A Multi-agent Approach to Controlling a Smart Environment

Diane J. Cook, Michael Youngblood, and Sajal K. Das

Department of Computer Science Engineering,
The University of Texas at Arlington
{cook, youngbld, das}@cse.uta.edu
http://mavhome.uta.edu

Abstract. The goal of the MavHome (**M**anaging **A**n Intelligent **V**ersatile **Home**) project is to create a home that acts as a rational agent. The agent seeks to maximize inhabitant comfort and minimize operation cost. In order to achieve these goals, the agent must be able to predict the mobility patterns and device usages of the inhabitants. Because of the size of the problem, controlling a smart environment can be effectively approached as a multi-agent task. Individual agents can address a portion of the problem but must coordinate their actions to accomplish the overall goals of the system. In this chapter, we discuss the application of multi-agent systems to the challenge of controlling a smart environment and describe its implementation in the MavHome project.

1 Introduction

The **MavHome** smart home project is a multi-disciplinary research project at the University of Texas at Arlington focused on the creation of an intelligent and versatile home environment. We define an intelligent environment as one that is able to acquire and apply knowledge about its inhabitants and their surroundings in order to adapt to the inhabitants and meet the goals of comfort and efficiency. Our goal is to create a home that acts as a rational agent, perceiving the state of the home through sensors and acting upon the environment through effectors (in this case, device controllers). The agent acts in a way to maximize its goal, which is a function that maximizes comfort of its inhabitants and minimizes operation cost.

With these capabilities, the home can adaptively control many aspects of the environment such as climate, water, lighting, maintenance, and multimedia entertainment. Intelligent automation of these activities can reduce the amount of interaction required by inhabitants, reduce energy consumption and other potential wastages, and provide a mechanism for ensuring the health and safety of the environment occupants [1].

In order to achieve these goals, the house must be able to reason about and adapt to its inhabitants. In particular, a smart home agent must be able to accurately predict mobility and other activities of its inhabitants. Using these predictions the home can accurately route messages and can automate activities that would otherwise be manually performed by the inhabitants.

MavHome operations can be characterized by the following scenario. At 6:45 am, MavHome turns up the heat because it has learned that the home needs 15 minutes to warm to optimal temperature for waking. The alarm goes off at 7:00, which signals the bedroom light to go on as well as the coffee maker in the kitchen. Bob steps into the bathroom and turns on the light. MavHome records this interaction, displays the morning news on the bathroom video screen, and turns on the shower. While Bob is shaving MavHome senses that Bob has gained two pounds over the last week. MavHome informs Bob that this has been a trend over the last two months and offers suggestions for changing his lifestyle. When Bob finishes grooming, the bathroom light turns off while the blinds in the kitchen and living room open (an energy-efficient alternative to Bob's normal approach of turning on the living room, kitchen, and hallway lights). When Bob leaves for work, MavHome secures the home, lowers the temperature, starts the robot vaccuum, and turns on the lawn sprinklers despite the fact that the Internet forecast predicts a 40% chance of rain. MavHome tracks Bob's activities while he is away from home in order to inform him of problems at home and to have the house temperature and hot tub prepared for his return at 6:00.

A number of capabilities are required for this scenario to occur. For a house to be able to record inhabitant interaction and trigger sequences of events such as the bedroom light / coffee maker sequence, advances in active database techniques are needed. Machine learning techniques are required to predict inhabitant movement patterns and typical activities, and to use that information in automating house decisions and optimizing inhabitant comfort, security, and productivity. In order for MavHome to find him away from the home, mobile computing capabilities must be present.

As can be observed from the scenario, MavHome automates the control of numerous devices within the home. To scale to this size problem, the MavHome agent needs to be decomposed into lower-level agents responsible for subtasks within the home, including robot and sensor agents, and this organization should be dynamically composable. Finally, these capabilities must be organized into a multi-agent architecture that seamlessly connects these components while allowing improvement in any of the underlying technologies.

2 Related Research

As the need for automating these personal environments grows, so does the number of researchers investigating this topic. Some, like the MIT AIRE group [2] and the Stanford Interactive Workspaces Project [3], design conference rooms, kiosks, and offices with seamless integration between heterogeneous devices and multiple user applications in order to facilitate collaborate work environments. The Gaia project [4] adds operating system functionality to these spaces, so that the applications do not rely upon hardware or software configurations, and ultimately so that both physical and virtual devices can seamlessly interact.

Abowd and Mynatt's work focuses on ease of interaction with a smart space [5], and work such as the Gator Tech Smart House [6] customizes devices for elder care. The problem has become so recognized that NIST has identified integration of mobile components into smart spaces as a target area for standardization [7].

Mozer's Adaptive Home uses a neural network and reinforcement learning to control lighting, HVAC, and water temperature to reduce operating cost [8], although this may occur at the sacrifice of inhabitant comfort. In contrast, the approach taken by the iDorm project [9] is to use a fuzzy expert system to learn rules that replicate inhabitant interactions with devices, but will not necessarily find an alternative control strategy that improves upon manual control for considerations such as energy expenditure. The interest of industrial labs in smart home and networked appliance technologies is evidenced by the creation of Jini, Bluetooth, and SIP (Session Initiation Protocol) standards, and by supporting technologies such as Xerox PARC's Zombie Board, Microsoft's Easy Living project, the Cisco Internet Home, and the Verizon Connected Family project.

These projects have laid a foundation for our work. However, unlike related projects, we learn a decision policy to control an environment in a way that optimizes a variety of possible criteria, including minimizing manual interactions, improving operating efficiency, and ensuring inhabitant health and safety. We also ensure that our software need not be redesigned as new devices are registered, new spaces are tested, or new inhabitants move into the environment. To accomplish this goal, our intelligent environment must harness the features of multiple heterogeneous learning algorithms in order to identify repeatable behaviors, predict inhabitant activity, and learn a control strategy for a large, complex environment.

3 MavHome Architecture

The MavHome architecture is a hierarchy of rational agents which cooperate to meet the goals of the overall home. Figure 1 shows the architecture of a MavHome agent. The technologies within each agent are separated into four cooperating layers. The **Decision** layer selects actions for the agent to execute based on information supplied from other layers. The **Information** layer gathers, stores, and generates knowledge useful for decision making. The **Communication** layer includes software to format and route information between agents, between users and the house, and between the house and external resources. The **Physical** layer contains the basic hardware within the house including individual devices, transducers, and network hardware. Because the architecture is hierarchical, the Physical layer may actually represent another agent in the hierarchy.

Perception is a bottom-up process. Sensors monitor the environment (e.g., lawn moisture level) and, if necessary, transmit the information to another agent through the Communication layer. The database stores this information while other information components process the raw information into more useful knowledge (e.g., patterns, predictions). New information is presented to the Decision layer upon request or by prior arrangement. Action execution information

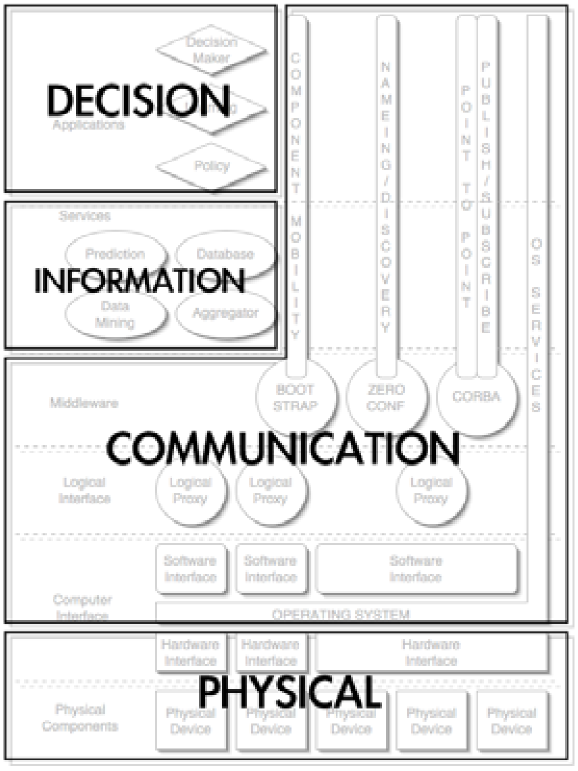

Fig. 1. MavHome abstract architecture

flows top down. The Decision layer selects an action (e.g., run the sprinklers) and relates the decision to the Information layer. After updating the database, the Communication layer routes the action to the appropriate effector to execute (e.g., powerline controller). If the effector is actually another agent, the agent receives the command through its effector as perceived information and must decide upon the best method of executing the desired action. A specialized interface agent provides interaction capabilities with users and with external resources such as the Internet. As shown in Figure 2, agents can communicate with parent/child agents or with other agents at the same level in the hierarchy.

The abstract layers of the MavHome architecture are realized through a set of concrete functional layers. These concrete layers are shown with some example components in Figure 3. The base layer is the *Physical Components* layer which consists of all real devices utilized in the system. These devices include powerline control interface hardware, touch screens, gesture input devices, cameras, and so forth, with the exception of the computer with which equipment is interfaced. The physical computer(s) and associated network this system resides on is considered the host of all layers above the physical. The *Computer Interface* layer contains the hardware interfaces to physical devices

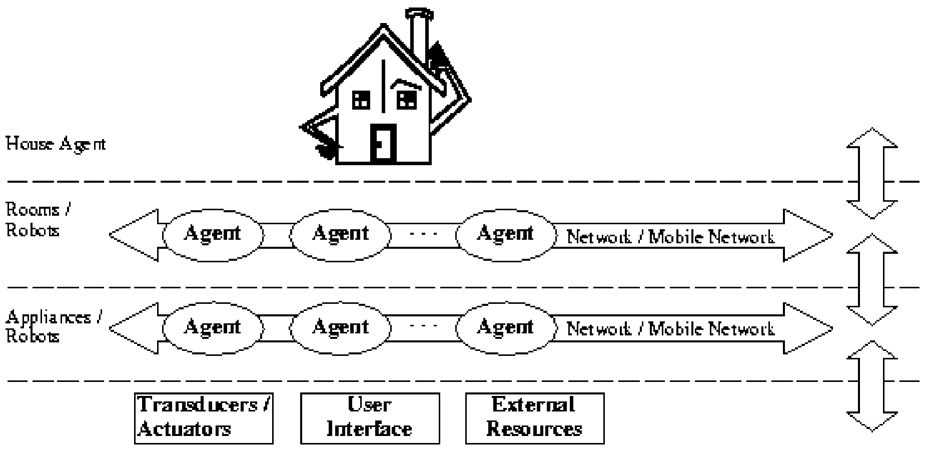

Fig. 2. MavHome multi-agent configuration

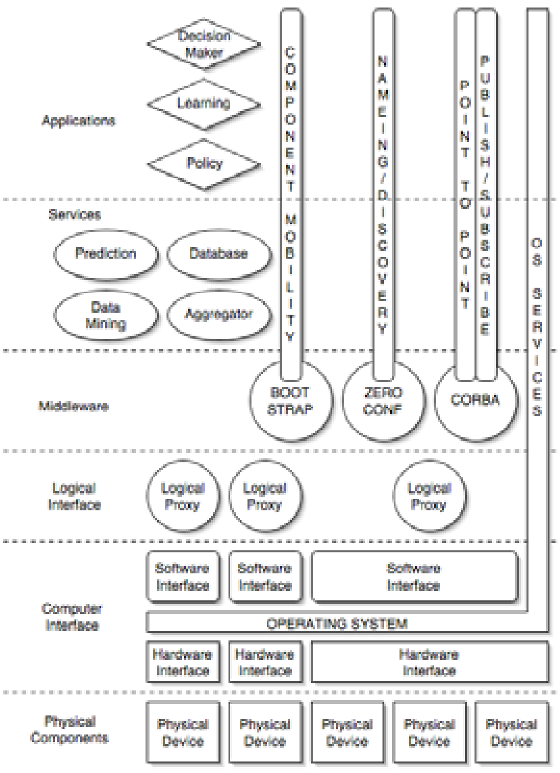

Fig. 3. MavHome concrete architecture

(e.g., PCI card interfaces, USB, firewire), device drivers to utilize the hardware, the operating system of the computer, and all software interfaces that provide services or APIs for hardware access. Note that since all components of above layers reside and utilize operating system services, these services are shown to extend to all layers.

In the *Logical Interface* layer, the hardware device services and APIs are utilized to create simple, light-weight programs that create a series of atomic services around each sensor and effector in the system. These *logical proxies* provide information and control via socket and shared memory based interfaces in a modular design. All of the lower layers are based on simple single application components, but in higher layers the components become more complex. The *Middleware* layer provides valuable services to the upper layers of the architecture to facilitate communication, process mobility, and service discovery. The MavHome architecture specifies middleware that provides both point-to-point and publish-subscribe types of communication, naming/service discovery provisions, and a mechanism to move system components between physical computing hardware devices. The *Services* layer utilizes the middleware layer to gather information from lower layers and provide information to system applications above. Services either store information, generate knowledge, aggregate lower level components, or provide some value-added non-decision making computational function or feature (e.g., user interfaces). The *Applications* layer is where learning and decision-making components operate.

4 Cooperation of Multiple Agent Components in MavHome

Automation of large, real-world, complex tasks is a pervasive goal of AI algorithms. Researchers develop innovative ideas and test them in simulation. Often, however, these researchers find that the ideas do not scale well or accurately to the larger problem that was initially targeted.

Smart environments represent one such large, real-world problem. Our approach to this problem is to combine the benefits of diverse, heterogeneous machine learning algorithms into a multi-agent system for controlling the environment. We have found that the combination of these algorithms performs better than each algorithm alone, and that the combined whole is effective for automating a complex, real-world home environment.

To automate our smart environment, we collect observations of manual inhabitant activities and interactions with the environment. We then mine sequential patterns from this data using a sequence mining algorithm. Next, we predict the inhabitant's upcoming actions using observed historical data. Finally, a hierarchical Markov model is created using low-level state information and high-level sequential patterns, and is used to learn an action policy for the environment. Figure 4 shows how these components work together to improve the overall performance of the smart environment.

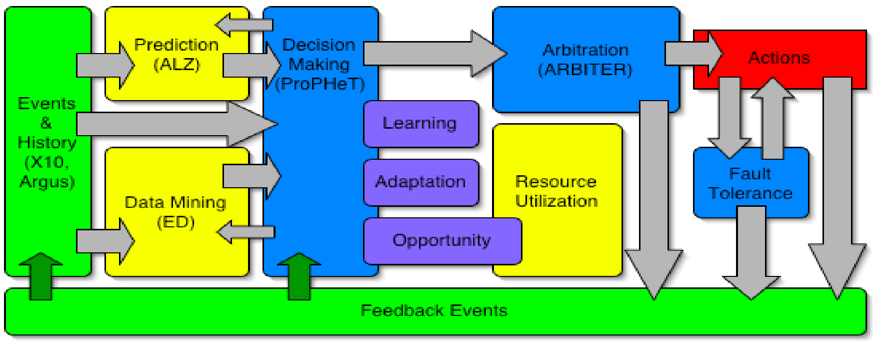

Fig. 4. Integration of AI techniques into MavHome architecture

Multi-agent technologies pervade artificial intelligence research [10, 11]. Their benefit to smart environments has been found primarily in unifying the diverse technologies that comprise a working smart environment [12, 13].

In order to create a cooperating multi-agent environment, we need to define a methodology for the agents to share information. MavHome uses the Common Object Request Broker Architecture (CORBA) to communicate between agents because of its clarity of interface design, ease of integration, and object-oriented design. In addition, all agent components register their presence using zero configuration (ZeroConf) technologies, which allows new agents to be dynamically introduced without redesigning the system architecture. Here we describe the learning algorithms that currently comprise the learning agents in MavHome.

4.1 Mining Sequential Patterns Using ED

In order to minimize resource usage, maximize comfort, and adapt to inhabitants, we rely upon machine learning techniques for automated discovery, prediction, and decision making. A smart home inhabitant typically interacts with various devices as part of his routine activities. These interactions may be considered as a sequence of events, with some inherent pattern of recurrence. Agrawal and Srikant [14] pioneered work in mining sequential patterns from time-ordered transactions, and our work is loosely modeled on this approach.

Typically, each inhabitant-home interaction event is characterized as a triple consisting of the device manipulated, the resulting change that occurred in that device, and the time of interaction. We move a window in a single pass through the history of events or inhabitant actions, looking for episodes (sequences) within the window that merit attention. Candidate episodes are collected within the window together with frequency information for each candidate. Candidate episodes are evaluated and the episodes with values above a minimum acceptable compression amount are reported. The window size can be selected automatically using the size that achieves the best compression performance over a sample of the input data.

When evaluating candidate episodes, the Episode Discovery (ED) algorithm [15] looks for patterns that minimize the description length of the input stream, O, using the Minimum Description Length (MDL) principle [16]. The MDL principle targets patterns that can be used to minimize the description length of a database by replacing each instance of the pattern with a pointer to the pattern definition.

Our MDL-based evaluation measure thus identifies patterns that balance frequency and length. Periodicity (daily, every other day, weekly occurrence) of episodes is detected using autocorrelation and included in the episode description. If the instances of a pattern are highly periodic (occur at predictable intervals), the exact timings do not need to be encoded (just the pattern definition with periodicity information) and the resulting pattern yields even greater compression. Although event sequences with minor deviations from the pattern definition can be included as pattern instances, the deviations need to be encoded and the result thus increases the overall description length. ED reports the patterns and encodings that yield the greatest MDL value.

Deviations from the pattern definition in terms of missing events, extra events, or changes in the regularity of the occurrence add to the description length because extra bits must be used to encode the change, thus lowering the value of the pattern. The larger the potential amount of description length compression a pattern provides, the more representative the pattern is of the history as a whole, and thus the potential impact that results from automating the pattern is greater.

In this way, ED identifies patterns of events that can be used to better understand the nature of inhabitant activity in the environment. Once the data is compressed using discovered results, ED can be run again to find an abstraction hierarchy of patterns within the event data. As the following sections show, the results can also be used to enhance performance of predictors and decision makers that automate the environment.

4.2 Predicting Activities Using ALZ

To predict inhabitant activities, we borrow ideas from text compression, in this case the LZ78 compression algorithm [17]. By predicting inhabitant actions, the home can automate or improve upon anticipated events that inhabitants would normally perform in the home. Well-investigated text compression methods have established that good compression algorithms also make good predictors. According to information theory, a predictor with an order (size of history used) that grows at a rate approximating the entropy rate of the source is an optimal predictor. Other approaches to prediction or inferring activities often use a fixed context size to build the model or focus on one attribute such as motion [18, 19].

LZ78 incrementally processes an input string of characters, which in our case is a string representing the history of device interactions, and stores them in a trie. The algorithm parses the string $x_1, x_2, \ldots, x_i$ into substrings $w_1, w_2, w_{c(i)}$ such that for all $j > 0$, the prefix of the substring w_j is equal to some w_i for $1 < i < j$. Thus when parsing the sequence of symbols *aaababbbbbaabccddcbaaaa*, the substring *a* is created, followed by *aa*, *b*, *ab*, *bb*, *bba*, and so forth.

Our Active LeZi (ALZ) algorithm enhances the LZ78 algorithm by recapturing information lost across phrase boundaries. Frequency of symbols is stored along with phrase information in a trie, and information from multiple context sizes are combined to provide the probability for each potential symbol, or inhabitant action, as being the next one to occur. In effect, ALZ gradually changes the order of the corresponding model that is used to predict the next symbol in the sequence. As a result, we gain a better convergence rate to optimal predictability as well as achieve greater predictive accuracy. Figure 5 shows the trie formed by the Active-LeZi parsing of the input sequence *aaababbbbbaabccddcbaaaa*.

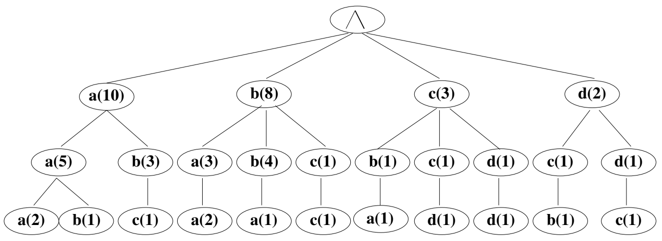

Fig. 5. Trie formed by ALZ parsing

To perform prediction, ALZ calculates the probability of each symbol (inhabitant action) occurring in the parsed sequence, and predicts the action with the highest probability. To achieve optimal predictability, we use a mixture of all possible higher-order models (phrase sizes) when determining the probability estimate. Specifically, we incorporate the Prediction by Partial Match strategy of *exclusion* [20] to gather information from all available context sizes in assigning the next symbol its probability value.

We initially evaluated the ability of ALZ to perform inhabitant action prediction on synthetic data based on six embedded tasks with 20% noise. In this case the predictive accuracy converges to 86%. Real data collected based on six students in the MavLab for one month was much more chaotic, and on this data ALZ reached a predictive performance of 30% (although it outperformed other methods). However, when we combine ALZ and ED by only performing predictions when the current activity is part of a sequential pattern identified by ED, ALZ performance increases by 14% [21].

4.3 Decision Making Using ProPHeT

In our final learning step, we employ reinforcement learning to generate an automation strategy for the intelligent environment. To apply reinforcement learning, the underlying system (i.e., the house and its inhabitants) could be modeled as a Markov Decision Process (MDP). This can be described by a four-tuple $< S, A, Pr, R >$, where S is a set of system states, A is the set of available actions, and $R : S \rightarrow R$ is the reward that the learning agent receives for

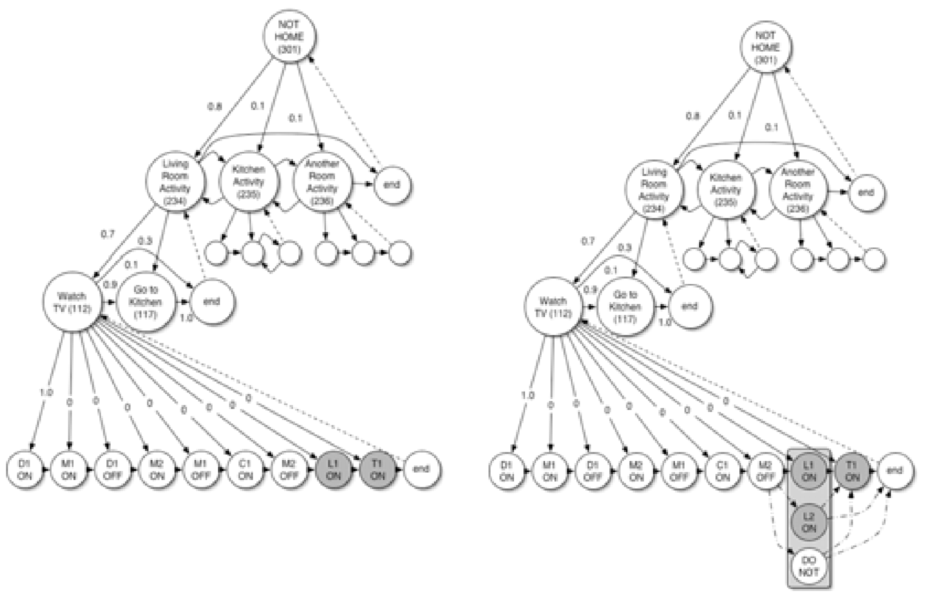

Fig. 6. Hierarchical model constructed from static (left) and dynamic (right) smart home data

being in a given state. The behavior of the MDP is described by the transition function, $Pr : S \times A \times S \rightarrow [0, 1]$, representing the probability with which action a_t executed in state s_t leads to state s_{t+1}.

With the increasing complexity of tasks being addressed, recent work in decision making under uncertainty has popularized the use of Partially Observable Markov Decision Processes (POMDPs). Recently, there have been many published hierarchical extensions that allow for the partitioning of large domains into a tree of manageable POMDPs [22, 23]. Research has shown that strategies for new tasks can be learned faster if policies for subtasks are already available [24]. Although a Hierarchical POMDP (HPOMDP) is appropriate for an intelligent environment domain, current approaches generally require *a priori* construction of the hierarchical model. Unlike other approaches to creating a hierarchical model, our decision learner, ProPHeT, actually automates model creation by using the ED-mined sequences to represent the nodes in the higher levels of the model hierarchy.

The lowest-level nodes in our model represent a single event observed by ED. Next, ED is run multiple iterations on this data until no more patterns can be identified, and the corresponding abstract patterns comprise the higher-level nodes in the Markov model. The higher-level *task* nodes point to the first event node for each permutation of the sequence that is found in the environment history. Vertical transition values are labeled with the fraction of occurrences for the corresponding pattern permutation, and horizontal transitions are seeded using the relative frequency of transitions from one event to the next in the observed history. As a result, the n-tier hierarchical model is thus learned from collected data. An example hierarchical model constructed from MavHome test data is shown on the left in Figure 6.

Given the current event state and recent history, ED supplies membership probabilities of the state in each of the identified patterns. Using this information along with the ALZ-predicted next action, ProPHeT maintains a belief state and selects the highest-utility action.

To learn an automation strategy, the agent explores the effects of its decisions over time and uses this experience within a temporal-difference reinforcement learning framework [25] to form control policies which optimize the expected future reward. The current version of MavHome receives negative reinforcement (observes a negative reward) when the inhabitant immediately reverses an automation decision (e.g., turns the light back off) or an automation decision contradicts Arbiter-supplied safety and comfort constraints.

Before an action is executed it is checked against the policies in the policy engine, Arbiter. These policies contain designed safety and security knowledge and inhabitant standing rules. Through the policy engine the system is prevented from engaging in erroneous actions that may perform actions such as turning the heater to 120°F or from violating the inhabitant's stated wishes (e.g., a standing rule to never turn off the inhabitant's night light).

5 Initial Case Study

All of the MavHome components are implemented and are being tested in two physical environments, the MavLab workplace environment and an on-campus apartment, the MavPad (portions of the environments are shown in Figure 7. Powerline control automates all lights and appliances, as well as HVAC, fans, and miniblinds. Perception of light, humidity, temperature, smoke, gas, motion, and switch settings is performed through a sensor network developed in-house. Inhabitant localization is performed using passive infrared sensors yielding a detection rate of 95% accuracy [26].

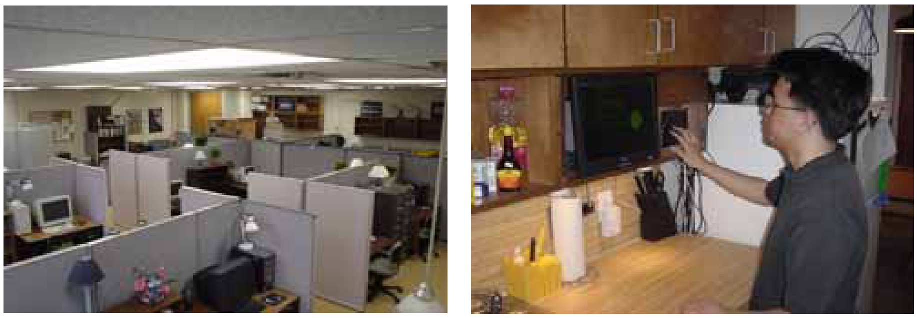

Fig. 7. The MavLab (left) and MavPad (right) environments

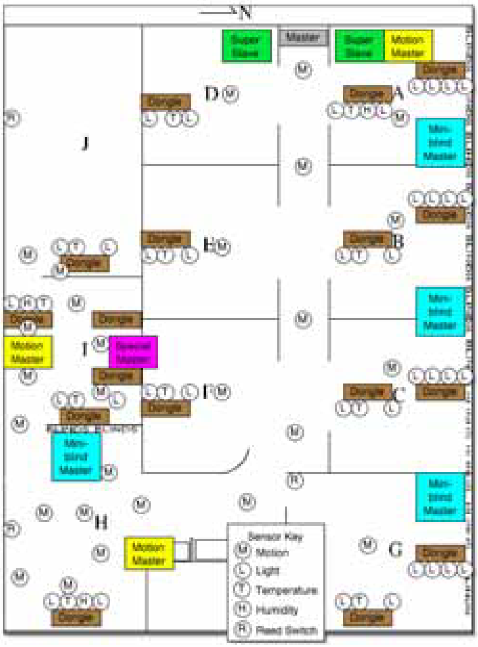

Fig. 8. MavLab sensor layout

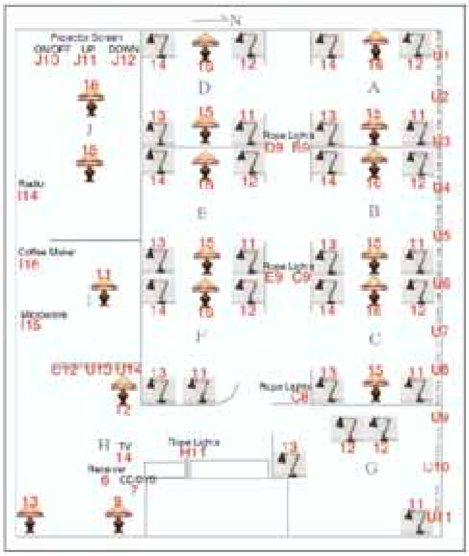

Fig. 9. MavLab actuator layout

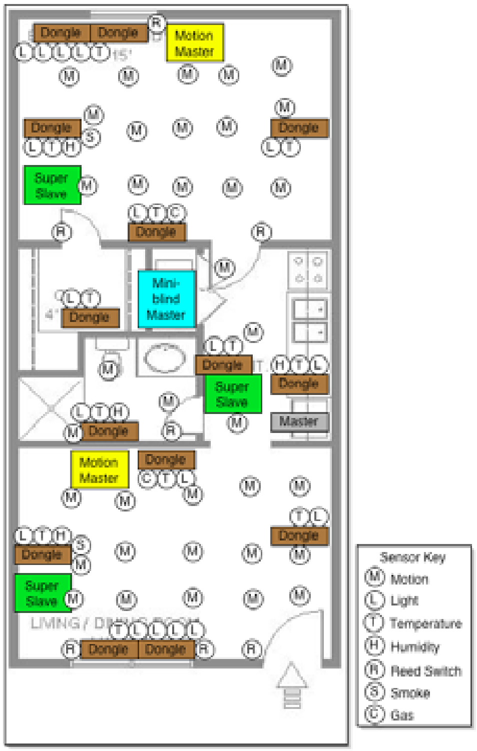

Fig. 10. MavPad sensor layout

The MavLab environment contains work areas, cubicles, a break area, a lounge, and a conference room. MavLab is automated using 54 X-10 controllers and the current state is determined using light, temperature, humidity, motion, and door/seat status sensors (see Figures 8 and 9). The MavPad is an on-campus apartment hosting a full-time student occupant. MavPad is automated using 25 controllers and provides sensing for light, temperature, humidity, leak detection, vent position, smoke detection, CO detection, motion, and door/window/seat status sensors. Figures 10 and 11 show the MavPad sensor and actuator layout. MavHome is designed to optimize a number of alternative functions, but for this evaluation we focus on minimization of manual interactions with devices.

As an illustration of the above techniques, we have evaluated a week in an inhabitant's life with the goal of reducing the manual interactions in the MavLab. The data was generated from a virtual inhabitant based on captured data from the MavLab and was restricted to just motion and lighting interactions which account for an average of 1400 events per day.

ALZ processed the data and converged to 99.99% accuracy after 10 iterations through the training data. When automation decisions were made using ALZ alone, interactions were reduced by 9.7% on average. Next, ED processed the data and found three episodes to use as abstract nodes in the HPOMDP. Living

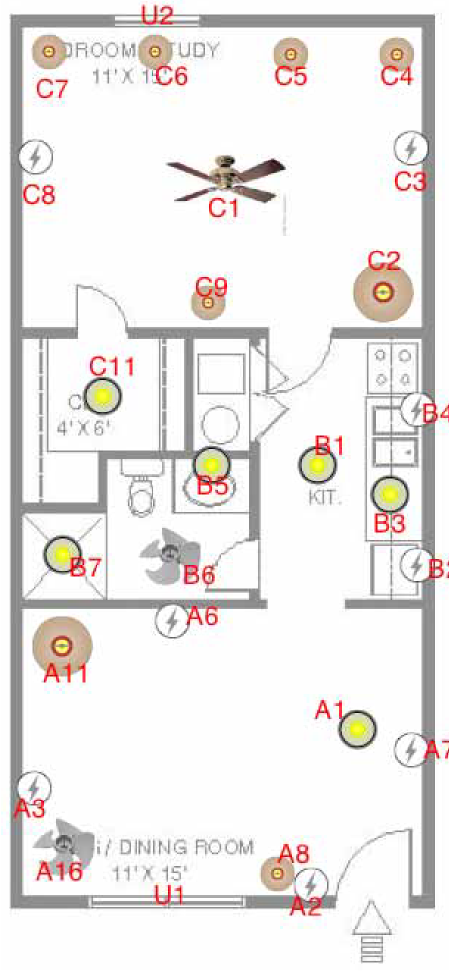

Fig. 11. MavPad actuator layout

room patterns consisted of lab entry and exit patterns with light interactions, and the office also reflected entry and exit patterns. The other patterns occurred over the remaining 8 areas and usually involved light interactions at desks and some equipment upkeep activity patterns. The hierarchical Markov model with no abstract nodes reduced interactions by 38.3%, and the combined-learning system (ProPHeT bootstrapped using ED and ALZ) was able to reduce interactions by 76%, as shown in Figure 12 (left).

Experimentation in the MavPad using real inhabitant data has yielded similar results. In this case, ALZ alone reduced interactions from 18 to 17 events, the HPOMDP with no abstract nodes reduced interactions by 33.3% to 12 events, while the bootstrapped HPOMDP reduced interactions by 72.2% to 5 events. These results are graphed in Figure 12 (right).

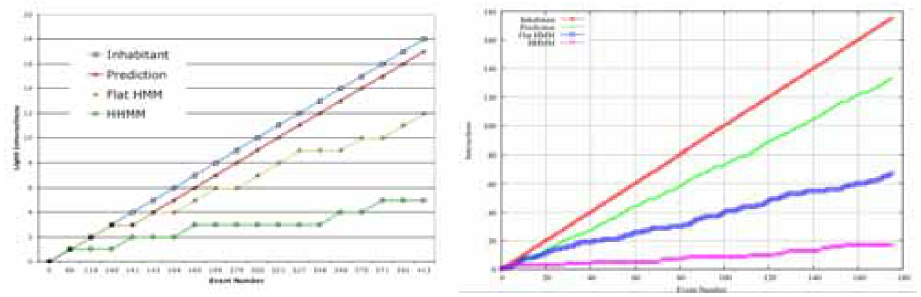

Fig. 12. Interaction reduction

6 Using a Multi-agent Smart Home to Assist Elderly and Disabled

An important application of the multi-agent technologies available in MavHome is to provide health care assistance in living environments. Specifically, models can be constructed of inhabitant activities and used to learn activity trends, detect anomalies, intelligently predict possible problems and make health care decisions, and provide automation assistance for inhabitants with special needs.

A variety of approaches have been investigated in recent years to automate caregiver services. Many of the efforts offer supporting technologies in specialized areas, such as using computer vision techniques to track inhabitants through the environment and specialized sensors to detect falls or other crises. Some special-purpose prediction algorithms have been implemented using factors such as measurement of stand-sit and sit-stand transitions and medical history [27, 28, 29], but are limited in terms of what they predict and how they use the results. Remote monitoring systems have been designed with the common motivation that learning and predicting inhabitant activities is key for health monitoring, but very little work has combined the remote monitoring capabilities with prediction for the purpose of health monitoring. Some work has also progressed toward using typical behavior patterns to provide reminders, particularly useful for the elderly and patients suffering from various types of dementia [30, 31].

Our smart environment can identify patterns indicating or predicting a change in health status and can provide inhabitants with needed automation assistance. Collected data includes movement patterns of the individual, periodic vital signs (blood pressure, pulse, body temperature), water and device usage, use of food items in the kitchen, exercise regimen, medicine intake (prescribed and actual), and sleep patterns [32, 1]. Given this data, models can be constructed of inhabitant activities and use to learn lifestyle trends, detect anomalies, and provide reminder and automation assistance.

6.1 Capability 1: Identify Lifestyle Trends

Our ED algorithm is designed to process data as it arrives. Because of this feature, trends in the data including increasing / decreasing pattern frequency, introduction of patterns, and change in pattern details can be automatically detected [33]. When changing patterns include health-specific events (vital signs, medication intake, or events targeted by the caregiver), a report will be given to the inhabitant and caregiver of these trends.

6.2 Capability 2: Detect Anomalies in Current Data

The ED data mining algorithm and ALZ predictor can work together to detect anomalies in event data. ED identifies the most significant and frequent patterns of inhabitant behavior, as well as the likelihood that the current state is a member of one of these patterns. Whenever the current state falls within one of these patterns, ALZ can determine the probability distribution of next events. As a result, when the next event has a low probability of occurrence, or when the expected next event does not occur at the expected time, the result is considered an anomaly.

When an anomaly occurs, the home will first try to contact the inhabitant (through the interactive display for a lesser critical anomaly, or through the sound system for a more critical anomaly). If the inhabitant does not respond and the criticality of the anomaly is high, the caregiver will be

6.3 Capability 3: Design Reminder Assistance System

Reminders can be triggered by two situations. First, if the inhabitant queries the home for his next routine activity, the activity with the highest probability will be given based on the ALZ prediction. Second, if a critical anomaly is detected, the environment will initiate contact with the inhabitant and remind him of the next typical activity. Such a reminder service will be particularly beneficial for individuals suffering from dementia.

As described in the initial MavHome design, automation assistance is always available for inhabitants, which is beneficial if some activities are difficult to perform. A useful feature of the architecture is that safety constraints are embedded in the Arbiter rule engine. If the inhabitant or the environment is about to conflict with these constraints, a preventative action is taken and the inhabitant notified. This can prevent accidents such as forgetting to turn off the water in the bathtub or leaving the house with doors unlocked.

7 Conclusion

The MavHome software architecture has successfully monitored and provided automation assistance for volunteers living in the MavPad site. We are currently collecting health-specific data and in the MavHome sites and will be testing in recruited residents of the C.C. Young Retirement Community in Dallas, Texas.

Acknowledgements

This work is supported by US National Science Foundation under ITR grant IIS–0121297.

References

1. Das, S.K., Cook, D.J.: Health monitoring in an agent-based smart home by activity predition. In Zhang, D., Mokhari, M., eds.: Toward a Human-Friendly Assistive Environment. IOS Press (2004) 3–14
2. AIRE Group: MIT Project AIRE – About Us (2004) http://www.ai.mit.edu/projects/aire.
3. Fox, A., Johanson, B., Hanrahan, P., Winograd, T.: Integrating information appliances into an interactive space. IEEE Computer Graphics and Applications **20** (2000) 54–65
4. Romn, M., Hess, C.K., Cerqueira, R., Ranganathan, A., Campbell, R.H., Nahrstedt, K.: Gaia: A middleware infrastructure to enable active spaces. IEEE Pervasive Computing (2002) 74–83
5. Abowd, G.D., Mynatt, E.D.: Designing for the human experience in smart environments. In Cook, D.J., Das, S.K., eds.: Smart Environments: Technology, Protocols, and Applications. Wiley (2005) 153–174
6. Helal, A., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., Jansen, E.: The gator tech smart house: A programmable pervasive space. IEEE Computer **38** (2005) 50–60
7. NIST: Smart space NIST laboratory. http://www.nist.gov/smartspace/ (2005)
8. Mozer, M.C.: Lessons from an adaptive home. In Cook, D.J., Das, S.K., eds.: Smart Environments: Technology, Protocols, and Applications. Wiley (2005) 273–298
9. Hagras, H., Callaghan, V., Colley, M., Clarke, G., Pounds-Cornish, A., Duman, H.: Creating an ambient-intelligence environment using embedded agents. IEEE Intelligent Systems **19** (2004)
10. Adams, J.A.: Multiagent systems: A modern approach to distributed artificial intelligence. AI Magazine **22** (2001) 105–108
11. Stone, P., Veloso, M.: Multiagent systems: A survey from a machine learning perspective. Autonomous Robots **8** (2000) 345–383
12. CSIRO: Intelligent interactive technology. http://www.cmis.csiro.au/iit/ (2005)
13. Haigh, K.Z., Phelps, J., Geib, C.W.: An open agent architecture for assisting elder independence. In: Proceedings of the First International Joint Conference on autonomous Agents and Multiagent Systems. (2002) 578–586
14. Agrawal, R., Srikant, R.: Mining Sequential Patterns. In: Proceedings of the 11th International Conference on Data Engineering. (1995) 3–14
15. Heierman, E.O., Cook, D.J.: Improving home automation by discovering regularly occurring device usage patterns. In: Proceedings of the International Conference on Data Mining. (2003)
16. Rissanen, J.: Stochastic Complexity in Statistical inquiry. World Scientific Publishing Company (1989)
17. Ziv, J., Lempel, A.: Compression of individual sequences via variable rate coding. IEEE Transactions on Information Theory **IT-24** (1978) 530–536
18. Cielniak, G., Bennewitz, M., Burgard, W.: Where is ...? learning and utilizing motion patterns of persons with mobile robots. In: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence. (2003) 909–914

19. Philipose, M., Fishkin, K., Perkowitz, M., Patterson, D., Fox, D., Kautz, H., Hahnel, D.: Inferring activities from interactions with objects. Pervasive Computing **3** (2004) 50–56
20. Bell, T.C., Cleary, J.G., Witten, I.H.: Text compression. Prentice Hall (1990)
21. Gopalratnam, K., Cook, D.J.: Online sequential prediction via incremental parsing: The Active LeZi algorithm. IEEE Intelligent Systems (2005)
22. Pineau, J., Roy, N., Thrun, S.: A Hierarchical Approach to POMDP Planning and Execution (2001) Workshop on Hierarchy and Memory in Reinforcement Learning (ICML).
23. Theocharous, G., Rohanimanesh, K., Mahadevan, S.: Learning Hierarchical Partially Observable Markov Decision Processes for Robot Navigation (2001) IEEE Conference on Robotics and Automation.
24. Precup, D., Sutton, R.S.: Multi-time models for temporally abstract planning. Advances in Neural Information Processing Systems **10** (1997) 1050–1056
25. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)
26. Youngblood, G.M., Holder, L.B., Cook, D.J.: A learning architecture for automating the intelligent environment. In: to appear in Proceedings of the Conference on Innovative Applications of Artificial Intelligence. (2005)
27. Cameron, K., Hughes, K., Doughty, K.: Reducing fall incidence in community elders by telecare using predictive systems. In: Proceedings of the International IEEE-EMBS Conference. (1997) 1036–1039
28. Najafi, B., Aminian, K., Loew, F., Blanc, Y., Robert, P.: Measurement of stand-sit and sit-stand transitions using a miniature gyroscope and its application in fall risk evaluation in the elderly. IEEE Transactions on Biomedical Engineering **49** (2002) 843–851
29. Najafi, B., Aminian, K., Paraschiv-Ionescu, A., Loew, F., Bula, C., Robert, P.: Ambulatory system for human motion analysis using a kinematic sensor: Monitoring of daily physical activity in the elderly. IEEE Transactions on Biomedical Engineering **50** (2003) 711–723
30. Kautz, H., Arnstein, L., Borriello, G., Etzioni, O., Fox, D.: An overview of the assisted cognition project. In: Proceedings of the AAAI workshop on automation as caregiver. (2002)
31. Pollack, M.E., Brown, L., Colbry, D., McCarthy, C.E., Orosz, C., Peintner, B., Ramakrishnan, S., Tsamardinos, I.: Autoreminder: An intelligent cognitive orthotic system for people with memory impairment. Robotics and Autonomous Systems **44** (2003) 273–282
32. Das, S.K., Cook, D.J.: Health monitoring in an agent-based smart home. In: Proceedings of the International Conference on Smart Homes and Health Telematics (ICOST). (2004)
33. Heierman, E.O.: Using information-theoretic principles to discover interesting episodes in a time-ordered sequence. PhD thesis, The University of Texas at Arlington (2004)