Intelligent Automation and Soft Computing, Vol. 15, No. 2, pp. 277-289, 2009

APPLICATION OF REINFORCEMENT LEARNING IN MULTISENSOR FUSION PROBLEMS WITH CONFLICTING CONTROL OBJECTIVES

SHICHAO OU[1], ANDREW H. FAGG[2], PRASHANT SHENOY[1], LIANCHENG CHEN[3]

[1]*Department of Computer Science,*
University of Massachusetts,
Amherst, MA 01003, USA
{chao,shenoy}@cs.umass.edu

[2]*School of Computer Science*
University of Oklahoma
Norman, OK 73019
fagg@cs.ou.edu

[3]*College of Information*
South China Agricultural University
Guangzhou, China, 510642
liancheng@scau.edu.cn

ABSTRACT—Smart agents are equipped with sensors that enable them to be sensitive to their surrounding environment. However, the mapping of multiple raw streams of sensory data to the appropriate actions is not an easy problem, especially when multiple conflicting objectives are involved. This type of multi-sensor fusion problem through the domain of power management for smart mobile devices was investigated in this study. In this application domain, the objective is to keep the user's mobile devices in an "on" state as long as possible such that just-in-time services (such as reminder announcements) can be provided. However, this conflicts with the mobile device's inherent goal – to turn off to conserve power. Due to the stochastic nature of human behavior, a hand-coded fixed strategy may not be the best solution. A learning control approach to the problem is presented in this paper. The experimental results show that the approach learns the appropriate mapping from multiple streams of raw sensory data to power conserving actions that can out-perform the hand-crafted policies. The learned policies are also shown to be more robust in handling unscheduled events.

Key Words: Multi-sensor fusion, conflicting objectives, reinforcement learning, adaptive power management

1. INTRODUCTION

When smart agents, physical (robots) or virtual (software agents), to work to assist humans, sensors are needed for detecting changes in the environment. However, the mapping of multiple raw streams of sensory data to the appropriate actions is a difficult task, especially when multiple *conflicting* objectives are involved. This type of *multi-sensor fusion* problem is studied in this paper through the domain of smart mobile devices. Smart assistant digital agents running on a

PDA, a wearable computer or a capability-rich cell phone can provide digital services to a user based on the user's current activity [1, 13]. Such tasks may include reminding the user of relevant upcoming events [15], autonomously handling incoming phone calls under different context [2, 8], or pre-caching web pages [16] or emails for offline browsing. These proactive assistive actions often occur in the background when the user is not physically interacting with the device. Therefore, the device must stay *ON* (even though the user is not actively using it) in order to collect information and to assist the user. Ideally, it should be *ON* all the time such that no event is missed. However, this is not possible with the current battery technology. Therefore, for the device, there exist two conflicting objectives: to stay *ON* for as long as possible to assist the user, and to take power conservation actions without causing annoyance to the user or missing important events. Due to the stochastic nature of human behavior, a hand-coded fixed strategy may not be the best solution.

The approach taken by this work is a learning control algorithm, Reinforcement Learning (RL) [21]. With various physical sensors (e.g., location sensors [5, 3, 10, 12]) as well as data streams from software applications (e.g., calendar events, email notifications, application usage patterns), states of the world (from the user's perspective) can be captured. Any control actions taken by the system has consequences, some positive and some negative. They are referred to in this paper as costs. By associating control actions with costs, and designing an appropriate cost model for the application, the RL system can autonomously learn the mapping between multiple sensory streams and the correct sequence of system actions such that total cost can be minimized, thus balancing the two opposing objectives.

2. A LEARNING CONTROL APPROACH

To address the issue of mapping sensor data to the appropriate action, an approach from the feedback control domain is employed. In the form of a control problem, the actions that the system takes are treated as control actions that change the state of the world and the user: after action a_t state s_t is changed to a new state s_{t+1}. Consequently, the system receives an updated state estimate as well as a signal that represents the cost associated with taking the last action. From this feedback, using traditional RL learning algorithms such as Q-learning, the agent can learn a value function such that given the current state, the agent can determine which action it should take next so as to minimize the total cost in the future.

However, unlike conventional control problems, where control decisions are made at regular intervals, in the domain of context-aware applications, the intervals between events can be irregular. As a result, the control decisions also occur at irregular intervals. For example, the user may receive emails consecutively and then wait a long time before the next email arrives. In this case, the system may choose to take actions to reduce the frequency of checking for emails and direct the available resources for other more urgent tasks. The Semi-Markov Decision Process (SMDP) approach to Q-learning approach handles the irregularity issue [21], and allows the system to change behavior as a function of user context.

In the formulation of SMDPs (Figure 1), the world model is that from a given state, the system makes a choice of an action. Then at some nondeterministic time later, the state changes and system receives a report of the new state, plus the immediate cost. The goal is to take the sequence of actions that minimizes the discounted sum of these costs. In particular, the system would choose an action a_t to minimize the expected discounted return (total cost starting from time t):

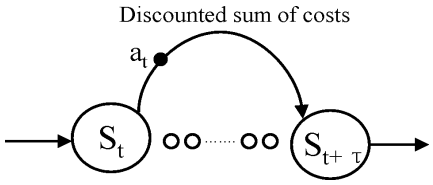


Figure 1. Semi-Markov Decision Processes (SMDP) Q-learning.

$$E_t = C_{t+1} + \gamma C_{t+2} + \gamma^2 C_{t+3} + \cdots = \sum_{k=0}^{n} \gamma^k C_{t+k+1}$$

where $\gamma < 1$, is a discount factor that causes costs for further away (time-wise) actions to have less of an impact on the value of the current state-action pair. To achieve minimizing the discounted sum of costs, a way is needed to estimate the "value" of taking an action from a given state. In this work, the value of an action a_t is defined as the negative discounted sum of costs that result from first selecting a_t and then selecting the estimated highest value action for the rest of the sequence. SMDP Q-learning is an algorithm that incrementally estimates these action values, optimizing the parameters to minimize the discounted sum of the future costs. The discrete SMDP Q-learning value update function is given as below [21]:

$$Q_\tau(s,a) \leftarrow Q_t(s,a) - \alpha[C_{t+1} - \gamma C_{t+2} - \gamma^2 C_{t+3} - \ldots \gamma^{-1} C_{t+\tau} + \gamma^\tau \max_a Q_t(s_{t+\tau}, a_{t+\tau}) - Q_t(s,a)] \quad (1)$$

where $Q_t(s,a)$ is the estimated action value for action a when state is *s*. We assume the duration of the action *a* is τ , which varies with different actions. The cost associated with the selected action at time *t* is represented as C_t .

State spaces for general problems can be prohibitively large, especially when continuous variables such as time are considered and thus the state space becomes continuous. Therefore, function approximation (FA) is employed in this work. Instead of learning an enormous action-value lookup table as in the discrete case, with FA, an approximate continuous multi-dimensional function is learned to best fit the real action-value function. Furthermore, when facing unforeseen situations or unvisited states, FA allows for extrapolation on the past learned action-values that are similar, and thus informed guesses of the state-action value can be made in these cases.

3. CASE STUDY: ADAPTIVE POWER CONSERVATION

To demonstrate the application of learning control optimization approach for multi-sensor fusion with conflicting objectives, the mobile device power conservation problem is selected as an example. This approach applies equally well to other domains, such as mobile robotics. Under the context of mobile devices, the system needs to take the appropriate sequence of actions that makes the trade-off between two conflicting goals of both conserving power to stay *ON* for longer and of minimizing missed events. The follow table is the cost model for our problem.

3.1. A SMDP Learning System

To formulate this as a SMDP Q-learning problem, a cost model is first established. The costs for staying *ON* and for missing events has to be combined into the same cost model such that the

Table I. Cost model

Case	Action taken	Events	C_{power}(joules)	C_{event}
1	OFF	user event	279	1
2	OFF	non user event	0	0.5
3	OFF	None	279	0
4	KEEP ON	Any	186	0

value of an action can be computed (using equation (1)) for evaluation. The cost for missing events is denoted as C_{event}, and the cost for power consumption as C_{power}. There are two types of power consumption: keeping the PDA running and switching from *OFF* to *ON*. Using the Sharp Zaurus as an example (whose power consumption is 6.2 watts), power needed to keep the PDA ON per unit time (30 seconds) is $C_{power\_run} = 6.2 \times 30 = 186$ joules. It is assumed that more energy is required to switch the PDA ON than to keep it running. Therefore, we define $C_{power\_on} = 1.5 C_{power\_run} = 1.5 \times 186 = 279$ joules. For missed events, there are also two types: user events and non-user events; the associated cost for both is denoted as C_{event}. However, due to the fact that the units for C_{power} and C_{event} are different, they cannot be combined directly. Therefore, a regularization parameter α is introduced for the purpose of costs aggregation. Specifically, using α, C_{event} can be converted to units in joules, denote as $C'_{event} = \alpha C_{event}$ joules. Finally both costs can be combined to compute the total cost:

$$C_t = C_{power} + C'_{event} = C_{power} + \alpha C_{event}$$

For any action *a*, assuming that the action execution time is τ, at any given time instance *t* on the timeline, the costs C_{power} and C_{event} are defined as follows:

In case 1, the system decides to turn OFF for time τ, and a user event occurrs during the sleep time. The user was forced to manually turn on the PDA since he needed to use the PDA. Thus, the total cost c_i includes both the cost of switching ON the PDA, and the cost of missing a user event. For case 2, again, the OFF action was chosen, and we assume for this duration, only non-user events (such as changing rooms, or incoming emails) were missed. Compared with missing user events that forces the user to turn on the PDA repeatedly, missing non-user events causes less annoyance. Therefore, we define C_{event}= 0.5 in this case to reflect lesser annoyance. Since the PDA does not turn *ON* until the current *OFF* action ends, no cost for power consumption is needed. For case 3, since nothing happened during the *OFF* period, no event is missed and the cost only involves the power spent for switching ON at the end of the OFF period. For the last case, where the system chose to stay ON to anticipate for future events, since no events will be missed during this time, the only cost is the accumulative cost of staying ON for τ time, which is the length of the selected action. These cover all the cases for all possible actions.

It is assumed that the state of the world (with respect to the user) can be captured using the available sensors. The state representation in this application includes the following state variables: location, application usage, time of events, scheduled events and recent activities. A

linear neural network is used as the function approximator to both handle continuous variables and unforeseen situations. 4 *ON* actions and 4 *OFF* actions are chosen as the available action set. The durations of these actions are: 30 seconds, 1 minute, 2 minutes and 5 minutes, both for the *ON* and *OFF* set.

3.2 User Activity Simulator

To allow for analysis the SMDP Q-learning algorithm under various situations, a large set of diverse, interesting and repeatable user activities are needed. A user activity simulator is implemented to generate simulated user experiences. This guarantees repeatability, which is crucial for in-depth analysis.

The simulator contains five different probabilistic activity models for the same user: each corresponds to one of the weekdays. Thus, the user has a different schedule for each day. For instance, for Monday (Figure 2), the circles represent the state of the user and the edges indicates the transition probability between the states. Throughout the day, the user context transitions from one state to another based on the defined the activity duration model and the probabilistic transition model. Each activity has a predefined a duration. However, each person has different habits of arrival time for different event. Also, unpredictable events (traffic, weather) add randomness to these events. To model this variability, a *user action uncertainty* parameter σ is introduced. The arrival and departure time is defined by the nominal transition time t and a random offset generated from a Gaussian distribution: $t + N(0, \sigma_1)$, where σ_1 is proportional to σ and is different under different contexts. The relationship between σ and σ_1 is given in Table 2. As for the transition probabilities, by design, the scheduled events have a high transition probability (e.g. 90%) such that they are more likely to occur. Unscheduled events are also modeled with transition probabilities, e.g. with 5% probability, class 1 may run late and the user may skip the free time and directly transition to class 2.

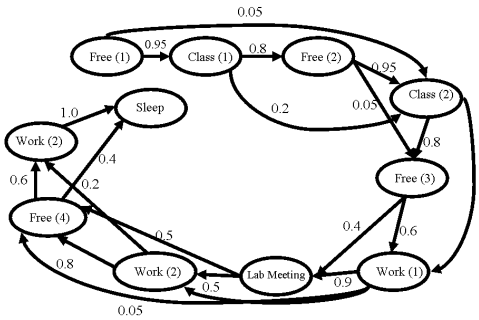


Figure 2. Monday Schedule User Activity Transition Probabilistic Model.

Once the simulator transitions into a state, user events are simulated using actions from the behavior model, such as note-taking or email-checking behavior defined under the current context. For example, during class the user exhibits a frequent note-taking behavior. When the user works in the lab, much longer rest intervals are more common since the user tends to use the desktop during this period, with only occasional checks of calendar schedules. An idle interval period is associated with each of these behaviors. This interval period is defined by $N(\mu_2, \sigma_2)$,

where μ_2 is the mean transition time, and σ again is the user action uncertainty parameter. The relationship between σ and σ_2 is also given in Table II. Upon each simulated event, the event is logged as well as the corresponding sensor readings (such as time or scheduled activity) are generated and recorded at the same time. These form the simulated sensory stream to be observed by the learning system.

Table II. Activity duration and transition time model.

Event	*t*(start/end)	σ_1	μ_2	σ_2
Class(1)	9:30am/10:45am	2σ	10	σ
Class(2)	11:15am/12:30pm	2σ	10	σ
Work(1)	1:00pm/3:15pm	4σ	5	σ*
LabMeeting	3:30pm/5:00pm	2σ	15	1.5σ
Work(2)	5:00pm/7:00pm	2σ	5	σ*

(*) The idle interval for the *work* period depends on a different transition model: work duration between 20 minutes and 1 hour is drawn from a uniform distribution. The PDA will be idle during the work duration. Between the work periods, the user uses the PDA to download emails or check schedules and thus the similar activity model used in other events are used again.

4. EXPERIMENTAL RESULTS AND DISCUSSION

The goal of the following experiments is to explore the effects of the control optimization technique: (1) with this approach, can the system learn to translate multiple raw sensory inputs into appropriate sequence of system actions (for which we will use the term policy in the remaining discussion), optimizing on two conflicting goals? (2) Can the learned policy outperform the user-defined policies? (3) Can the learned policy robust handle unscheduled events?

4.1 Learning Appropriate Action Policies From Multiple Sensor Streams

The first experiment examines the feasibility of the learning control approach to produce appropriate action policies from multiple raw sensor data streams. The system is trained on a series of simulated experiences drawn from the Monday activity model. One hundred daily experiences are then drawn separately for evaluation. Performance for a single trial is measured in terms of the amount of power consumed and the number of missed events.

The top panel of Figure 3 shows how the system's behavior changes throughout the day. When the user is in a class, the system prefers to turn the PDA *OFF* for 2 minutes over the other *off* actions. Occasionally, 5-minute *OFF* actions were used. This is the case because the user exhibits a frequent note-taking behavior, with idle intervals drawn from Gaussian distributions with a mean value of either 2 or 5 minutes. When the user works in the lab, much longer rest intervals are more common since the user tends to use the desktops during this period, with only occasional checks of calendar schedules. Likewise for lab meetings, although the user also takes notes during lab meetings, a different usage pattern is used. Therefore, the system chooses a distinctively different *ON/OFF* pattern compared with the class sessions. Most of the user events are anticipated since the system is able to turn on the system at the right times. Notice that although the agent can select 1-minute or 30-second *OFF* actions, these are never chosen. This is because the cost associated with turning on the PDA after an *OFF* action is higher than simply staying on for same the duration.

The bottom panel of figure 3 summarizes the maximum values of the *ON* action and *OFF* actions over a single trial. At any given time, the action selection policy is greedy with respect to the action values, i.e. the action with the highest value is chosen. The shapes of the Q action value function are noticeably different when the context of the user switches from one activity to the next. For instance, the greedy action switches between *ON* and *OFF* more frequently in the class period than the work period in the lab. As a result, the system using the learned policy wakes up much more frequently in anticipation for user events. On the other hand, the *OFF* period is much longer when the user is working in lab. The choices that the learned policy made are reasonable because during classes, the user works in an active note-taking mode with some intermittent idle time when listening to lectures. Therefore, more frequent wake-up actions are needed. When working in the lab, the user tends to use the desktop instead of the PDA. During this period, the PDA is occasionally used for checking upcoming appointments. As a result, longer sleep periods are used to conserve power. This shows that the system is able to learn to differentiate between user activity patterns.

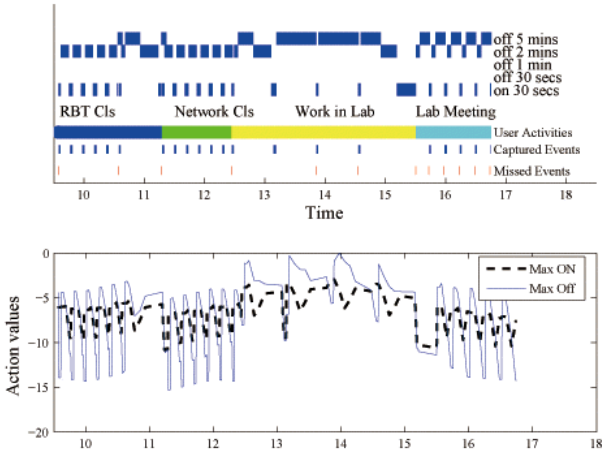


Figure 3. Learned behavior over a single trial (α = 8).

The action value for the *OFF* action dramatically decreases shortly prior to the occurrence of an event (allowing the on action to take over). This is a reasonable strategy since the cost function assigns a cost to those actions that result in missed events. On the other hand, the choice of an *ON* action immediately following an event that has more value than an *OFF* action, since the likelihood of an event remains high sometime after the previous event. Staying on allows the system to avoid receiving the high cost of missing a user event. Therefore, over time, the system learns that the values for the *OFF* actions are much lower than the *ON* action when an activity occurs. Similarly, there are also costs associated with *ON* actions. The phenomenon is reversed when during idle periods: that the value of *ON* action decreases and the value of *OFF* action recovers. Thus, *OFF* mode takes over during idle periods. The training experiences and the design of the cost model dictate the expected action values. In turn, the action values dictate the resulting policy that uses different *ON* and *OFF* patterns under different user contexts.

The top panel illustrates the system behavior under different user contexts. The horizontal axis is the time line, from 9:15 to 18:00. The top 5 rows show which actions were selected at any given time. The 6th row shows the duration of each scheduled activity (e.g., in a class, in a meeting). Anticipated and missed are shown as tick marks in the two remaining rows. The bottom

panel summarizes the maximum values of the *on* (dotted line) and *OFF* (solid line) actions over time. The system has learned to adapt to the user's habits by switching to the appropriate *ON/OFF* pattern when the user context changes.

4.2 Performance Comparison

Since the cost model is one of the important factors that dictates the resulting policy, it is conceivable that changing the cost model will result in a series of different policies and thus a variety of system behaviors (Figure 4). Compared with the top panel of figure 3, the action selection behavior change most occurred in the lab meeting session: due to a different cost model, less 5-minutes *OFF* actions, more 2-minute *OFF* actions and ON actions were selected. As a result, fewer events were missed. In the design of the cost model, a regularization power/correctness trade *OFF* parameter α is introduced, to linearly combine the costs associated with power consumption and missed events.

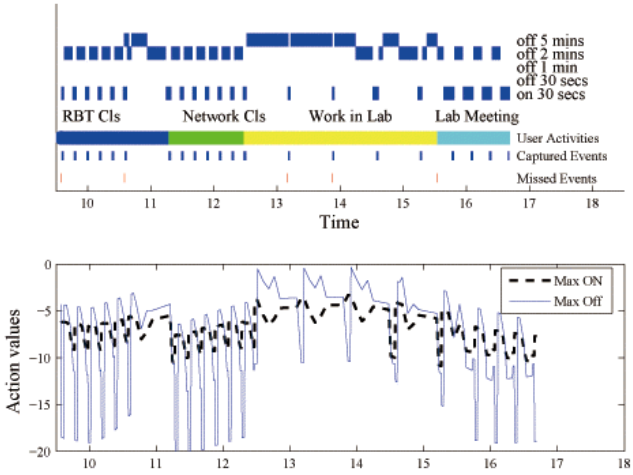


Figure 4. Learned behavior when the cost of missed events is higher (α = 16).

As a side-effect, changing α also alters the relationship between the two conflicting costs. These effects are shown in Figure 5.

Each data point represents the performance of the learned control policy over a single trial. For each learned policy, the performance on 30 distinct trials is shown. As shown in the figure, there is a distinct distribution associated with each α value. As α is increases, better performance is achieved with respect to the number of missed events. This comes at the cost of increased power consumption. To view the trend with a wider range of α values, the overall performance of an individual control policy is measured as an average over one hundred trials of the number of missed events and of the consumed power. Figure 6 shows the overall performance for policies resulting from N-values of α. A Least-Mean-Squared-fit inverse function (E= α/P+b: where E denotes the number of missed event and P denotes the amount of power consumed) is superimposed on the data, and shows a general trend of trading power consumption for missed events as α is increased.

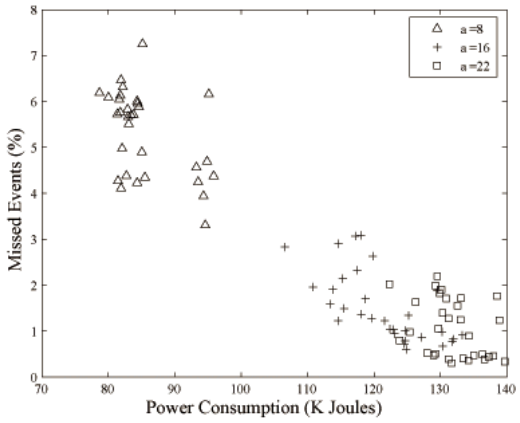


Figure 5. Trial performance under three learned policies (α = 8, 16, 22)

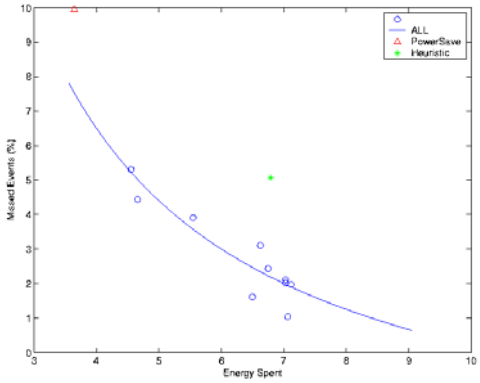


Figure 6. Performance of a range of learned policies trained under 10 cost models, compared against the performance of the user-defined policies. For the learned policies, as α increases, better performance is achieved with respect to the number of missed events. This comes at the cost of increased power consumption. Comparing against the user-defined policies, a learned policy can be found such that it performs significantly better than the user-defined policy in one dimension if the performance over the other dimension is matched.

The α parameter can be used as an intuitive mechanism for the user to alter the system behavior in order to suit his needs. If the user feels that the system the needs to be more attentive to his activities, then he can increase the cost for missing events by increasing α. If he is willing to sacrifice missed events in exchange for longer battery life, then α should be lowered. The control optimization formulation of the problem allows us to translate cost functions into a specific prescription for action under different contexts, and thus an appropriate performance trade-off solution for a certain behavior value can be computed. The end result is a much simpler and more flexible way to adjust system behavior than the complex multivariate rule set system that requires manual configuration by the user.

For comparison purposes, two heuristic policies are designed. The *Power-saving* heuristic is defined to aggressively conserve power with no regard for missing events: it turns off the PDA

whenever the system is idle. This is the lower-bound for power conservation because it represents the minimal power necessary to address the user's requests, with no regard for missing user events. The balanced heuristic consists of a set of hand-designed rules that are intended to balance power consumption against the number of missed events. The performance of these heuristics is measured by averaging the power consumption and the number of missed events over one hundred trials.

4.3 Robustness Analysis

Although human behavior exhibits patterns that have strong correlation with scheduled activities, there are times when the user will deviate from the typical schedule. Some of these deviations are dramatic, e.g. canceling meetings, changing locations, or even moving scheduled events to an earlier time and date. Some deviations are minor, e.g. arriving late for classes. Robustness of the learned control policies is tested on both of these unscheduled human activities. For scheduled activities, both the training and test trial experiences are drawn from the same activity model. The unscheduled activities are simulated by drawing training and test trials from different activity models. Since each policy is trained for a specific day's activity model, presenting trials drawn from a different day model to the system creates the effect of drastic deviation from original schedules. In the first experiment, both scheduled, and unscheduled trials are presented to the system for evaluation. Overall performance for a single learned control policy is measured as an average over the trials of the same type (scheduled vs. unscheduled). As with the previous results, a series of cost models (corresponding to different α's) were used for training in each case. The results are shown in Figure 7. Although performance is consistently made worse by the introduction of the unscheduled activities, the control policies in the latter case still demonstrate a clear trade-off between power consumption and the number of missed events that is controlled by the selection of a particular value of α. This shows that the system is able to learn a policy that handles unscheduled activities robustly.

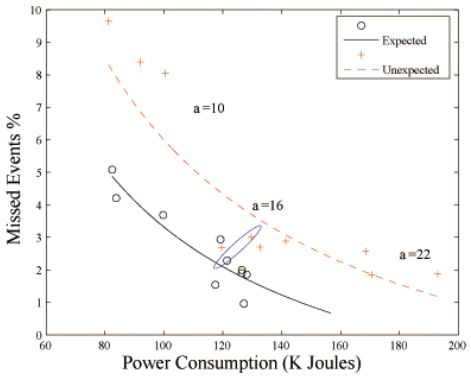


Figure 7. Learned policy performance with respect to scheduled and unscheduled activities. In general, the learned policies performed better when the user activities follows the schedule under which they were trained. The ellipse highlights the performances for α = 16 for both cases.

In designing context-sensitive devices (and in particular those that are calendar-sensitive), some users are expected to adhere closely to their scheduled activities, whereas others are expected to demonstrate a significant amount of variability in their arrival times to some activities. It is critical that any approach to learning context-sensitive behavior be robust to this range of

users. Here, the latter users are modeled with the user action uncertainty parameter such that more different random idle intervals between user events can be generated. Experiences are drawn from this model for both training and test trials. Evaluations show (Figure 8) that as randomness of the activities increases, the performance of the learned policy (performance trend with respect to α (the ALL case) slightly dropped. The drop is more apparent toward the power saving end. However, the important thing is that the trend of performance trade-off is with respect to the corresponding α value is retained. This demonstrates the robustness of the learned policy obtained using the current state representation.

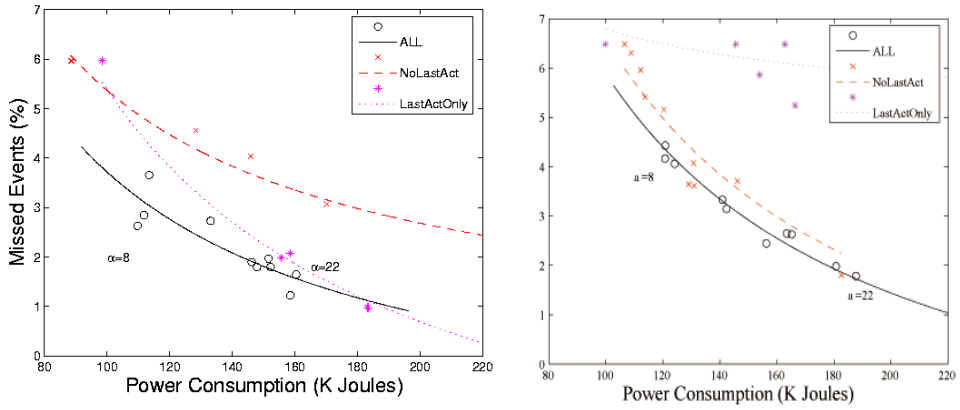


Figure 8. Sensitivity analysis. Performance of learned policies when the user action uncertainty parameter is varied ((a) σ = 0.5 and (b) σ = 4). The solid line represents the policy learned using full state representation. The long dotted line is the policy learned using a state representation that includes recent activity only, and the short dotted line is one that only excludes the recent activity.

Finally, the sensitivity of the system is examined with respect to the chosen state representation. This is important since choosing the appropriate state variable (where a feature or a set of features is used to encode a variable) to comprise our state representation is crucial to the performance of the learning system. To analyze the impact of state variables (e.g. calendar or time) in the performance of the resulting policy, each should be removed individually. If the variable has significance, then a drop in performance would be expected. Results (Figure 8(a)) show that when the user's action is more predictable (i.e. low variance), the recent activity state variable has a strong correlation with the user event patterns, and therefore significantly impact the performance of the system if removed. For situations when the user's action is highly variable (subject to change), an opposite trend is shown Figure 8(b): there is essentially no difference between the full representation and the one that excludes recent activity, but a clear disadvantage by the representation that only captures recent activity. From these two experiments it can be concluded that it is important to use the full representation in order to capture the range of users.

5. CONCLUSIONS

This paper presents a Reinforcement Learning approach to multi-sensor fusion problems with conflicting objectives. An example application of the approach is given in the domain of adaptive power management for mobile devices. Experimental results show that (1) the system can learn to translate multiple raw sensory inputs into appropriate sequence of system actions, optimizing on

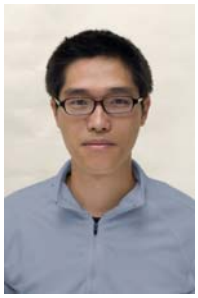
two conflicting goals; (2) the learned policy can outperform the user-defined policies; (3) by combining multiple sensor information, the learned policy handles unscheduled events reliably.

REFERENCES

1. E. Issac et al., Hubbub, "A sound-enhanced mobile instant messenger that supports awareness and opportunistic interactions," 2002.
2. R.H. Katz, B. Raman and A. D. Joseph, Universal inbox: "Providing extensible personal mobility and service mobility in an integrated communication network," In Proc. of the Workshop on Mobile Computing Systems and Applications (WMSCA'00), 2000.
3. P. Bahl and V.N. Padmanabhan. Radar, "An in-building RF-based user location and tracking system," In Proceedings of the IEEE Infocom 2000, 2000.
4. B. Davison and Hirsch, "Predicting sequences of user actions. In Workshop on Predicting the Future," AI Approaches to Time-Series Analysis, pp. 5-12.
5. R. Want, E.D. Ynatt, M. Back and Frederick, "W4: Real-time system for detection and tracking people in 2.5d," In Proceedings of European Conference on Computer Vision, Banff, Canada, pp. 877-892.
6. A. Krause et al., "Unsupervised, dynamic identification of physiological and activity context in wearable computing," In Proceedings of the 7th International Symposium on Wearable Computing, New Y, al," Agile application-aware adaptation for mobility," In Sixteen ACM Symposium on Operating Systems Principles, Saint Malo, France, 1997.
7. M. Roussopoulos et al. "Person-level routing in the mobile people architecture," In Proceedings of the USENIX Sym.le people architecture. In ACM Mobile Computing and Communications Review (MC2R), 1999.
8. R. Want et al. "The active badge location system," In ACM Transactions on Information Systems, 1992.
9. S. Thrun et al., "Robust Monte Carlo localization for mobile robots," In Artificial Intelligence, 2000.
10. N. B. Privantha et al., "The cricket location-support system," In Proceedings of the Sixth Annual ACM International Conference on Mobile Computing and Networking, Boston, pp. 32-43.
11. H. Chalupsky et al., "Electric elves: Applying agent technology to support human organizations," In Proceedings of IAAI-2001, Seattle, WA, 2001.
12. J. Hightower and G. Borriello. "Location systems for ubiquitous computing," In IEEE Computer, pp. 57-66, 2001.
13. N. Kern and B. Schiele, "Context-aware notification for wearable computing." In Proceedings of the 7th International Symposium on Wearable Computing, New York, USA, 2003.
14. H. Lieberman. Letizia, "An agent that assists web browsing," In Chris S. Mellish, editor, Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Quebec, Canada, 1995. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, pp. 924-929.
15. N. Marmasse, commotion: "A context-aware communication system," In Proceedings of CHI 99, pp. 157-171, 1999.
16. S. Mohapatra and N. Venkatasubramanian. Parm, "Power-aware reconfigurable middleware," In ICDCS 2003:23.
17. T. Pering and T. Burd. "Voltage scheduling on the lparm microprocessor system," In Proceedings of the 2000 International Symposium on Low Power Electronics and Design, Volume 3, 30448, 2000.

18. Y. Qi and R. W. Picard, "Context-sensitive Bayesian classifiers and application to mouse pressure pattern classification," In Proceedings of International Conference on Pattern Recognition, 2002.
19. R. Sutton and A. Barto, "Reinforcement Learning," MIT Press, Cambridge, Massachusetts, 1998.

ABOUT THE AUTHORS



S. Ou received his M.S. degrees in Computer Science in 2006 from the University of Massachusetts Amherst. He is currently working towards a Ph.D. degree under the guidance of Professor Roderic Grupen, in the Laboratory for Perceptual Robotics, at the University of Massachusetts Amherst. His research reflects a broad interest in computer science, including human-robot interaction, knowledge discovery, mobile devices, distributed sensor network, assistive technologies, and computer vision.

A. H. Fagg received his M.S. and Ph.D. degrees in Computer Science from University of Southern California in 1991 and 1996, respectively. He joined the Computer Science faculty at the University of Oklahoma in 2004, where he is currently Associate Professor. His research focuses on the relationships between biological systems and machines. In this area of symbiotic computing, he studies the interaction of humans with machines, machines as models of how biological systems represent and learn motor and cognitive skills, and primates as inspiration for new robot control and learning techniques.

P. Shenoy is an Associate Professor of Computer Science at University of Massachusetts Amherst. His research interests lie in operating and distributed systems, sensor networks, Internet systems and multimedia. He heads the Laboratory for Advanced Systems Software at University of Massachusetts Amherst, and leads the research in building systems and understanding them through analysis and experimentation.

L. Chen is a Professor of College of Information at South China Agricultural University. She heads the Laboratory for Information Systems at South China Agricultural University. Her research interests are data mining, machine learning methods and embedded systems.