12 Lessons from an Adaptive Home

MICHAEL C. MOZER
University of Colorado

For the past four decades, the impending arrival of the smart home has been touted in the popular press. The newspaper copy hasn't changed much over the years. Consider this recent example.

> If time is money, then here's a definition of wealth for the 21st century: You're leaving the house for the day. You grab your things, hop into the car and back out of the garage. Then you pick up a device that looks like a four-button version of your car-lock remote control and punch a "goodbye" button. The garage door closes. Inside the house, all the exterior doors are bolted shut, all windows close and lock, drapes swing closed over west-facing windows, the temperature control adjusts to a more frugal level, and the lights are turned off. (San Diego Union Tribune, 8/4/03, *'Smart home' technology creates true digital domain*, p. C1)

Something is inherently wrong with this scenario. The operations described are initiated not by a smart home, but by smart inhabitants who indicate when they are leaving the home. Although the user interface is simple—the push of a button—it is not an action people easily remember to perform. If you doubt, read on.

> Before retiring for the night, ... the Alexanders will set a hello or morning mode, which will wake the family at 7 a.m. to lights, music, or television morning shows. Coffee begins brewing. The thermostat adjusts for family comfort. The hot water kicks in, and hot water fills the tub...
>
> "Our biggest problem so far has been in forgetting to activate the home mode when we arrive home at the end of the day," says Alexander. "My wife, in particular, likes to take a hot bath at the end of her work day, but if she or I forget to press the home mode, there won't be any warm water in the hot water heater for her bath because we've turned off the appliance during the day to conserve electricity." (USA Today, 9/2/97, *Homes get smart with automation systems)*

Even with touch screens and remote controls to simplify the user interface, home inhabitants have been less than satisfied with installed automation systems.

> Flipping a light switch, bumping down a thermostat and changing the channel on the TV are all exercises that homeowners around the world perform intuitively several times each and every day. You'd think that an automation system would magically eliminate these tasks completely from the daily routine. At least that's what Dave and Sharyl Faganel figured when they decided to have an automation system installed in their new... home... But as with any technology, it takes a good deal of time at the controls to feel completely comfortable living with and using an automated system. Unfortunately, a little too much time for the Faganels...[A]fter weeks of wrestling with the setup menus, Dave and Sharyl realized that they and that

> particular automation system were simply a bad match. The only solution at this point was to tear out the system..."We spent a good 40 hours showing them how to set up temperature schedules on the old keypad," says [home automation system installer] Michael. (*Electronic House*, February 2003, *A period of adjustment*, pp. 52–59)

If Dave and Sharyl are technologically challenged, they have plenty of company:

> “When I consider all the aggravation and the money, I wouldn’t spend as much again the next time, said Tiburon homeowner Bob Becker. In 1991, he built a high-tech castle atop a hill...with about $70,000 in electronic upgrades. “I use about half of it,” said Becker... (San Francisco Examiner, 4/14/96, *The smart house)*

> The Christmas party at the ... house of Robert Soros ... was in full swing...Children raced upstairs to experience...a home theater screening of...“Peter Pan.” The large screen descended from the ceiling with an extraterrestrial hum, but the house lights refused to dim—despite a horde of caterers stabbing frantically at a panel of small buttons on the wall. As bright spots flashed and faded overhead, Mr. Soros...bent prayerfully over a control panel, playing its glowing touch screen like a pipe organ as he tried to raise the volume on Tinkerbell. He had given up when the sound suddenly surged to IMAX level. Children screeched. “I would give anything to go back to good old toggle switches,” his wife...said later, sighing at the memory. (New York Times, 1/13/00, *When smart houses turn smart aleck*, p. B1)

> Elliot Fishkin...is one of the new digital caretakers, on 24-hour call, seven days a week...to indulge human frailty. Mr. Fishkin watched as one client sold a house lock, stock, and remotes soon after it was wired because he found the controls too daunting. (New York Times, 1/13/00, *When smart houses turn smart aleck*, p. B1)

Anecdotes like these are common. Smart homes have failed to become a reality for two reasons. First, inhabitants are fairly satisfied with traditional home controls. Second, the obstacle to understanding new interfaces is high. Technology will be adopted only if the perceived return outweighs the effort required to understand the new technology. Cell phones have become commonplace because they offer many benefits at a relatively minor cost to master. In contrast, personal digital assistants (PDAs) are not commonplace outside the business and academic communities, because the benefits to other individuals are small relative to the significant overhead required for mastery.

Even when designed by experts, user interfaces often fail due to the user’s perception of benefit versus cost. For example, when I was a graduate student at UC San Diego, my Ph.D. advisor, Donald Norman, well known for his popular books on the design of user interfaces and everyday consumer products, had a pet project to apply the principles proposed by his User-Centered System Design group to develop a novel interface for the lights in his laboratory. The original light panel was the standard row of a half dozen switches, and like all such panels, one could not determine a priori which switch controlled which light, because there was no natural mapping between the switches and the lights. The innovative replacement involved cutting a wedge from the wall and laying an illuminated map of the lab horizontally in the wedge. Switches were placed at the locations on the map corresponding to the locations of the lights they controlled. Although this design was ingenious and innovative, it was only moderately successful. Because the lab layout was complex, the map was too, and reading the map was nontrivial especially for someone unfamiliar with the lab. Further, the correspondence between corners of the map and corners of the room depends on whether one perceives the map to be of the floor or ceiling; if perceived as a ceiling map, one could mentally flipped the map around, reversing the coordinates by aligning the left edge of the map with the right edge of the ceiling. Because of these difficulties, it was easiest to simply tap all the switches simultaneously with a swipe of the hand. In a lab meeting, Dave Rumelhart suggested tongue-in-cheek that Don Norman write a user manual for the device and place it on the door leading to the switches.

Even if Don’s effort was not fully appreciated, it was compelling relative to some commercially available lighting solutions. Figure 1 shows two different existing lighting controllers for residences. The controller on the left allows the inhabitant to program the lights to turn on based on the time of day and the day of the week, ideal if your weekly schedule is rigidly bound to the clock. The controller on the right allows for various lighting scenarios based on activities, e.g., watching TV or dining, as long as the user is willing to fish through the array of buttons to find the correct scenario.

For lights, stereos, TVs, and other devices we encounter in daily life, *any* sort of novel interface is likely to fail, simply because the benefits offered by smart control are unlikely to be compensate for the fuss of learning to use the interface, especially when existing familiar interfaces are adequate for most purposes. Indeed, one might make the strong conjecture that the only way to improve an ordinary familiar environment is to eliminate the interface altogether.

Such an invisible interface is found in BMW (and perhaps other) automobiles. Vehicles have a memory for the configuration of the driver's seat, the steering wheel, and the outside mirrors. Three distinct memory settings are available, allowing for a customized configuration for up to three different drivers. A naive interface might require the drivers to identify themselves each time they entered the vehicle in order for the correct configuration to be chosen. Instead, the interface is hidden from the user by associating a different memory with each car key, and inserting the key in the ignition or pressing the remote entry button selects the memory configuration associated with the key.

Another example of an invisible interface is the *Internet Chair* (Cohen, 2003). The chair senses the direction that users are facing, and adjusts the soundscape presentation appropriately so as to achieve orientation invariance: users hear the same binaural sound regardless of the direction they are facing, as if they were wearing headphones. The chair is noninvasive and does not require explicit user actions (such as pointing) or complex technology (such as computer vision).

12.1 THE ADAPTIVE HOUSE

During the past eight years, we have proposed and implemented a smart home environment premised on the notion that there should be no user interface beyond the sort of controls one ordinarily finds in a home—no touch pads, no speech input, no gaze tracking or hand gesturing, etc. The intelligence of the home arises from the home's ability to predict the behavior and needs of the inhabitants by having observed them over a period of time. We have focused on home comfort systems, specifically air temperature regulation, water temperature regulation, and lighting. Instead of being programmed to perform certain actions, the house essentially *programs itself* by monitoring the environment and sensing actions performed by the inhabitants (e.g., turning lights on and off, adjusting the thermostat), observing the occupancy and behavior patterns of the inhabitants, and learning to predict future states of the house. To the extent that the inhabitants' needs can be anticipated, the inhabitants are freed from manually controlling the environment via some type of interface. When the predictions are incorrect, the inhabitants can simply indicate their preferences via ordinary interfaces they are used to, e.g., light switches, thermostats, and simply turning on the hot water.

Scenarios in the home include the following: On a rainy weekday when the inhabitant leaves the home at 8 a.m. and on the previous three days had returned by 7 p.m., the home predicts a return by 6:30 and runs the furnace in order to achieve a setpoint temperature by that time. When the inhabitant does return, he prepares dinner and banks of lights in the kitchen and great room are turned on full intensity. The inhabitant then relaxes on a couch and watches TV in the great room; lights are turned off behind the TV and are dimmed elsewhere in the room. On a weekend when the inhabitant leaves the house at 4 p.m., his return is not anticipated until past midnight, and the house is not heated before then. When the inhabitant does return home at 1 a.m., the prediction is made that only the master bedroom will

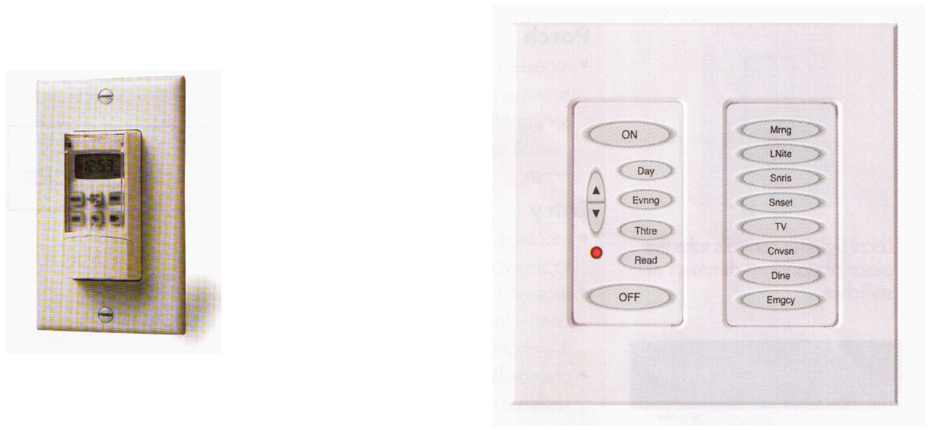

FIGURE 1. Examples of two lighting controls for residences.

be used for the next 7 hours, and instead of running the whole-house furnace, electric space heaters in the bedroom are used for generating heat; the hot water heater lowers its setpoint based on the prediction that hot water will not be required for at least another 7 hours. When the inhabitant awakens at 4 a.m. and climbs out of bed, the home predicts a trip to the bathroom, and the bathroom light is turned on at a low intensity before the inhabitant reaches the bathroom.

We have focused on home comfort systems because the correction involved in a misprediction is trivial, e.g., if the house failed to anticipate the return of the inhabitants, it can simply turn on the heat or air conditioning when the inhabitants actually return; or if the house does not correctly turn on a light, the inhabitants can do so themselves. Predictions are based on statistical regularities in the behavior and preferences of inhabitants. If these regularities were based solely on time of day and day of the week, the accuracy of predictions would be severely limited. However, by basing predictions on dozens of variables (e.g., for predicting occupancy patterns, variables include recent room occupancy patterns, outside weather, times of recent departures and returns, home occupancy patterns on the same day of the week in recent weeks, and home occupancy patterns at the same time of day on recent days), subtle higher-order regularities can be discovered and exploited.

The *adaptive house* is an actual residence—my own—in Boulder, Colorado, that has been outfitted with over seventy-five sensors that monitor various aspects of the environment, including room temperature, ambient light, sound level, motion, door and window positions, and outside weather and insolation. Actuators control air heating via a whole-house furnace and electric space heaters, water heating, lighting, and ventilation. We call the control system in the adaptive house *ACHE*, an acronym for Adaptive Control of Home Environments. Our research involves several subprojects, each with its own challenges and peculiarities, but in this chapter I focus on the lighting control problem.

12.2 LIGHTING REGULATION

To regulate lighting, ACHE specifies the setpoint of twenty-two independently controlled banks of lights, each of which has sixteen intensity settings. Most rooms have multiple banks of lights, e.g., the great room (containing the entertainment center, dining table, and kitchen) alone has seven banks of lights. ACHE is trained via actions taken by inhabitants to adjust the light setpoints (via dimming switches). ACHE is also influenced by energy consumption, in a manner we describe below. Figure 2 summarizes the framework in which ACHE operates.

12.2.1 What Makes Lighting Control Difficult?

Lighting control in the Adaptive House is a difficult task for a variety of reasons.

- More than simply turning lights on and off, lighting control involves setting lighting *moods*—the pattern and intensity of lighting. In a room used for multiple activities (e.g., a living room might be used for entertainment, reading, or watching television) and having several independently controlled banks of lights, determining the appropriate lighting mood is nontrivial.
- Although motion sensors can detect occupancy, it is not sufficient to simply switch on lights when motion is sensed. If one rolls over in bed at night, the lights should not go on. If one sits still in a chair while reading, the lights should not go off after a motion time-out period. Further, there is a 700 ms time lag between the firing of a motion sensor and the response of ACHE. This is due almost entirely to an inefficient and archaic protocol (X10) for sending commands to the lights. This lag is long enough to inconvenience the inhabitant.

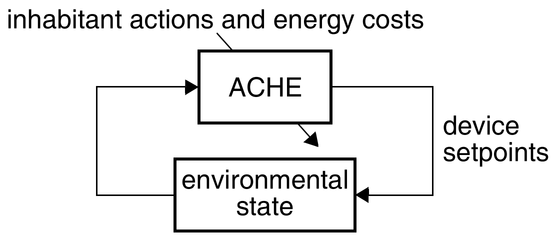

FIGURE 2. ACHE specifies device intensity setting (brightness), which affects the state of the environment, which in turn serves as input to ACHE. The training signals to ACHE are the actions taken by the inhabitants and energy costs.

- The range of time scales involved in lighting control spans many orders of magnitude. Control decisions must be responsive—in a fraction of a second—to changing environmental conditions. However, decisions can have implications that span many hours, e.g., in energy consumption.
- Two constraints must be satisfied simultaneously: maintaining lighting according to inhabitant preferences and conserving energy. These two constraints often conflict. For example, leaving all lights on may be sufficient to satisfy the inhabitants, but will be costly. Similarly, if minimizing energy consumption is the goal, lights will never be turned on.

12.3 OPTIMAL CONTROL

In what sort of framework can the two constraints—appeasing the inhabitants and conserving energy—be integrated? Supervised learning will not do: If lighting device settings chosen by the inhabitant serve as targets for a supervised learning system, energy costs will not be considered. Instead, we have adopted an *optimal control* framework in which failing to satisfy each constraint has an associated cost. A *discomfort cost* is incurred if inhabitant preferences are not met, i.e., if the inhabitant is not happy with the settings determined by ACHE and chooses to manually adjust the light settings. An *energy cost* is incurred based on the intensity setting of a bank of lights. The *expected average cost*, $J(t_0)$, starting at time t_0 can then be expressed as

$$J(t_0) = \mathrm{E}\left[\lim_{\kappa \to \infty} \frac{1}{\kappa} \sum_{t=t_0+1}^{t_0+\kappa} d(\boldsymbol{x}_t) + e(\boldsymbol{u}_t)\right]d$$

where $d(\boldsymbol{x}_t)$ is the discomfort cost associated with the environmental state $\boldsymbol{x}$ at t, and $e(\boldsymbol{u}_t)$ is the energy cost associated with the control decision $\boldsymbol{u}$ at t. The goal is to find an optimal control *policy*—a mapping from states $\boldsymbol{x}_t$ to decisions $\boldsymbol{u}_t$—that minimizes the expected average cost.

This description of the control problem assumed a quantization of time into intervals indexed by t. Most research in optimal control treats these intervals as clock based. However, the framework applies equally well to intervals produced by an event-based segmentation. We exploit this observation.

12.3.1 Reinforcement Learning

Although dynamic programming (Bellman, 1957) can be used to find a sequence of decisions $\boldsymbol{u}$ that minimize J, it has two significant drawbacks. First, it requires a model of the environment and the immediate cost function. Second, computing expectations over highly uncertain future states can be expensive. Consequently, we use *reinforcement learning*, a stochastic form of dynamic programming that samples trajectories in state space.

The particular version of reinforcement learning we consider is called *Q learning* (Watkins, 1992; Watkins & Dayan, 1992). Q learning provides an incremental update algorithm for determining the minimum expected discounted cost given that action $\boldsymbol{u}$ is taken in state $\boldsymbol{x}$:

$$Q(\boldsymbol{x}_t, \boldsymbol{u}_t) \leftarrow (1-\alpha)\, Q(\boldsymbol{x}_t, \boldsymbol{u}_t) + \alpha \max_{\hat{\boldsymbol{u}}}[c_t + \lambda\, Q(\boldsymbol{x}_{t+1}, \hat{\boldsymbol{u}})]$$

where α is the learning rate, λ is the discount factor, c_t is the immediate cost incurred at t, and $\hat{\boldsymbol{u}}$ is an index over all possible actions. Once this algorithm converges, the optimal action to take is the one that minimizes the Q value in the current state. Because the algorithm requires that all states be visited to learn their Q values, the control policy, $\pi(\boldsymbol{x}_t)$, requires exploration:

$$\pi(\boldsymbol{x}_t) = \begin{cases} \operatorname{argmin}_{\boldsymbol{u}} Q(\boldsymbol{x}_t, \boldsymbol{u}_t) & \text{with probability } (1-\theta) \\ \text{random} & \text{with probability } \theta \end{cases}$$

where θ is the exploration rate. Given a fully observable state and an infinite amount of time to explore the state space, Q learning is guaranteed to converge on an optimal policy.

12.3.2 Temporal Credit Assignment and the Issue of Time Scale

Figure 3 presents an alternative view of the sequential decision framework. At the beginning of each time interval, the current state is observed and a control decision must be made. Following the decision, a cost is observed. This cost can be attributed to the current decision or to any earlier decision, as depicted by the arrows. The *temporal credit assignment problem* involves determining which decision or decisions in the sequence are responsible for the observed costs. The challenge of learning is to correctly assign credit in time (Sutton & Barto, 1998).

On the surface, the temporal credit assignment problem for lighting control seems exacerbated due to the range of time scales involved. Because control decisions must be responsive to changing environmental conditions, the time interval between decisions must be brief, on the order of 200 msec. However, the shorter the time interval, the more difficult the temporal credit assignment problem becomes. Consider a decision to turn on lights when a room becomes occupied. As long as the room remains occupied, an energy cost is incurred at each time interval and must be attributed to the initial decision. With a 200 msec fixed interval and an hour-long occupancy period, this amounts to 18,000 time intervals over which credit assignment must be performed. Consider another scenario: The inhabitant enters a room, ACHE fails to switch on the light, and the inhabitant does so manually. The gap between the room entry and the manual override might be about five seconds, meaning that punishment for failing to turn on the light must be propagated back 25 time intervals.

12.4 MAKING THE LIGHTING CONTROL PROBLEM TRACTABLE

Although the standard approach to sequential decision problems using reinforcement learning involves a clock-based segmentation that divides the stream of time into uniform intervals whose durations correspond to the finest time grain of the domain, we have argued that this approach is unlikely to succeed for lighting control. ACHE would require orders of magnitude more training than any inhabitant would be willing to provide. For this reason, we were forced to consider an alternative to clock-based segmentation. Using event-based segmentation, along with three other techniques that take advantage of peculiarities of the lighting domain, the temporal credit assignment is eliminated and learning becomes almost trivial. In the following sections, we describe the key features of our solution that simplify the control problem.

12.4.1 Decomposing the Task Based on Lighting Zones

To a first order, the setting of a light in one *zone* (room) will not affect the ambient light level in another zone, nor will it affect inhabitant preferences in other zones. This is not strictly true, because light in one zone may spill into another, but by assuming independence of state and lighting decisions across zones, one can decompose the overall control problem into multiple smaller problems. This type of decomposition is extremely helpful because standard reinforcement learning methods do not have any way of taking advantage of compositional structure in the decision space, i.e., 22 banks of lights with 16 intensity settings each would result in a decision space of 16^{22} alternatives. The Adaptive House is naturally divided into eight lighting control zones, the largest of which has seven banks of lights and the smallest has only one. In the remainder of this paper, we focus on the control task for a particular zone.

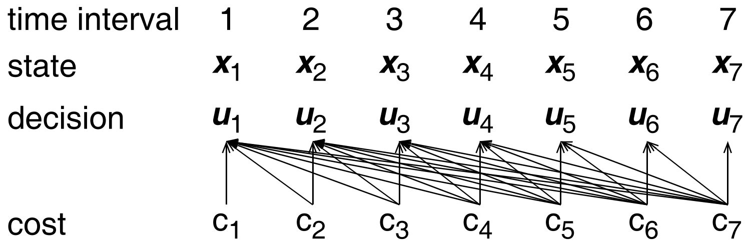

FIGURE 3. The temporal credit assignment problem

12.4.2 Defining Time Intervals Using Event-Based Segmentation

The key to an event-based segmentation is an orienting mechanism that determines salient events. These events were defined to be:

- zone entry
- zone exit
- a significant change in the outdoor light level
- change in inhabitant activities (as indicated by the fact that the inhabitant manually adjusts light settings after having been satisfied with the previous settings for more than two minutes)
- in the largest zone, the great room, movement from one region of the zone to another
- anticipation of a zone entry

We discuss the mechanics of how these events are detected in later sections. When an event is detected, a lighting control decision is made. The window of time between events is treated as the basic interval. It is unitary in that only one lighting decision is made within the window, at the start of the window. Consequently, a discomfort cost—when the inhabitant manually overrides the device settings selected by ACHE—can be incurred at most one time within the window, and is attributable to the decision made at the start of the window. Similarly, energy costs can be summed over the window and attributed to the decision made at the start of the window.

Because costs do not extend beyond the window, it might seem that the problem of temporal credit assignment is avoided. However, this is not necessarily the case, because a decision made at one time can affect the future state of the environment which in turn can affect future decisions and hence future costs.

12.4.3 Eliminating Long-Term Consequences of Decisions

In the lighting control domain, the long-term consequences of decisions can in fact be eliminated due to three additional properties of the domain:

- The effect of a decision on a device is completely undone by a subsequent decision. This is true only if the decisions indicate absolute, not relative, device settings.
- Inhabitant activities are basically unaffected by ACHE's decisions. The inhabitant may have to switch on a light if ACHE fails to do so, but the inhabitant will not stop preparing dinner and instead watch television if a light doesn't turn on.
- The current device settings are irrelevant to decision making. Thus, they need not be considered part of the environmental state.

The net consequence of these properties is that the current environmental state does not depend on earlier decisions. By eliminating the long-term consequences of decisions and using event-based segmentation, we establish a finite horizon on the effect of a decision—the end of the event window. Lighting control thus becomes a single-stage decision problem, and the temporal credit assignment problem vanishes.

12.4.4 Getting the Most of Each Experience

In the standard reinforcement learning framework, a controller that makes some decision ***A*** will learn only about the consequences of ***A*,** not any other decision ***B***. This is because the two decisions ***A*** and ***B*** are unrelated. However, in the case of lighting control, the decisions represent device intensity settings and hence have an intrinsic relationship to one another. With additional domain knowledge, ACHE can learn about some choices that were not selected.

Specifically, suppose that ACHE decides to set a device to intensity ***A***, and the inhabitant overrides ACHE and sets the device to intensity ***C***, thereby incurring a discomfort cost for decision ***A***. If ***A*** was lower than ***C***, then any ***B*** which is lower than ***A*** will also incur the discomfort cost. Similarly, if ***A*** was higher than ***C***, then any ***B*** which is higher than ***A*** will also incur the discomfort cost. Because the total cost is the sum of the discomfort cost and the energy cost, and the energy cost can be computed based on trivial knowledge of the device, ACHE can determine the cost that *would have been incurred* had it made decision ***B***. Thus, although reinforcement learning generally involves learning from experience, ACHE can learn about the consequences of some decisions it did *not* experience because it has a partial model of the cost function.

12.5 ACHE ARCHITECTURE

Figure 4 sketches the overall architecture of ACHE. Starting on the right side of the Figure, the *Q learning controller* selects device intensity settings based on the current state. The *event-trigger mechanism* acts to gate the controller such that decisions are made only when salient events have been detected. The controller receives a training signal, in the form of a total cost and depicted in the Figure by the arrow cutting through the controller, from the *cost evaluator*. Cost evaluation is performed when an event is detected, and hence the cost evaluator is also gated by the event-trigger mechanism. The cost evaluator needs to know when the inhabitant manually overrides the device settings produced by the controller, and hence it receives input in the Figure from the light switches. The inhabitant can adjust the intensity of a device as well as switch it on or off; this information is also provided to the cost evaluator.

The state used for decision making is provided by the *state estimator*, which attempts to form a high-level state representation that explicitly encodes information relevant for decision making. In particular, we view two types of information as central: inhabitant activities and the level of natural light in the zone. Inhabitant activities cannot easily be determined from the available sensor data, but we can characterize the activities in a coarse way by observing short-term occupancy patterns across zones in the house. If the inhabitant is cleaning house, we would expect many zone changes in a short time; if the inhabitant is reading quietly in a corner, we would expect few zone changes; if the inhabitant is getting ready for work in the morning, we might expect an occupancy pattern that alternates between the bedroom and bathroom. An *occupancy model* and *anticipator* provide information about these occupancy patterns to the state estimator. We discuss the occupancy model and the anticipator in a following section.

The second key bit of information useful for decision making is the level of natural light in the zone. This is a tricky problem, as light sensors in a zone measure the ambient light level which depends on the current state of the lighting devices, the outside light level, and whether shades are open or closed. The *natural light estimator* attempts to determine the level of natural light in the zone if the lighting devices were turned off.

Although the state representation in ACHE attempts to recover some important information about the environment, the available sensor data does not contain all information relevant to the control task. For example, ACHE cannot determine the exact location of the inhabitants, their state of dark adaptation, or their intentions at the moment. Consequently, the state provided to the controller is non-Markovian (meaning that the instantaneous state is incomplete), and Q learning is not guaranteed to converge on an optimal policy.

12.5.1 Q-Learning Controller

As we indicated earlier, each zone is treated as an independent control task and has a separate Q controller. In addition, the control task for each *device* in each zone is treated as independent of the others. Although the optimal setting of one device in a zone certainly depends on the settings of others, the independent-controller approach still appears to capture these dependencies (Markey & Mozer, 1992).

The Q controller for a particular zone and a particular device in a zone is implemented as a pair of look-up tables—one for when the zone is occupied, one for when the zone is empty—that map a state and a decision to an expected cost. The occupied table takes as its state:

- natural light level (5 bins)
- number of zone changes by inhabitants in the last minute (0–1, 2–5, 6+)

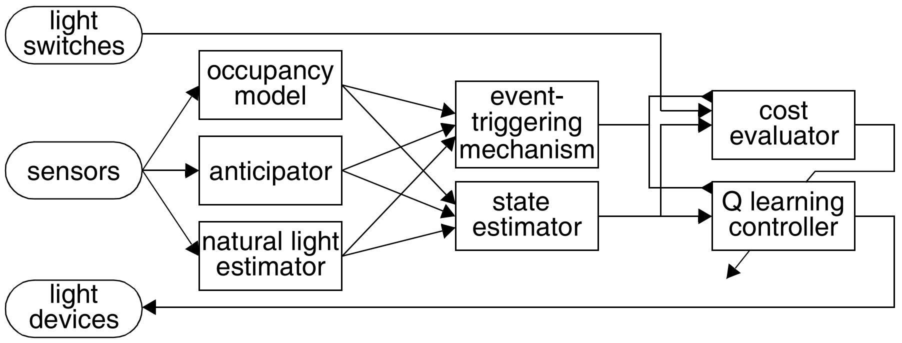

FIGURE 4. The ACHE architecture.

- number of zone changes by inhabitants in the last five minutes (0–1, 2–5, 6+)
- if zone is great room, location in room (south, north, or moving)

and allows five control decisions: intensity setting 0 (device off), 6, 9, 12, or 15 (device fully on). We might have allowed sixteen decisions, corresponding to each of the sixteen intensity settings our devices support, but subjectively many of these settings are indistinguishable. The empty table takes as its state:

- number of entries to zone under consideration in the last five minutes (0–1, 2+)
- number of entries to zone under consideration in the last 20 minutes (0–1, 2+)
- relative power consumption of device in its current state (5 bins)

and allows for two control decisions: leave the device at its current setting, or turn off the device.

12.5.2 Occupancy Model

The occupancy model determines which zones in the house are currently occupied based on motion detector signals and a finite-state model of the house. When motion is sensed in a currently unoccupied zone, the zone is flagged as occupied. It remains so until motion is sensed in a physically adjacent zone and no additional motion signals are received in the zone for at least κ seconds. The occupancy model also uses opening and closing of the front and back doors, in conjunction with motion signals, to determine when an occupant enters or exits the house.

The value for κ depends on whether a single or multiple occupants are in the home. It is necessary to be conservative in declaring a zone to be empty if the home contains multiple occupants, for the following reason. If zone 1 is occupied, and motion is sensed in adjacent zone 2, it could be that multiple inhabitants have moved from zone 1 to zone 2 and zone 1 is now empty, or it could be that a single inhabitant has moved from 1 to 2, and zone 1 is still occupied by another inhabitant who is stationary at the time. Thus, κ is set to the conservative value of 600 seconds when the home contains multiple occupants, but only 10 seconds if the home contains a single occupant. The single/multiple status is also determined by the occupancy model: When multiple zones are occupied for longer than ten seconds, the “multiple occupant” status is set, and remains set until the home becomes empty.

12.5.3 Anticipator

The occupancy model tags a zone as occupied when motion in that zone is first sensed. This is inadequate for lighting control, however, due to the fact that the motion detectors are sometimes sluggish—the inhabitant can walk half way across a room before they fire—and once motion has been detected, the time to transmit commands to the lights is about 700 msec. Consequently, one would really like ACHE to predict an impending zone occupancy and to issue a lighting command shortly *before* the zone becomes occupied. A neural network, called the *anticipator*, is used for this purpose; it predicts which zone or zones will become occupied in the next two seconds.

One might argue that the anticipator is needed only because of limitations of the hardware in the Adaptive Home, and could be avoided with more expensive state-of-the-art sensors and actuators. We believe, however, that noisy and undependable sensors pose a constant challenge in real-world control, and prediction provides a means of increasing reliability in the face of noise.

The anticipator takes the following data as input:

- average value of the binary motion detector signal in a 1-, 3-, and 6-second window (36 inputs)
- instantaneous and 2-second average of the binary door status (20 inputs)
- instantaneous, 1-second, and 3-second averages of sound level (33 inputs)
- current zone occupancy status and durations (16 inputs)
- time of day (2 inputs, in a circular 24 hour clock-based representation)

The purpose of including multiple time averages of sensor values is to encode information about the recent temporal history in a static representation. Although one could use a tapped-delay line to serve this purpose, the time-averaged values allow for a more compact and explicit representation of critical information. For example, one can determine that a door just opened by checking that the instantaneous door status is 1 (open), and the two-second average is less than 1; and one can determine that motion just ceased if the three-second average is larger than the one-second average.

The output of the anticipator is interpreted as the probability, for each of the eight zones, that the zone will become occupied in the next two seconds, given that it is currently unoccupied. The output of the anticipator is ignored if the zone is currently occupied. The anticipator runs every 250 msec. It is a standard single-hidden-layer neural network with 107 inputs, 50 hidden units, 8 output units, direct input-output connections, and a symmetric sigmoidal activation function. The number of inputs is determined by the available data and the chosen state representation; the number of outputs is determined by the number of zones. The neural network architecture is fairly generic: including direct input-output connections allows for the linear structure of the domain to be easily captured. The number of hidden units used is generally crucial to the performance of a network (this is the problem of *model selection*), but because of the availability of large amounts of training data, the large number of free parameters in the network is justified and had less impact. (For further advice on building neural network models in practice, see Orr and Müller, 1998.)

The occupancy model provides the training signal to the anticipator. The anticipator's job is to detect cues in the environment that reliably predict the zone entry announced by the occupancy model. The training procedure is inductive: a partially trained anticipator net is run to make predictions, and when it produces an error, new data is added to the training set. When a sufficient quantity of new data is added (200 examples), the network is retrained. The anticipator can produce two types of errors: a *miss*, when a zone entry fails to be predicted within two seconds of the event, and a *false alarm*, when a zone entry is predicted and none in fact occurs. To avoid a miss in the future, a new training example is generated in which the sequence of eight input states leading up to the zone entry—corresponding to the states at *t–2000* msec, *t–1750* msec, ..., *t–250* msec—are all associated with the zone entry. To avoid a false alarm, a training example is generated in which the state at the time the entry is predicted is associated with no zone entry. A temporal difference training procedure (Sutton, 1988) is used for the sequence of states leading to the miss. This is appropriate because in the sequence, each state becomes an increasingly better predictor of the event. Using the temporal difference procedure yielded slightly better results than standard supervised learning.

Figure 5 shows a measure of performance of the anticipator as a function of the amount of collected training data. The horizontal axis also corresponds to time on the scale of about a month. The performance measure is the ratio of the number of *hits* (correct predictions of a zone entry) to the sum of misses plus false alarms collected in a small time window. Although the curve is noisy, performance is clearly improving as additional data is collected. Because the anticipator outputs a continuous probability, it is necessary to threshold the output to produce a definite prediction that can be used by ACHE. Through empirical tests, we found that a threshold of 0.7 roughly balanced the miss and false alarm rates.

Anecdotally, the anticipator net does seem to have captured important behavioral regularities of the house inhabitant. We illustrate several examples using the house floor plan shown in Figure 6. In the evening, when the inhabitant walks out of the great room and into the entry (trajectory A), the anticipator predicts that the master bedroom is about to become occupied. However, when the same pattern of movement occurs in the morning, the anticipator predicts that the inhabitant is headed toward bedroom 2, which is used as an office. At night, when the inhabitant gets out of bed and walks toward the master bath (trajectory C), the anticipator uses the onset of motion in the master bedroom, along with a sound produced by a creaky floor, to predict that the bathroom is about to become occupied. This combination of cues is apparently necessary, as sound alone (e.g., telephone ringing) or motion alone (e.g., rolling over in bed) is insufficient to produce a strong prediction. The anticipator sometimes uses odd but reliable cues. For example, when the inhabitant showers in the morning, he has a habit of listening to a radio in the bathroom. Before walking out of the bathroom, he shuts off the radio. Consequently, a sudden offset of a sustained sound level in the bathroom is a reliable indicator that the bedroom is about to become occupied. On the whole, the anticipator is not entirely depend-

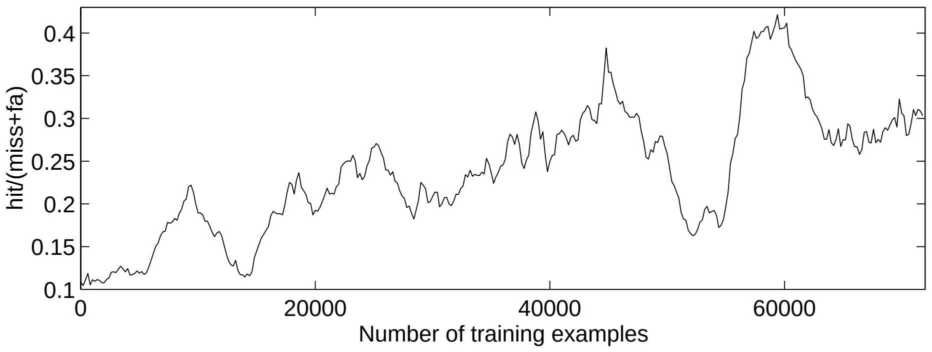

FIGURE 5. Performance of anticipator network as the number of training examples increases

able, though, primarily because the sparse representation of the environment produced by the sensors does not support perfect predictions.

12.5.4 ACHE Parameters and Costs

In the current implementation of ACHE, we use an exploration probability, θ, of 0.05, causing the controller to take the action believed to be optimal with probability 0.95, and to try another alternative at the remaining times. Rather than choosing the alternative arbitrarily from the set of actions, as one might do with undirected reinforcement learning, ACHE selected the action with next lower energy cost from the action believed to be optimal. With this choice, ACHE will gradually lower the device setting to minimize energy consumption, as long as the inhabitant does not express discomfort.

Determining the appropriate Q table learning rate, α, is tricky. The learning rate must be large enough that ACHE adapts after several experiences, but it must not be so large that a single experience causes ACHE to forget what had been learned in the past. We used a learning rate of 0.3, which provided a reasonable balance between adaptability and stability. The Q learning discount factor, λ, was zero, because event-based segmentation simplified reinforcement learning to a single-step problem with immediate payoff.

Finally, we list the various costs in the lighting control problem. We have already mentioned an energy cost, which was set to \$.072 per kilowatt-hour, the actual cost of electricity charged by the local utility company. The discomfort cost was set to \$.01 per device whose setting was manually adjusted by the inhabitant. Because this cost could be incurred for *each* device in a zone, and a zone has as many as seven devices, up to \$.07 could be charged for inhabitant discomfort each time a zone is entered, which is quite steep relative to energy costs. Two additional costs were incorporated, related to the anticipator. Consider the situation in which the inhabitant exits a zone, the lights in the zone are turned off, and when the inhabitant returns to the zone, the anticipator fails to predict the return. The resulting delay in turning the lights back on will cause inconvenience to the inhabitant, quantified as a cost of \$.01 per device which was turned off but should have been set to a non-zero intensity. This cost is incurred only when the anticipator misses the zone entry. The complementary situation is when the anticipator false alarms, i.e., incorrectly predicts a zone entry, and causes lights to be turned on in the zone. (The lights are turned back off after a time-out period, if the zone entry does not occur.) A cost should be incurred in this situation to reflect annoyance to the inhabitant who may notice lights turning on and off in unoccupied zones. We set this cost to \$.01 per device which was turned on as a result of an anticipator false alarm.

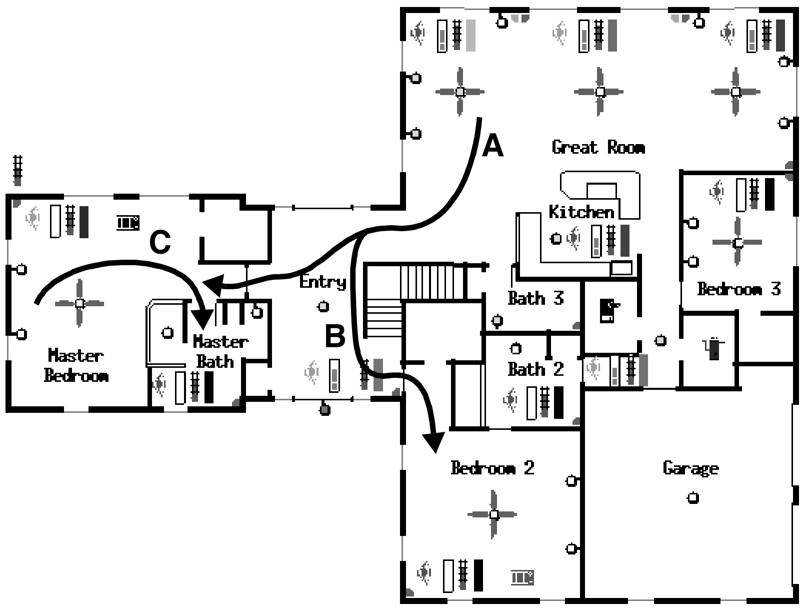

FIGURE 6. A floor plan of the adaptive house, including locations of sensors and actuators. Three trajectories are indicated by arrows labeled A, B, and C.

ACHE's Q table was initialized to the expected energy cost for the corresponding decision, which assumes that the inhabitant has no preference for the device setting. Consequently, devices will not be turned on unless the inhabitant expresses discomfort. Rather than training the anticipator and the controller simultaneously, the anticipator was trained first, over a period of a month, before the controller was turned on.

12.6 RESULTS AND DISCUSSION

Figure 7 shows energy and discomfort costs as a function of the number of events experienced by ACHE. The discomfort cost includes the anticipator miss and false alarm costs. The Figure reflects twenty-four days of data collection, and events were logged only during times when it was likely that indoor lighting would be required, from 19:00 to 06:59. To smooth out some of the noise, data points in the Figure reflect the mean value in a moving window of fifty events centered on the current event. Although the energy cost decreases fairly steadily, the discomfort costs are quite variable. This is due, at least in part, to a programming error in ACHE that caused the misattribution of some costs. Because time limitations did not allow us to restart ACHE, we corrected the problem near event 1700 and continued ACHE's training. From this point on, ACHE appears to have converged quickly on a low-energy cost, low-discomfort cost solution.

12.6.1 A Training Scenario

ACHE learns quite rapidly, as we illustrate in the following training scenario. To simplify the scenario, assume that the state which serves as input to the Q-learning controller is fixed. The first time that the inhabitant enters a zone (we'll refer to this as a *trial)*, ACHE decides, based on the initialization Q values, to leave the light off. If the inhabitant overrides this decision by turning on the light, ACHE immediately learns that leaving the light off will incur a higher cost (the discomfort cost) than turning on the light to some intensity (the energy cost). On the next trial, ACHE decides to turn on the light, but has no reason to believe that one intensity setting will be preferred over another. Consequently, the lowest intensity setting is selected. On any trial in which the inhabitant adjusts the light intensity upward, the decision chosen by ACHE will incur a discomfort cost, and on the following trial, a higher intensity will be selected. Training thus requires just three or four trials, and explores the space of decisions to find the lowest acceptable intensity. ACHE also attempts to conserve energy by occasionally "testing" the inhabitant, selecting an intensity setting lower than the setting believed to be optimal. If the inhabitant does not complain, the cost of the decision is updated to reflect this fact, and eventually the lower setting will be evaluated as optimal.

This scenario plays out nicely, at least in part because we assumed that the state serving as input to the Q-learning controller was fixed. In practice, changes in the state lead to complications. For example, one component of the state representation is the number of times the inhabitant has recently moved from one zone to another. As this number

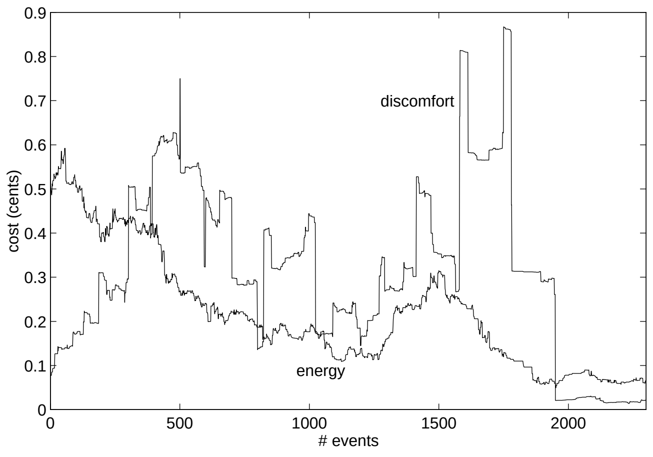

FIGURE 7. Energy and discomfort costs incurred by ACHE as a function of the number of events experienced. The discomfort cost includes anticipator miss and false alarm costs.

increases, the state changes, the Q look-up table switches to a different bin for decision making, and the experience gained in the previous bin is no longer accessible. From the inhabitant's perspective, ACHE appears to forget its recent training. In the long run, this is not a problem, as eventually ACHE acquires sufficient experience in all states. One way of avoiding the inconvenience in the short run is to use a memory-based approach, such as *k*-nearest neighbor, to implement the Q function, rather than a look-up table.

12.6.2 Informal evaluation

Beyond the performance curve, it is difficult to evaluate ACHE formally. One would really like to know whether ACHE is useful and whether people would want such a system in their homes. Although ACHE appears surprisingly intelligent at times, it can also frustrate. Most of the frustration, however, was due to the implementation using X10 controls, which are painfully slow to respond, especially when ACHE is adjusting multiple banks of lights at once. I also found it disconcerting when ACHE would incorrectly predict my passage into another room and lights would turn on or off in an unoccupied area of the house. This problem can be rectified by increasing the cost of false anticipation errors.

Overall, I found the benefits to outweigh the inconveniences, rapidly becoming accustomed to the automatic control of lighting. Indeed, when ACHE is disabled, the home seems cold and uninviting. Although this may sound implausible to one who hasn't lived in an automated home, reverting to manual control of the lights is annoying and cumbersome. An informal but compelling testament to the value of ACHE is that I left it running long beyond the termination of data collection.

12.7 THE FUTURE

In this final section, I discuss possible extensions of ACHE.

12.7.1 Extensions Not Worth Pursuing

The Adaptive House project has inspired much brainstorming about ways to extend the project further, most of which seem entirely misguided. One idea often mentioned is controlling home entertainment systems—stereos, TVs, radios, etc. The problem with selection of video and audio in the home is that the inhabitants' preferences will depend on state of mind, and few cues are directly available from the environment—even using machine vision—that correlate with state of mind. The result is likely to be that the system mispredicts often and annoys the inhabitants more than it supports them. The annoyance is magnified by the fact that when inhabitants seek audio or video entertainment, they generally have an explicit intention to do so. This intention contrasts with, say, temperature regulation in a home, where the inhabitants do not consciously consider the temperature unless it becomes uncomfortable. If inhabitants are aware of their goals, achieving the goal is possible with a simple click of a button and errors—such as blasting the stereo when one is concentrating on a difficult problem—are all but eliminated. The benefit/cost trade off falls on the side of manual control.

ACHE could always be improved by information provided explicitly by the inhabitants. For example, providing ACHE with the inhabitants' calendar entries in their PDAs would greatly simplify predictions concerning departure and return times; providing ACHE with information about current activities (e.g., hosting a party, preparing dinner, relaxing, etc.) would certainly simplify the lighting regulation problem and would eliminate errors the system might make. However, the premise of the project—as I stated at the outset—was to see how intelligent a system could be with a minimal user interface. Any additional information provided to ACHE by the inhabitants would require an additional user interface. It is clear that more information would make for a smarter system, but if a system passes the threshold of usability without requiring this information, it is a strong proof of concept.

12.7.2 Activity Classification

ACHE'S control decisions are based on the current state of the environment and predictions concerning future states. For example, the motion and sound cues lead to a prediction of future location, which might lead to a decision to turn on a light. ACHE, however, has no explicit representation of inhabitant activities, e.g., whether the inhabitants are preparing dinner, relaxing, studying, etc. Such an intermediate level of representation could serve a valuable purpose, particularly in integrating sensor information over time and increasing the coherence of control decisions over time. For example, inferring that inhabitants are preparing dinner would allow ACHE to maintain a fixed pattern of lighting, even if the moment-to-moment behavior of the inhabitants was difficult to interpret. In a home, the set of activities sufficient to characterize the typical and frequent behaviors of the inhabitants is finite and readily characterized; thus we pose the problem as activity *classification*—choosing one ore more activities from the finite set based on sensor data.

Machine learning researchers have addressed the problem of inferring activities of individuals (Brand & Kettnaker, 2000; Brand, Oliver, & Pentland, 1997; Ivanov & Bobick, 2000; Oliver, Rosario, & Pentland, 2000), although the work has been primarily directed at recognizing actions involving multiple interacting individuals using machine vision and hidden Markov models. In the case of home environments, other sources of sensor data are available, and a primary research challenge is handling interleaved activities (e.g., the inhabitant interrupts dinner preparation to answer the phone or put the clothes in the dryer). Recent work of Girolami and Kaban (2003) does address the problems of multiple activities.

12.7.3 Educating the Inhabitants

In the course of its operation, ACHE constructs models of the inhabitants and the environment. One interesting lesson from the Adaptive House is that as the inhabitant, I also constructed a model of ACHE, or more specifically, a model of ACHE's model of the inhabitant. For instance, if I were at work at 8 p.m., I would realize that under ordinary circumstances, I might have left several hours earlier; consequently, ACHE would be expecting me, and I felt compelled to return home. To some degree, I regularized my schedule in order to accommodate ACHE and its actions. Living with ACHE makes one aware of one's own occupancy and movement patterns. It is not entirely facetious to claim that ACHE trains the inhabitant, just as the inhabitant trains ACHE. Indeed, this interactive training is one of the virtues of living with an adaptive house. To the extent that the house discovers regularities of the inhabitants' behavior, and inhabitants regularize their behavior to accommodate the house, the interaction converges on an ideal situation—inhabitants whose schedules and behavior are predictable, allowing ACHE to both maximize comfort and minimize energy utilization.

Generalizing from this scenario, it seems useful for a smart home to educate its inhabitants concerning their behavior and needs. This principle was fortuitously revealed via a hardware bug in the Adaptive House. The waterflow sensor sometimes malfunctioned and needed to be reset. Rather than checking the sensor daily, we used a heuristic to warn us when a reset was necessary: if the bathroom was occupied for more than 5 minutes and hot water flow was not detected, a warning message was broadcast throughout the house. This would invariably catch occasions when the inhabitants were showering but the sensor had failed. Additionally, it would also catch occasions when the inhabitants dawdled in the bathroom, e.g., reading on the toilet. Long after the hardware problem was resolved, we left the broadcast message in the system, because it provided useful feedback to the inhabitants about how their time was being spent. Because time is such a precious commodity, a useful function of an intelligent home would be to help inhabitants use their time more efficiently.

Another scenario in which the Adaptive Home could educate its inhabitants concerns energy use. Whenever the inhabitant overrides current setpoints, there are consequences for the expected cost. For example, raising the target temperature from 68 to 70 degrees will cost more in energy to heat the home. The exact cost depends on the occupancy patterns in the home, the weather, and thermal properties of the home and heating system. Because ACHE models all of these elements, forecasting the additional cost due to a change in setpoint is straightforward. Essentially, ACHE can say, "If you really want the temperature another 2 degrees higher, you should expect to pay another $30 in the coming month."

I described three scenarios in which a smart home could provide feedback to its inhabitants, allowing them to make more informed decisions or better use of their time. Such feedback is not necessarily incompatible with the principle I argued for at the outset of the chapter—the principle that novel user interfaces should be avoided. In the scenarios described here, the feedback provided is expressed in terms individuals can comprehend, and supports informed decision making. Perhaps the next step in intelligent environments will come from environments that stick their virtual nose in our business.

12.8 REFERENCES

Bellman, R. (1957). *Dynamic programming*. Princeton, NJ: Princeton University Press.

Brand, M., & Kettnaker, V. (2000). Discovery and segmentation of activities in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 844–851.

Cohen, M. (2003). The internet chair. *International Journal of Human-Computer Interaction*, *15*, 297–311.

Davis, J. W., & Bobick, A. F. (1996). The representation and recognition of action using temporal templates. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'97)*, San Juan, Puerto Rico.

Girolami, M, & Kaban, A. (2003). Sequential activity profiling: Latent Dirichlet allocation of Markov models. To appear in *Neural Information Processing Systems XVII.*

Ivanov, Y.A. & Bobick, A.F. (2000). Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22,* 852–872.

Markey, K., & Mozer, M. C. Comparison of reinforcement algorithms on learning discrete functions: Learnability, time complexity, and scaling. *Proceedings of the International Joint Conference on Neural Networks* (Volume I, pp. 853–859). San Diego, CA: IEEE Publishing Services, 1992.

Oliver, N. M., Rosario, B., & Pentland, A. P. (2000). A Bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22,* 831–843.

Orr, G. B., & Müller, K.-R. (1998). *Neural networks: Tricks of the trade*. Berlin: Springer.

Sutton, R. S. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Watkins, C.J.C.H. (1992). Learning from Delayed Rewards. Unpublished Doctoral Dissertation. King's College, Cambridge, UK.

Watkins, C.J.C.H., & Dayan, P. (1992). Q learning. *Machine Learning*, *8*, 279–292.