from building sensors. Regardless, we considered four possible algorithms during our design process: *1) Already Known*: returns whatever rating the building manager has already provided for the algorithm; *2) User Average*: returns the average of all ratings provided by the building manager for all algorithms; *3) TopN Deviation*: returns the average of all normalized ratings of the algorithm provided by other users, normalized for the current building manager; *4) Social Filtering*: returns the weighted average of all normalized ratings of the algorithm provided by other users, normalized for the current building manager, where the similarity between each user and the current building manager is used to determine the weights.

The Already Known algorithm is trivial and obviously useful, so we use it whenever the building manager has already rated an algorithm.

The User Average algorithm is only useful for determining how satisfied the building manager is with algorithms he has tried in the past and predicts that he will be equally satisfied with all untried algorithms. This prediction technique can help a user to allocate his resources between different types of content. For example, if a user is consistently more satisfied with magazines than TV shows, and they are rated using independent recommender systems, the User Average algorithm when applied in each of those recommender systems will predict that the user will be more satisfied by continuing to read magazines rather than watching TV. However, we assume the building manager has already committed to using an automated BAS and is unwilling to abandon it even if he is generally dissatisfied with all algorithms tried in the past, since there is no good substitute for the BAS. Thus, we do not consider the User Average algorithm further.

The TopN Deviation prediction technique provides different predictions for distinct control algorithms but assigns an equal weighting to all users. This is an invalid assumption in our assumed deployment scenario, since we envision managers of multiple building types using the same recommender system to select control algorithms for different types of buildings. Managers of residential and commercial buildings may share control algorithm ratings with each other, but their buildings may have vastly different characteristics. Thus, the TopN Deviation algorithm is only suitable for deployments in which a single recommender system only serves managers of very similar buildings.

Social Filtering overcomes this limitation by assigning different weights to the ratings of different users according to their similarity to the building manager in question. In reality, control algorithms affect buildings themselves, so Social Filtering must weight users according to how similar their buildings are to the building managed by the manager seeking recommendations. The correlation between two buildings is calculated as follows:

$$S_{a,b} = \frac{|C_a \cap C_b|}{|C_a|}$$

where C_i is the set of characteristics associated with building i. Intuitively, the correlation represents the number of characteristics that are shared by buildings a and b, divided by the total number of characteristics associated with a. All buildings have a fixed set of characteristics, described below, so this is also the number of characteristics associated with building b.

The ratings of other building managers and the correlations between their buildings are used to create control algorithm predictions, as follows:

$$P_{a,y} = \mu_a + \sigma_a \frac{\sum_{i=1}^{n} \left[\left(\frac{R_{i,y} - \mu_i}{\sigma_i} \right) S_{a,i} \right]}{\sum_{i=1}^{n} S_{a,i}}$$

where μ_i and σ_i are the mean and standard deviation, respectively, of all ratings provided by building manager i, and $R_{i,y}$ is the rating of control algorithm y provided by i. If a particular building manager has provided less than two ratings, his rating mean and standard deviation are set to the averages of all other building managers, to permit meaningful predictions to be made immediately to managers who are new to the system.

To optimize the predictions produced by Social Filtering, we suggest at least comparing the following characteristics of a building: *1) Usage*: whether the building is residential, commercial, agricultural, etc.; *2) Climate*: whether the building is located in a hot, cold, or temperate zone, how much precipitation is expected on a yearly basis, etc.; *3) Average Number of Occupants*: how many people can be expected to simultaneously occupy the building on average; *4) Number of Rooms*: number of distinct locations being controlled; *5) Number of Energy Sources*: granularity of energy sources in the building, typically corresponding to the number of circuit breakers installed and individually metered; *6) Number of Appliances*: granularity of device control available to the system. Some of these characteristics are correlated (e.g. number of occupants and number of rooms), but only loosely so.

Recommender systems rely on ratings from users to make recommendations to other users. There are several possible ways to generate these ratings. They can be provided manually by users, or they can be automatically generated if the broader system has a mechanism for automatically evaluating the performance of an algorithm. For buildings that have been in operation for a long time and that have monitored their power consumption and occupant comfort levels, it may be possible to compare the environmental effects of a new control algorithm against those records. For other buildings, it may be necessary to rely on manual ratings by the building manager.

5. Implementation

In this section, we discuss details of our prototype implementation of our architecture.

We implemented a complete prototype of the system described as in the previous section that is capable of either monitoring and controlling a physical prototype of a building using X10 home automation devices and an EnerSure electrical submeter, or alternatively monitoring and controlling a simulated building. The experimental equipment used in the physical prototype is expensive and requires a large amount of space to install, so it was necessary for us to construct simulated buildings to avoid those expenses while still being able to construct a complete prototype system comprising multiple buildings.

We now describe the prototype software that we constructed. The basic recommender system is provided by the Duine toolkit [9], which is a Java package that automatically manages a recommender system database and contains several pre-implemented recommender prediction techniques, including the Already Known and Social Filtering techniques that we highlighted in Section 4. The blackboard architecture was specially constructed for this project, as were all of the blackboard modules. We constructed a graphical interface to

our system using a framework that had been previously developed for the MGA project [8]. The framework was extended to include all of the software infrastructure necessary to interact with the hardware we used to monitor and control the emulated building in our experiments and the several simulated buildings we constructed. The graphical interface allows the building manager to monitor and configure any critical aspect of the system and its interfaces.

We implemented several control algorithms for our experiments, which we now describe. Two of these are appliance usage detectors. The first, called the "Knapsack Appliance Detector," attempts to solve a constrained 0-1 Knapsack optimization problem to "pack" appliance states into the energy consumption measurements observed on all energy sources, where an energy source typically represents an individual circuit breaker or some other independently-submetered segment of the electrical infrastructure in the building. The algorithm does a brute-force search of all possible appliance state combinations, predicts how much energy will be consumed by the appliances on each energy source, and then selects the configuration that most closely approximates the actual energy consumption observed on the energy sources.

The second appliance usage detector was based on the idea of clustering [7], and is referred to as the "Clustering Appliance Detector." The recent change in current consumption on each segment is compared with the reference data on the possible states of each appliance on that segment, which were measured prior to the experiments. Changes that are close to the current consumption of a particular appliance state may indicate that the state was either entered or exited, as appropriate. The reference data we collected consists of the center of a cluster, that is the real current for the states of each appliance.

We implemented three setpoint generators with very different approaches to system control. The first is a simple algorithm that sets all appliances in all currently occupied locations to their highest-powered states. It can also be configured to use occupancy predictions to turn on appliances in locations predicted to be occupied in the near future. This algorithm is only useful in buildings that have automatic controls installed only on simple appliances, such as overhead lights.

The second setpoint generator attempts to maximize occupant comfort while staying within an energy cost bound. It does this by setting all appliances in unoccupied locations to their lowest-power states and testing all combinations of appliance settings in occupied locations, selecting the combination that provides the greatest occupant comfort while remaining within a building manager-defined energy cost bound. Comfort contributions for each appliance state were specified prior to the experiments.

Finally, the most sophisticated setpoint generator uses a neural network to react to changing user requirements and activities by adapting and learning. The neural network consists of two layers and is based on inputs of time of day, day of week and room occupancy. We chose a two-layer neural network because a single layer perceptron network would not allow sufficient expressiveness for envisioned user activity schedules. The network learns when a user provides feedback to the system. If the feedback is positive, the current conditions are provided to the network as a set of training data. If the feedback is negative, the current conditions are complemented and provided to the network as a set of training data.

We implemented two types of sensor modules. The first type attaches to X10 input interfaces or other boolean input interfaces and generates occupancy indications when the appropriate input signal is received. X10 input signals can be generated by IR motion detectors, or manual remote controls. The second type of sensor attaches to an EnerSure submeter and produces energy consumption indications whenever the submeter provides a reading. The submeter is a Modbus device that is capable of simultaneously monitoring the real current consumed by up to 21 devices, and providing current measurements from those devices once every 0.1-3 seconds, depending on how many are enabled. We implemented one type of actuator capable of activating or de-activating power flow to X10-controlled devices.

Our energy modeler module simply aggregates energy usage indications provided by energy sensors into a snapshot of the building's current energy consumption. Likewise, the energy usage predictor in our prototype simply uses the data in the building specification file to transform an appliance usage map into an energy usage map in a straightforward fashion. The same principle is used to implement our simple discomfort predictor, since we have a trivial additive user comfort model.

Our energy cost predictor implements a simple real-time pricing scheme that alters the cost of electricity once every fifteen minutes. In reality, real-time prices would probably be downloaded from a website once per day or perhaps even more frequently, but for our experiments it suffices to statically encode a set of prices that is never updated.

The occupancy detector module not only aggregates

occupancy indications from motion detector sensors but also analyzes the current appliance usage map and adjusts its confidence level depending on whether that location is occupied, which it determines based on the number of appliances that are turned on in that location. An appliance is considered to be turned on if it is not in its default state. The occupancy detector divides the number of activated appliances by the total number of appliances in the location, and uses that value as its confidence level that the location is occupied. However, if an occupancy indication for the location has recently been received, a confidence level of 1 is used, indicating near-total certainty that the location is occupied. Indications expire after a manager-defined period, which is 10 minutes in our prototype.

We implemented occupancy and appliance usage predictors based on the LZ compression scheme to allow us to efficiently encode sequences of actions.

All these modules can be instantiated by the blackboard director, and the setpoint generators and appliance detectors can be rated and explicitly selected by building managers. The other modules are fixed in our initial prototype but could be integrated into the recommender system in future versions.

6. Experimental Evaluation

The goal of our experiments is to demonstrate the effectiveness of our recommender system in generating useful recommendations and also to evaluate the effectiveness of the modules that we have implemented, although they are not our main contribution and thus are not expected to perform as well as carefully-optimized implementations with similar functionality.

To evaluate the effectiveness of our recommender system, we have defined six distinct buildings that will share ratings among themselves. Three are small apartments with similar characteristics but different sets of appliances, two are small retail stores with identical appliances and layouts, and one is an industrial building. The first apartment is physically emulated in our experiments and contains a laptop computer, LCD monitor, and incandescent lamp in the bedroom, a fan and air purifier in the kitchen, and a fluorescent lamp in the bathroom. All of these devices can be switched on and off using X10 controllers. Each room is also equipped with an IR motion detector. This testbed is depicted in Figure 2. The metadata for the remaining buildings simply serves as input to the recommender system.

Our first experiment demonstrates the ability of the recommender to make useful predictions. Besides managers of the third apartment and second retail store, we cause all other building managers to provide ratings for all algorithms, as shown in the top portion of Table 1. One important feature of the recommender system is its ability to compensate for different styles of rating allocation, so we cause the managers of retail store #1 and the industrial building to submit relatively higher ratings, in general, as compared to the two apartment managers to test this functionality. After submitting the ratings in the top portion of the table, we requested predictions for each of the algorithms on behalf of the manager of apartment #3 and then on behalf of the retail store #2 manager. Those ratings are listed in the last rows of the table. The numbers in parentheses are the ratings after being normalized to compensate for each manager's style of rating allocation. It is clear that the recommender system does use the characteristics of buildings when generating personalized recommendations, since the predictions provided to the manager of retail store #2 are more heavily influenced by the ratings provided by the manager of retail store #1 than the other ratings, whereas the ratings provided to the manager of apartment #3 are more heavily influenced by the ratings provided by the managers of the two other apartments than the other ratings.

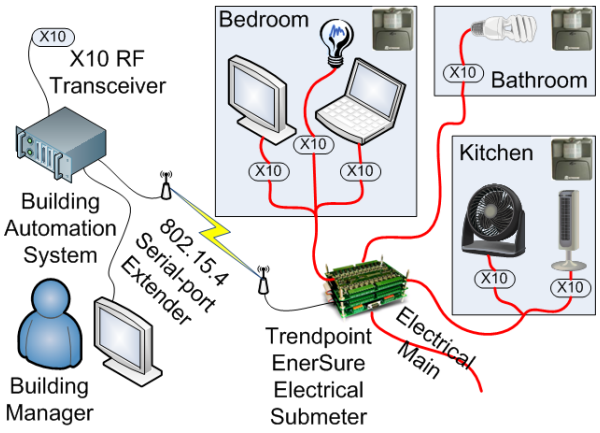

Figure 2. Experimental apparatus used to physically emulate Apartment #1 for evaluation.

Our second experiment evaluates the effectiveness of our knapsack appliance detection module. We monitored each of the virtual locations in Apartment #1 with a distinct CT. Then, we exhaustively created every possible combination of appliance states in our physical testbed, maintaining each combination for around 15 seconds, and recorded the appliances detected at each time step. Appliance detection was performed approximately twice per second, and the energy con-

Table 1. Control algorithm ratings for experiment #1. Ratings range from -1 to 1, and ratings in parentheses are unitless, normalized versions of the ratings to their immediate left.

	Appliance Detectors		Setpoint Generators		
Building	0-1 Knapsack	Clustering	Neural Network	Bounded Knapsack	All-In-Occupied
Apartment #1	0.80 (1.10)	-0.20 (-1.05)	-0.10 (-0.84)	0.90 (1.31)	0.05 (-0.52)
Apartment #2	0.75 (1.25)	-0.30 (-1.14)	-0.15 (-0.80)	0.70 (1.14)	0.00 (-0.46)
Retail Store #1	0.00 (0.54)	1.00 (1.06)	0.20 (-1.01)	0.90 (0.80)	0.05 (-1.39)
Industrial Building	0.80 (0.80)	0.80 (0.80)	0.25 (-0.89)	0.80 (0.80)	0.05 (-1.51)
Apartment #3	0.82 (1.03)	0.18 (-0.55)	0.06 (-0.85)	0.86 (1.12)	0.10 (-0.75)
Retail Store #2	0.70 (0.75)	0.56 (0.39)	0.02 (-0.94)	0.78 (0.94)	-0.05 (-1.14)

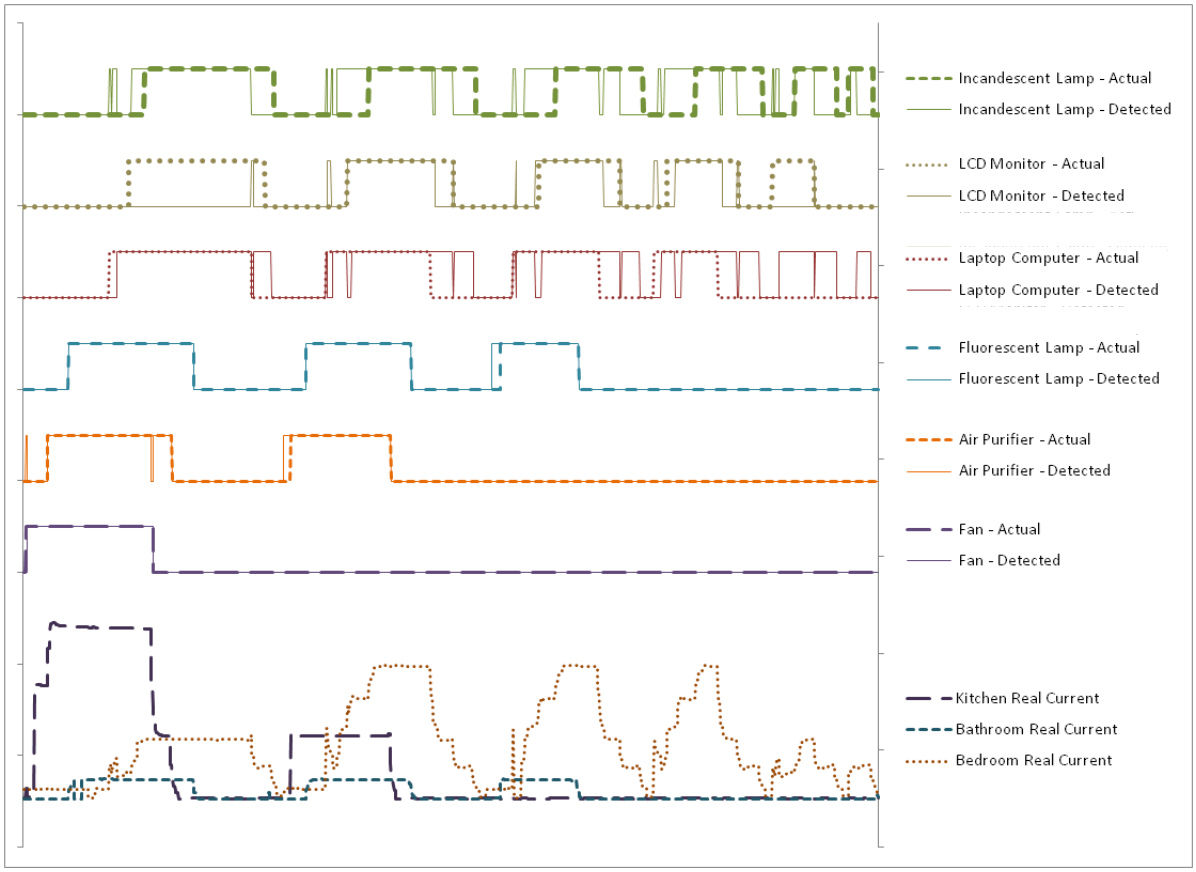

Figure 3. Evaluation data for knapsack-based appliance detector on Apartment #1, with three distinct CTs.

sumption map used by the detection algorithm was updated approximately three times per second. The actual appliance states are displayed along with the detected states and the actual energy consumption on a timeline in Figure 3.

In the absence of reactive current consumption data, our algorithm had significant difficulty distinguishing between appliances that consumed a relatively small amount of energy. In fact, it even failed to detect the air purifier for a significant portion of its runtime, despite it being the second most-consumptive device, after the fan. The statistics for each appliance are provided in Table 2. The unpredictability of the laptop's power consumption also complicated appliance detection in the bedroom of the emulated apartment. This experiment illustrates that it is possible to determine information about what appliances are in use in a building by simply metering and analyzing their aggregate energy consumption. However, they also confirm that simply analyzing appliances' real power consumption can lead to ambiguity in the detection results.

Table 2. Statistics for appliance detection experiments: False Negative (FN) and False Positive (FP) rates, as a percentage of total time.

	Single CT		Three CTs	
Appliance	FN	FP	FN	FP
Incandescent Lamp	22.9	16.8	19.3	14.9
LCD Monitor	21.8	23.0	27.1	1.9
Laptop Computer	24.0	6.3	4.5	24.6
Fluorescent Lamp	6.9	7.7	0.0	1.0
Air Purifier	16.7	6.5	0.2	1.0
Fan	0.9	18.8	0.2	0.0

7. Conclusion and Future Work

We have presented an architecture to help building managers select building control algorithms by using a collaborative recommender system that weights ratings from managers of similar buildings more heavily than ratings from other managers. This should permit building managers to easily select control algorithms that provide greater energy-efficiency and occupant comfort than generic control algorithms designed to operate in a wide variety of buildings. We developed a prototype system in Java that is capable of controlling an emulated physical building or a simulated building in software, and demonstrated that the recommender system does in fact provide recommendations that are tailored to the type of building being managed. In the future we would like to perform an evaluation of our system installed in several actual homes, including at least one that is unaffiliated with our research group, so that we can obtain actual user feedback.

Acknowledgements

This work was supported in part by NSF CNS 07-16626, NSF CNS 07-16421, NSF CNS 05-24695, ONR N00014-08-1-0248, NSF CNS 05-24516, DHS 2006-CS-001-000001, and grants from the MacAruthur Foundation and Boeing Corporation. Michael LeMay was supported on an NDSEG fellowship from the AFOSR. The views expressed are those of the authors only.

References

[1] United States Department of Energy, "Average retail prices of electricity," http://www.eia.doe.gov/emeu/mer/pdf/pages/sec9_14.pdf, April 2008.

[2] United States Treasury - Bureau of Labor Statistics, "CPI inflation calculator," http://data.bls.gov/cgi-bin/cpicalc.pl, April 2008.

[3] B. Hayes-Roth, "A blackboard architecture for control," *Artificial Intelligence*, vol. 26, no. 3, pp. 251–321, 1985.

[4] M. Mozer, "The neural network house: An environment that adapts to its inhabitants," in *Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, 1998, pp. 110–114.

[5] D. Cook, M. Youngblood, E. Heierman III, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "MavHome: an agent-based smart home," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 03)*, 2003, pp. 521–524.

[6] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[7] G. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE*, vol. 80, no. 12, pp. 1870–1891, Dec 1992.

[8] M. LeMay, R. Nelli, G. Gross, and C. A. Gunter., "An integrated architecture for demand response communications and control," in *IEEE Hawaii International Conference On System Sciences (HICSS '08)*, Waikola, Hawaii, January 2008.

[9] M. Setten, J. Reitsma, and P. Ebben, "Duine Toolkit–user manual," *http://www.telin.nl/index.cfm?type=doc&handle=62057&language=en*, 2003.