Appears in: C. L. Giles & M. Gori (Eds.), *Adaptive processing of temporal sequences and data structures* (pp. 370–388). Berlin: Springer Verlag. 1998.

Parsing the Stream of Time: The Value of Event-Based Segmentation in a Complex Real-World Control Problem

Michael C. Mozer and Debra Miller

Department of Computer Science
University of Colorado
Boulder, CO 80309–0430

Introduction

Temporal information-processing tasks generally require that the continuous stream of time be *parsed* or *segmented*, which involves determining boundaries that divide the stream into distinct intervals. In a *clock-based segmentation*, the boundaries are spaced equally in time, resulting in fixed-duration intervals. In an *event-based segmentation*, the boundaries depend on the state of the environment, resulting in variable-duration intervals. Models of temporal information processing in cognitive science and artificial intelligence generally rely on clock-based segmentation. However, event-based segmentation can greatly simplify temporal information-processing tasks. We illustrate by describing a complex control problem that appears intractable when cast in terms of a clock-based segmentation, but has a straightforward solution when cast in terms of an event-based segmentation.

Parsing Time

Figure 1 shows intuitive examples of two temporal patterns parsed both by clock- and event-based segmentations. The clock-based segmentation of a time series (panel A) produces intervals that are independent of the nature of the series, whereas the event-based segmentation of the series (panel B) produces intervals that are delimited by zero crossings. The clock-based segmentation of a musical excerpt (panel C) is represented by notes of equal duration, whereas the event-based segmentation of the same excerpt (panel D) is represented by one note per musical event. In a clock-based segmentation, it is natural for the end boundary of one interval to be the start boundary for

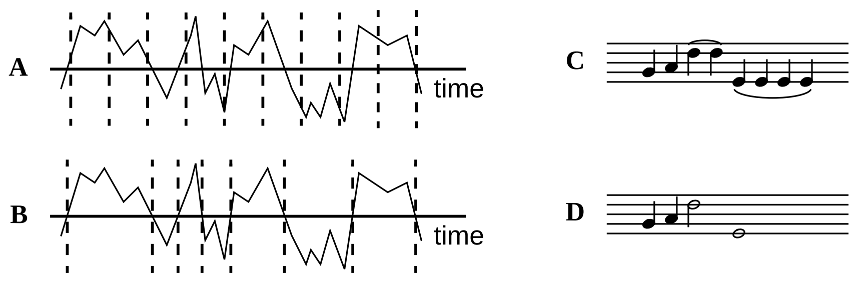


Fig. 1. A) A time series parsed according to a clock-based segmentation; B) the time series with an event-based segmentation; C) a musical excerpt parsed according to a clock-based segmentation; D) the musical excerpt with an event-based segmentation.

the next, leading to nonoverlapping, contiguous intervals. However, in an event-based segmentation, where the start and end boundaries may be distinct and dependent on different features of the temporal pattern, the resulting intervals may be overlapping and noncontiguous. Another consequence of event-based segmentations is that the resulting representations will often include an explicit encoding of the duration of an interval, e.g., the musical notation for half notes versus quarter notes; in contrast, a clock-based segmentation encodes this information implicitly.

Our distinction between clock- and event-based segmentation is not original. Although clock-based segmentation is more common and is often the more intuitive approach, event-based segmentation has appeared in many information processing models. In the following three sections, we describe models from the domains of biology, control engineering, and neural networks that utilize event-based segmentation. Our goal is to highlight the distinction between clock- and event-based segmentation and to point out some of the virtues of event-based segmentation, so that the reader will become more cognizant of the space of alternatives that can be used for temporal information-processing tasks.

Event-Based Segmentation in Biological Systems

Biological organisms are bombarded with stimulation from the environment and do not have the capacity to process all information in real time. Consequently, organisms require some type of *orienting mechanism* to alert the organism when an unusual or interesting stimulus appears, allowing the organism to process and respond to the stimulus. One can conceive of the orienting mechanism acting as a gate on the flow of information from the sensory organs to higher cognitive processes. The orienting mechanism does not need to know how to respond to the stimulus, but only that the stimulus is salient and must be dealt with. The latter problem may be significantly easier than the former. The orienting mechanism achieves a form of event-based segmentation. Its detection of a salient event in the temporal stream triggers information processing of the event.

The psychological reality of event-based segmentation can be illustrated through a familiar phenomenon. Consider the experience of traveling from one location to another, such as from home to office. If the route is unfamiliar, as when one first starts a new job, the trip is confusing and lengthy, but as one gains more experience following the route, one has the sense that the trip becomes shorter. One explanation for this phenomenon is as follows. On an unfamiliar route, the orienting mechanism constantly detects novel events, and a large number of such events will accumulate over the course of the trip. In contrast, few novel events occur on a familiar route. If our perception of time is event based, meaning that higher centers of cognition count the number of events occurring in a temporal window, not the number of milliseconds, then one will have the sense that a familiar trip is shorter than an unfamiliar trip. Experimental evidence clearly supports the notion that the event stream influences the perception of duration, although the relationship between novelty and the perception of duration is complicated due to interacting variables such as sequence complexity and attention (e.g., [1] [4] [7] [8] [20]).

Event-Based Segmentation in Hybrid Systems

In the engineering disciplines of automation, manufacturing, and robotics, event-based segmentation has been a key idea for real-world monitoring and control tasks [17] [18]. Figure 2 sketches what these disciplines call a *hybrid system*, consisting of a continuous-time and continuous-state plant (e.g., representing the state of robot and its environment over time), and an abstract model of the plant, called a *discrete event dynamic system* (*DEDS*), which might be a finite-state automaton, a Petri net, a Markov model, or a queueing model. The state-time trajectory in Figure 2A depicts the continuous dynamics of the plant. The state space is quantized into discrete regions, and an event is said to occur when the trajectory crosses from one region into another, denoted by circles on the state trajectory. Events trigger state transitions in the DEDS. Thus, the DEDS is presented with a sequence of events determined by salient changes in the environment, not by ticks of a clock—the key aspect of event-based segmentation. Perhaps a more familiar example of a hybrid system to most readers is a hidden Markov model used for speech recognition.

In a hybrid system, machine learning techniques can be used to adapt several distinct components of the system, including event identification [11] [16], DEDS identification [9] [10], and policy learning based on the DEDS model [2] [3].

Event-Based Segmentation in Neural Network Models

Although neural network models of temporal pattern analysis commonly rely on clock-based representations, some incorporate event-based segmentation. Figure 3 depicts three such models, which we briefly describe.

Figure 3A sketches a three-stage approach to speech recognition [14], motivated by the demands of a simple hardware implementation. The segmentation stage demarcates word boundaries, the normalization stage adjusts the word's duration, and the recognition stage processes the resulting fixed-size pattern. Surprisingly, in architectures such as this with distinct segmentation and recognition stages, speech recognition researchers anecdotally note that performance is far more dependent on the robustness of the segmentation stage than on the recognition stage.

Figure 3B shows the *long short-term memory* architecture [5]. Each circle depicts a connectionist unit that responds to the input stream. Detection of an event by the left-

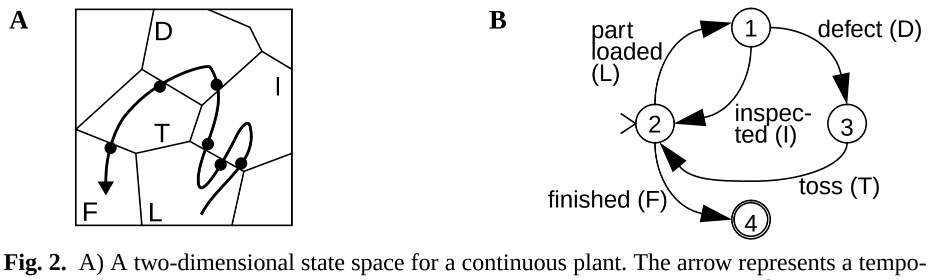


Fig. 2. A) A two-dimensional state space for a continuous plant. The arrow represents a temporal trajectory through state space. The labeled regions indicate a quantization of state space, and the circles on the trajectory indicate a transition from one region to another. B) A discrete-event dynamic system model of the plant.

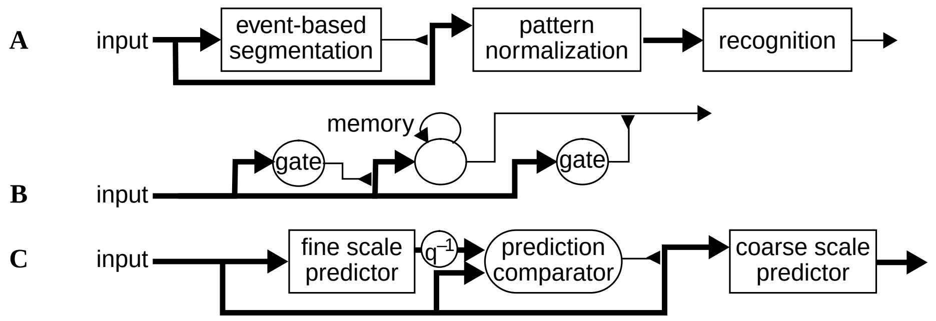
A
B
C


Fig. 3. Three neural network models that incorporate event-based segmentation: A) speech recognition model [14]; B) long short-term memory [5]; C) hierarchical temporal prediction [15]. The thick lines denote transmission of vector values, the thin lines scalars. The triangle junctions denote a multiplicative or gating connection.

most unit allows some transformation of the event to be stored in a memory cell that latches the value via a linear recurrent connection. The rightmost unit determines the conditions under which the latched value will become available to the rest of the system. Through event-based segmentation of the input stream, this architecture is able to learn long-term temporal dependencies that cannot be discovered by standard architectures and training algorithms.

Figure 3C shows a hierarchical temporal prediction architecture [15]. The *fine-scale predictor* is a neural net that processes an input sequence, at each time predicting the next element in the sequence. The *prediction comparator* detects surprising events in the input stream by examining the discrepancy between the prediction and reality. Surprising events are passed to a *coarse-scale predictor*, which attempts to discover structure that the fine-scale predictor did not learn. Thus, a hierarchical analysis of the input stream is achieved via event-based segmentation.

In each model, clock-based time is eliminated from the input representation, and the resulting event-based representation greatly reduces the complexity of learning.

A Real-World Control Problem

We now change topics and describe a problem that we have studied in the area of adaptive control and home automation. The problem appears intractable when cast in terms of a clock-based segmentation, but has an extremely simple solution when cast in terms of an event-based segmentation. We begin by describing the problem, explain why a clock-based segmentation of time leads to insurmountable practical difficulties in solving the problem, and then describe an approach using event-based segmentation.

Home Automation

The home automation industry has promised homes that can be programmed to perform such tasks as closing the drapes at night, turning down the stereo volume when the phone rings, flashing the bathroom lights as a reminder to the inhabitants to take their medication, and drawing a bath at a certain temperature at a certain time of day. Despite these promises, few homes—even in new construction—are automated. One reason that the industry has yet to reach its potential is that residents are reluctant to program their VCRs, let alone their homes. A programmable house seems like a tremendous burden, not a great boon. In response, several of the more popular home automation systems are set up to allow a service technician to regularly reprogram the home for the residents as their needs change.

Rather than having a home that can be programmed to perform various functions, one would really like a home that *programmed itself* to accommodate the schedules and lifestyles of the inhabitants. This is the goal of the *Adaptive House* project [13]. The Adaptive House is a residence 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. The project involves several subprojects, each with its own challenges and peculiarities, including the regulation of indoor lighting, temperature, and hot water. In the following section, we describe the lighting control problem.

Lighting Control

We call the control system in the Adaptive House *ACHE*, an acronym for Adaptive Control of Home Environments. ACHE monitors the state of the environment and senses actions performed by the inhabitants; these actions include turning lights on and off and setting their intensities. ACHE can then learn the inhabitants' lighting preferences, and can automatically adjust the lights according to these preferences, freeing the inhabitants from manual control. In addition, energy consumption influences control decisions, in a manner that we describe below. More important than turning lights on and off, ACHE sets 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. Figure 4 summarizes the framework in which ACHE operates.

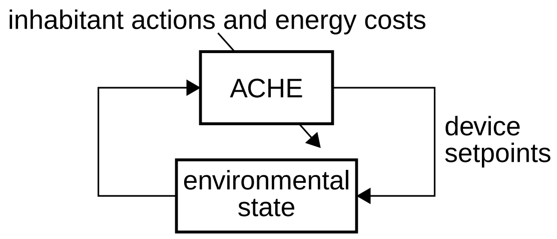


Fig. 4. ACHE specifies device intensity setting (brightness), which affects the state of the environment, which in turn serves as input to ACHE. The training signals to ACHE are the actions taken by the inhabitants and energy costs.

What Makes Lighting Control Difficult?

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

- The Adaptive House has twenty-two independently controlled banks of lights, each of which has sixteen intensity settings. The great room (consisting of the living room, dining room, and kitchen) alone has seven banks of lights.
- 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 protocol for sending commands to the lights. This lag is long enough to inconvenience the inhabitant.
- 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 implication 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.

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

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

where $d(\boldsymbol{u}_t)$ is the discomfort cost associated with the control decision $\boldsymbol{u}$ at t, and $e(\boldsymbol{x}_t)$ is the energy cost associated with the environmental state $\boldsymbol{x}$ 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 will take advantage of this observation below.

Reinforcement learning

Although dynamic programming 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* [21] [22]. Q learning provides an incremental update algorithm for determining the minimum expected discounted cost given that action $\boldsymbol{u}$ is taken in state $\boldsymbol{x}$:

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

where α is the learning rate, λ is the discount factor, and c_t is the immediate cost incurred at t. 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 fully observable state and infinite exploration, Q learning is guaranteed to converge on an optimal policy.

Temporal Credit Assignment and the Issue of Time Scale

Figure 5 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. This is analogous to the problem faced by back propagation in training recurrent networks, as described in several chapters of this volume.

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,

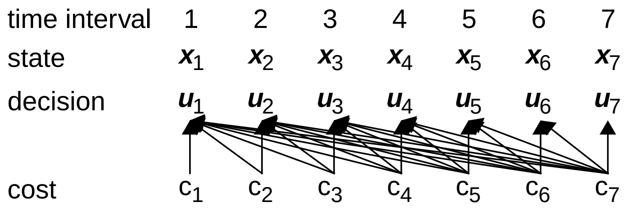


Fig. 5. The temporal credit assignment problem

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.

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.

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.

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.

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.

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.

ACHE Architecture

Figure 6 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 *orienting 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 orienting 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

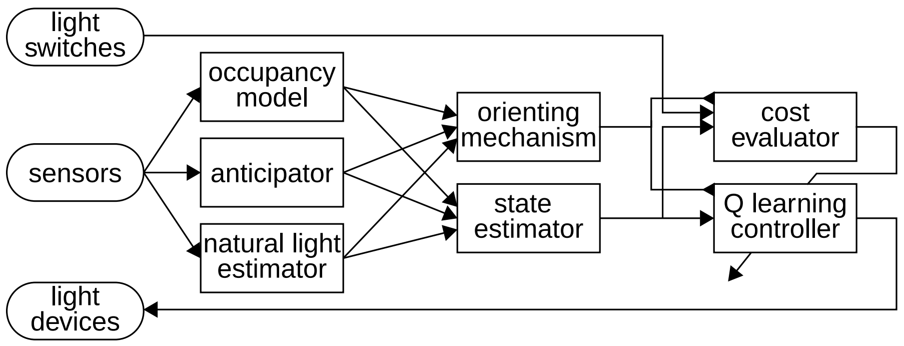


Fig. 6. The ACHE architecture.

changes; if the inhabitant 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, and Q learning is not guaranteed to converge on an optimal policy.

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 each another. 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 [12].

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+)
- 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.

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.

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 became 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 large number of free parameters in the network is justified by the availability of large amounts of training data.

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 [19] 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 7 shows a measure of performance of the anticipator as a function of the amount of training data collected. The horizontal axis also corresponds to time on the

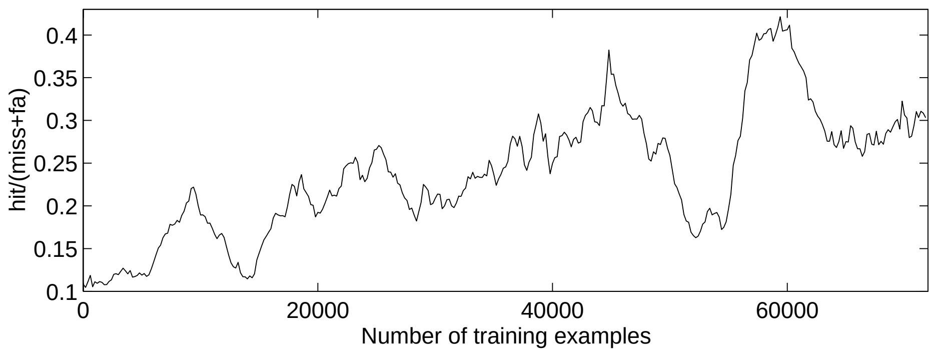


Fig. 7. Performance of anticipator network as the number of training examples increases

scale of about a month. The performance measure is the ratio of the number of *hits* (correct predictions of a zone entry) to 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 8. 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

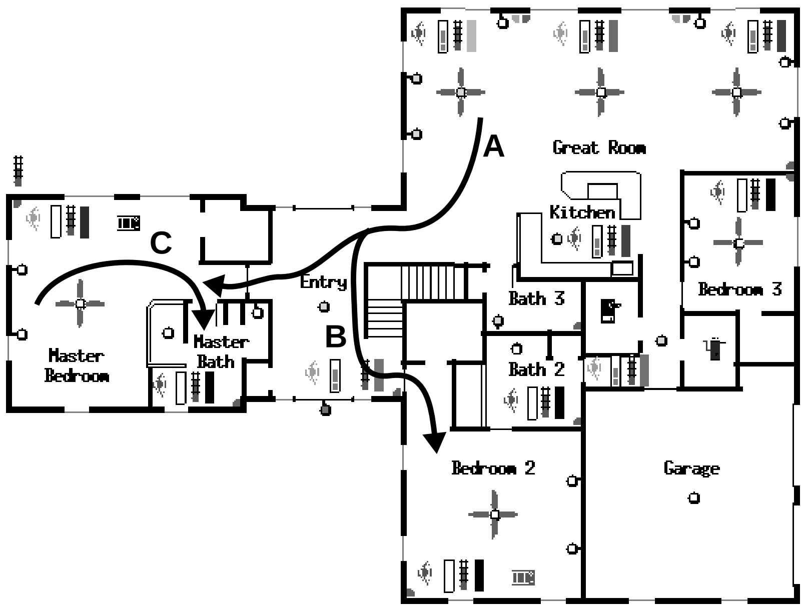


Fig. 8. A floor plan of the adaptive house, including locations of sensors and actuators. Three trajectories are indicated by arrows labeled A, B, and C.

anticipator is not entirely dependable, though, primarily because the sparse representation of the environment produced by the sensors does not support perfect predictions.

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.

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.

Results and Discussion

Figure 9 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.

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. Overall, the inhabitant found the benefits to outweigh the inconveniences, rapidly becoming accustomed to lights being controlled automatically. Indeed, when ACHE is disabled, the home seems cold and uninviting. Although this may seem implausible to one who hasn't lived in an automated home, reverting to manual control of the lights is annoying and cumbersome.

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

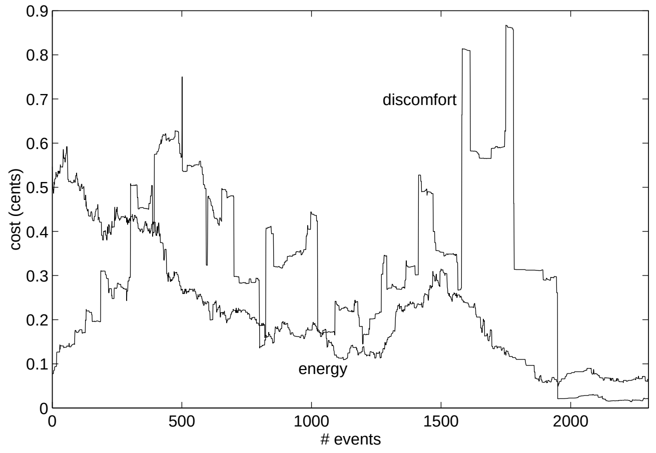


Fig. 9. Energy and discomfort costs incurred by ACHE as a function of the number of events experienced. The discomfort cost includes anticipator miss and false alarm costs.

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 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.

The Role of Event-Based Segmentation

The lighting control problem seems intractable when cast in the conventional framework of a clock-based segmentation of time. However, when cast in the framework of an event-based segmentation, we have shown that the solution is straightforward, almost trivial. However, event-based segmentation is not a panacea. In some conditions it will be superior to clock-based segmentation, in other conditions it will not. A characterization of these conditions is needed to advance the state of the art in temporal pattern processing. Although we cannot yet provide a formal characterization, we offer one step in this direction.

A task that involves responding appropriately and adaptively to a dynamic environment can, in principle, be decomposed into two subtasks: (1) orienting—deciding *when* an interesting, unexpected, or unusual event occurs in the environment; and (2) responding—deciding *what to do* in response to this event. In the lighting control domain we described in this chapter, as well as previous work we summarized that used event-based segmentation, *a solution to these two subtasks is simpler or more efficient than a solution to the overall task*. In particular, when salient events can readily be discriminated from the background, there is likely to be a win for this task decomposition. In the lighting control domain, orienting was straightforward based on an analysis of the domain. Even when a priori knowledge is insufficient to fully specify the orienting system, it is still possible that salient events are easily discriminated from the background, in which case a variety of learning procedures could be used to train an orienting system. The decomposition of the task into an orienting mechanism and a gated response system may impose a strong enough bias to constrain and simplify the learning task (e.g., [6]).

Acknowledgments

We are grateful to Marc Anderson and Robert Dodier, who developed the software infrastructure for the Adaptive House, and to Allen Zuverino for his assistance in testing and documenting the behavior or the lighting controller. We also thank Marco Gori and Lee Giles for organizing a stimulating, if not sweltering, workshop in Vietri, which served as the catalyst for completing this work. The Adaptive House project is supported by the Sensory Home Automation Research Project (SHARP) of Sensory Inc., as well as a CRCW grant-in-aid from the University of Colorado, NSF award IRI-9058450, and McDonnell-Pew award 97-18.

References

1. Block, R.A., George, E.J., & Reed, M.A. A watched pot sometimes boils: A study of duration experience. *Acta Psychologica*, *46*: 81–94, 1980.
2. Bradtke, S.J., & Duff, M.O. Reinforcement learning methods for continuous-time Markov decision problems. In G. Tesauro, D.S. Touretzky, & T.K. Leen (Eds.), *Advances in neural information processing systems 7*. Cambridge, MA: MIT Press, 1995.
3. Crites, R.H., & Barto, A.G. Improving elevator performance using reinforcement learning. In D.S. Touretzky, M.C. Mozer, & M.E. Hasselmo (Eds.), *Advances in neural information processing systems 8* (pp. 1017–1023). Cambridge, MA: MIT Press, 1996.
4. Hicks, R.E., & Allen, D.A. The repetition effect in judgments of temporal duration across minutes, days, and months. *American Journal of Psychology, 92*: 323–333, 1979.
5. Hochreiter, S., & Schmidhuber, J. H. LSTM can solve hard long time lag problems. In M.C. Mozer, M.I. Jordan, & T. Petsche (Eds.), *Advances in Neural Information Processing Systems 9* (pp. 473–479). Cambridge, MA: MIT Press, 1997.
6. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., & Hinton, G.E. Adaptive mixtures of local experts. *Neural Computation*, *3*: 79–87, 1991.
7. Kellaris, J.J., & Mantel, S.P. Shaping time perceptions with background music: The effect of congruity and arousal on estimates of ad durations. *Psychology and Marketing*, *13*: 501–515, 1996.
8. Kowal, K.H. Apparent duration and numerosity as a function of melodic familiarity. *Perception and Psychophysics, 42*: 122–131, 1987.

9. Kuniyoshi, Y., Inaba, M., & Inoue, H. Teaching by showing: Generating robot programs by visual observation of human performance. ISIR, 1989.

10. Kuniyoshi, Y., Inaba, M., & Inoue, H. Design and implementation of a system that generates assembly programs from visual recognition of human action sequences. IROS, 1990.

11. Lemmon, M., Stiver, J.A., & Antsaklis, P.J. Learning to coordinate control policies of hybrid systems. *Proceedings of the American Control Conference*, 1993.

12. 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.

13. Mozer, M.C., Dodier, R.H., Anderson, M., Vidmar, L., Cruickshank III, R.F., Miller, D. The neural network house: An overview. In L. Niklasson & M. Boden (Eds.), *Current trends in connectionism* (pp. 371-380). Hillsdale, NJ: Erlbaum, 1995.

14. Mozer, M.C. Neural network speech processing for toys and consumer electronics. *IEEE Expert*, *11*: 4–5, 1996.

15. Schmidhuber, J.H. Mozer, M.C., & Prelinger, D. Continuous history compression. In H. Huening, S. Neuhauser, M. Raus, & W. Ritschel (Eds.), *Proceedings of the International Workshop on Neural Networks, RWTH Aachen* (pp. 87–95). Augustinus, 1993.

16. Sikka, P., & McCarragher, B.J. Monitoring contact using clustering and discriminant functions. *Proceedings of the IEEE International Conference on Robotics and Automation,* 1996.

17. Sobh, T.M. Hybrid systems and control. In *Handbook on Industrial Electronics*. Boca Raton, FL: CRC Press, 1996.

18. Sobh, T.M., Owen, J.C., Valavanis, K.P., & Gracanin, D. A subject-indexed bibliography of discrete event dynamic systems. *IEEE Magazine on Robotics and Automation*, *1(2)*, 1994.

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

20. Thomas, E.A., & Brown, I. Time perception and the filled-duration illusion. *Perception and Psychophysics*, *16*:449–458, 1974.

21. Watkins, C.J.C.H. Learning from Delayed Rewards. Unpublished Doctoral Dissertation. King’s College, Cambridge, UK, 1992.

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