Social User Agents for Dynamic Access to Wireless Networks

P. Faratin and **G. Lee** and **J. Wroclawski**
Laboratory for Computer Science
M.I.T
Cambridge, 02139, USA
{peyman,gjl,jtw}@mit.edu

S. Parsons
Department of Computer and Information Science
Brooklyn College, City University of New York
NY,11210, USA
parsons@sci.brooklyn.cuny.edu

Abstract

We present a dynamic and adaptive decision model for an autonomous user agent whose task is to dynamically negotiate and procure wireless access for a mobile user. A user is assumed to have cognitive and motivational costs associated to providing subjective preference information to the agent. Therefore the task of the personal agent is to dynamically model the user, update its knowledge of a market of wireless service providers and select service providers that satisfies the user's expected preferences based on minimal, or missing, information that is derived from a simple user interface. In this paper we show how this user modeling problem can be represented as a Markov Decision Process. Adaptive reinforcement learning solutions are then evaluated for two subclasses of tractable MDPs via simulations of some representative user models.

Introduction

One of the goals of our research is the design of autonomous single and multi agent decision mechanisms for non-routine, dynamic and cognitively costly decision contexts or environments. The domain we consider is the future wireless networks where, in contrast to the vertically integrated structure of the current Internet, more complex network services can be represented and traded amongst multiple buyers and sellers in a dynamic and open market. However, compared to the decision problem of a stationary user in a LAN environment, a nomadic user in wireless network is required to solve an (access and connection) decision (or trading) problem for each possible location she visits. Furthermore, the decisions need to be made over complex goods (or network services) that may be multi-dimensional and described and represented in terms of the ontology at the network level. Such cognitive costs associated with these task-environments can be usefully reduced through a personal agent. Under such a scheme a user's high level requests are then mapped to a personal agent, referred to as the Personal Router (PR), that dynamically selects access to different wireless base-stations. However, central to this autonomous decision making is an information model of the user, where in decision contexts is equivalent to specification of user preferences over possible outcomes. Since the task-environment of the user is cognitively costly, we do not adopt the static, off-line and single-shot classical preference elicitation solutions but instead seek a mechanism that is *dynamic, on-line, incremental* and iteratively improves the model of the user in an adaptive manner with *minimal information*. That is, we seek an adaptive on-line solution that spreads the costs associated to one-off and off-line elicitation of the user information model over time. Central to this is an appropriate user-agent interface that provides useful information for on-line optimization task of the agent.

More specifically, we frame this Autonomous User Modeling (AUM) problem task as an adaptive (feedback) control problem where a personal agent (a controller) takes adaptive actions (control signals) in an environment (a control system) to achieve its goals. Furthermore, as opposed to the classic approaches where a *single* optimal decision is made *after* eliciting the *complete* preference structure of the user (Keeney & Raiffa 1976), the goal of the agent is to find an optimal *strategy* over sequences of decisions over time, where the sequence is composed of elicitation followed by decision making processes. In such a user-agent coupling, initially *user perceived* sub-optimal agent decisions can be improved over time with additional information acquired by elicitation via a user-agent interface. A sequential decision making approach is adopted because the dynamic decision environment creates a barrier for obtaining a priori information about the user and the user cannot be engaged in a costly elicitation process similar to traditional solutions.

Generally, we are interested in cost and benefit trade-offs involved between perfect information for a single optimal decision mechanism against imperfect information and sub-optimal but iterative and adaptive mechanisms. In this paper we propose a state-based model that approaches the latter mechanism.

The paper is organized as follows. A general description of the PR problem is briefly described in the first section. We then present a formal model of the service selection problem. Next we show how the *full* problem description can be computationally represented within a Markov Decision Process. This is followed by two simpler MDP models of the PR problem that are motivated by the intractability of the fully dimensioned model. Next we review how reinforce-

ment learning algorithms can be used to solve the agent's action selection problem. The behaviours of these algorithms are then evaluated in a set of user model simulations. Finally, representative related work is reviewed and followed by our conclusions together with the directions of future research.

The Personal Router

Optimal decision making requires access to a well defined preference structure of the decision maker that give a meaningful ordering to the set of possible outcomes. Traditionally, solution to such user modeling problems are formulated within the classical utility analysis framework, such as conjoint analysis (Keeney & Raiffa 1976; Urban & Hauser 1980).

However, although useful for most static and tractable domains, decision analysis techniques are inappropriate in wireless network due to the complexity in and dynamicity of the user's context (Faratin *et al.* 2002). The user context is defined by: a) the user's goals (or activities—e.g. arranging a meeting, downloading music), b) the class of application the user is currently running in order to achieve her goals (e.g. reading and sending emails, file transfer), c) her urgency in using the service and d) her location (e.g nomadic or stationary). This context is highly complex not only because a user may have multiple concurrent goals/activities but also because different elements of the user context (goals, locations, running applications, etc.) may change at different rates. Indeed, we claim that such complexities necessitate a personal agent-based approach because on-line agents can learn user's preferences over time rather than requiring users to participate in an unrealistic and expensive elicitation process. For example, the set of service choices (or outcomes) can change dynamically when current network connection(s) instantly become unreachable as mobile users change locations. Alternatively, preferences themselves can also be dynamic where different services may be needed as users dynamically change and begin different tasks. Furthermore, additional barriers exists for traditional preference modeling approaches than the complexity of the user context. For example, due to cognitive costs and nomadic nature of wireless access users may be reluctant to engage in costly communications over their preferences, specially if the elicitation space is combinatorially large. In the worst case users may be ignorant of their preferences, a real possibility with network services which the user, unlike common goods such as bread or milk, has little or no experience of in order to form a preference over. Indeed, intangibility of network services may necessitate a trial period before the user can begin to form preferences. Furthermore, there exists an inherent variability in the network itself resulting in uncertainties by both the buyers and the sellers of a service as to the guarantees that can be made over the quality of a service (QoS). To overcome this uncertainty users are therefore given a user interface to manipulate service features as free variables via *better* and *cheaper* buttons on the PR respectively. The assumption we make is that user will choose better or cheaper services if the current selected service is either of poor quality or high price respectively. This process of interaction with the PR may continue until the PR learns to select a service that satisfies the user's current tasks and goals. Note, because of relatively low wireless service prices we assume users are tolerant to suboptimal decisions in service selection because the (monetary) cost of decision errors are low.

Whereas the action space of the user can be defined by the agent designer (through the design of user interfaces), the actions of the PR are not under the control of the agent designer but are instead constrained by what we call the strategy space of the current market mechanism (Rosenschein & Zlotkin 1994). That is, actions of the agent are assumed to be defined exogenously by the rules of some interaction mechanism (e.g. an auction mechanism). We envisage two possible mechanisms in wireless networks (see (Clark & Wroclawski 2000)): a "take-it-or-leave-it" or an iterative negotiation mechanism. In the former mechanism the production capability of a service provider is assumed to be fixed and static. Therefore, the rules of such a market only permit the agents to select from a discrete and finite number of profiles on offer. Conversely, in a negotiation mechanism the provider is assumed to be capable of (and has the incentives to) generating infinitely divisible goods, which in this domain is bandwidth. Therefore, in such a market the agents actions are defined as requests for profiles that optimizes their demand profile. The rules of an iterative negotiation mechanism may in turn permit sellers to respond with actions that optimizes their supply function.

Representing the Problem as a Markov Decision Process

In this section we present the PR problem within the Markov Decision Process (MDP) modeling framework (Kaelbling, Littman, & Moore 1996; Boutilier, Dean, & Hanks 1999).

Problem Elements

We condition each service selection process instance on the current context of the user. As mentioned above a user context includes the current running application set, the time deadline and the location of a user for current goal. We let C represent the set of all possible contexts and $C^g \subseteq C$ be the set of contexts that are partitioned by the user goal g. An element $c \in C$ is composed of the tuple $c = \langle \beta, \gamma, \delta \rangle$, where β, γ and δ represent the set of running applications, user deadlines and locations respectively. Then, a particular user context $c^g \in C^g$, partitioned by the goal g, is defined by the tuple $c^g = \langle \beta^g, \gamma^g, \delta \rangle$, where β^g, γ^g and δ represent the set of running applications compatible with current goal g, the user deadline for current goal g and the concrete location of the user respectively. The location of a user at any instance of time is represented by both the physical location as well as the temporal location.

Next we let $\mathbf{P}$ represent the set of all possible service profiles, where each element of this set $P \in \mathbf{P}$ is composed of n features f_i, $P = (f_1, \ldots, f_n)$. Because service profiles available at any time change due to both user roaming (given a nomadic user) and changes in service offerings (given service providers' uncertainty in the state of the network) then we assume the (physical and temporal) location

δ of a user partitions the set of possible service profiles available. Therefore we let $P^{\delta} \in \mathbf{P}$ represent the subset of possible service profiles available to the user in location δ.

Next let the set of all user preferences be given by $\mathbf{U}$. We then let each element of this set, $U \in \mathbf{U}$, represent a unique orderings over all the possible pairs of service profiles $\mathbf{P}$, or $U = (P_i \succ P_j, \ldots, P_{l-1} \succ P_l)$[1] for all combination of l profiles. Similarly, the current user context and goal partition the set of all possible preference orderings, or $U^{c^g} \subseteq \mathbf{U}$.

The ordering generated by U can then be captured by a utility function u such that:

$$u(P_i) > u(P_j) \quad \text{iff} \quad P_i \succ P_j \tag{1}$$

The MDP Model

An MDP is a directed acyclic graph composed of a set of nodes and links that represent the system states $\mathbf{S}$ and the probabilistic transitions $\mathbf{L}$ amongst them respectively (Bertsekas 1987). Each system state $S \in \mathbf{S}$ is specified by a set of variables that completely describe the states of the problem. The value of each state variable is either discrete or continuous but with the constraint that each state's variable values be unique. In our problem each system state $S \in \mathbf{S}$ is fully described by the combination of: a) the user context ($c^g = \langle \beta^g, \gamma^g, \delta \rangle$) for goal g, b) the set of profiles available in the current location (P^{δ}) and c) the user interaction with the PR, which we will represent by the variable I.

Therefore, a complete description of *a* system state at time t is represented by $S^t = (\beta^g, \gamma^g, t, loc^g, P, I)$, where $\beta^g, \gamma^g, t, loc^g$ represent the context of the user for goal g. Note that for reasons to be given below we disaggregate δ, the user location and time, to two state variables t and loc^g, the location of the user in temporal and physical space respectively. We can also specify user goals g in a similar manner by a subset of system states $S^g \subseteq \mathbf{S}$.

The other element of a MDP is the set of possible actions $\mathbf{A}$. Actions by either the user, the PR or both will then results in a state transition, that change the values of the state variables, to another state in the set of all possible states $\mathbf{S}$. In an MDP these transitions are represented by links $\mathbf{L}$ between nodes that represent the transition of a system state from one configuration to another after performing some action. Additionally, each link has an associated value that represents the cost of the action. In our problem the set of actions A available to the user u are defined by the set $A^u = \{\Delta^{loc}, \Delta^{app}, \Delta^I, \phi\}$, representing changes in the user location, set of running applications, service quality and/or price demand and no action respectively.[2] The consequences of user actions are changes in values of state variables $\beta^g, \gamma^g, t, loc^g, P, I$; that is, changes in either: a) the user context (changes in running applications, the time deadlines for connections, the current time, the user location and/or price/quality demands, observed by interaction with the PR via better and cheaper responses) or b) the set of currently available profiles or the combination of the state variables.

The set of actions A available to the PR are defined by the set $A^{PR} = \{\Delta^{P_i \rightarrow P_j}, \phi\}$ representing PR dropping service profile i and selecting j and no action respectively. The consequence of a PR action is a change in the likelihood of future user interaction I, where decreasing likelihoods of user interactions is more preferred.

Additionally, in an MDP the transitions between states are probabilistic. Therefore there exists a probability distribution $Pr_{a_j}(S_k|S_j)$ over each action a_j reaching a state k from state j.

Finally, we can compute the utility of a service profile i in context c for goal g (or $u^{c^g}(P_i)$ as the utility of being in a unique state whose state variables $(\beta^g, \gamma^g, t, loc^g, P, I)$ have values that correspond to service i in context $c = \{\beta^g, \gamma^g, t, loc^g\}$. The utility of this corresponding state, say state m, is then referred to as $U(S_m)$.

Aggregated Decision Models

In general MDPs suffer from the "curse of dimensionality", where the state-space grows exponentially in the size of the variables (Bellman 1957). As a result it is computationally impractical to consider learning the highly expressive model given above. Thus, there exists some natural *computational* bounds on the granuality of the user model that can be represented and reasoned with. The strategy we adopt to address this problem is to incrementally search for a computationally tractable MDP model that *increasingly* approaches some acceptable level of expressiveness, derived through ecological experiments. Different expressive-computational tradeoff regimes can then be constructed that range from single state MDPs, with coarse state signals, to richer state signal MDP, described above, each with relatively different expressive and predictive power. The natural expectation, verifiable through ecological experiments, is that richer models result in more adequate behaviours because they can form better associations of states to actions (or policies) than simpler non-associative models. In the remainder of the paper we describe the first step in this strategy where the complex state-space of the above MDP described is "collapsed", through disjunction of all of the states, to either a single state signal or a slightly more complex state signal consisting of service profiles. We show how the former reduced problem is equivalent to the k armed bandit problem and review heuristic solution methods for both classes of problems based on reinforcement learning. Finally, we evaluate the adequacy of the model through simulations.

Single State PR—a Bandit Problem

The computationally simplest model of the PR problem is to cast the agent action selection problem as a k armed bandit problem. Bandit problems have been extensively stud-

[1] The operator $\succ$ is a binary preference relation that gives an ordering. For example, $A \succ B$ iff A is preferred to B.

[2] Note, that since time is an element of the state description then the system state always changes in-spite of no action by either the user or the PR or both. Furthermore, the granularity of time is likely to be some non-linear function of user satisfaction, where for example time is finely grained when users are not satisfied with the service and crudely grained when they are satisfied. However, the granularity of time is left unspecified in our model.

ied in statistical and mathematical literatures with application to medical diagnosis and sequential decision making in general. In this class of problems an agent has a choice of pulling one arm of a k-armed bandit machine at each time step. When the arm i of machine is pulled the machine pays off 1 or 0 according to some underlying probability p_i, where payoffs are independent events and unknown to the agent. The game often has a finite horizon where the agent is permitted to pull h pulls. The goal of the agent is to select a policy that maximize some function of the total expected payoffs R_t, typically given by $R_t = E(\sum_{t=0}^{h} r_t)$, where r_t is the payoff or reward at time t.

Similarly, we model the PR problem as a single state infinite horizon MDP where at each discrete time the agent takes an action (pulls an arm i), receives a reward from the user for its action and returns to the same state. Generally, the size of k, or the number of the arms of the bandit machine, is determined by the rules of the market mechanism the agent is trading. Thus in a negotiation mechanism the set of agent actions, or agent's strategy space, at each state is given by $A^{PR} = \{LBW, HBW, LP, HP, \phi\}$. That is, under such a mechanism the agent is playing $k = 5$ one-armed bandit machines. Furthermore, since we are interested in *continual* learning of user preferences the goal of the agent is to maximize its total rewards over long-run, or infinite horizon, of the game. The reward model in turn is constructed from the actions of the user *with the agent* through the interface and not the environment. Thus, we only consider a subset of user actions $A^u = \{\Delta^I, \phi\}$, where Δ^I is changes in price and bandwidth demands (or "cheaper" or "better" button presses respectively) by the user and ϕ is no action. We map these user actions to binary agent reward values using the simple binary rule of $r_t = +1$ if ϕ else $r_t = -1$, representing positive rewards for lack of user intervention. We model the user's preferences for different agent actions by a reward probability distribution with mean $Q^*(a_i)$ for each action a_i. Finally, if this probability distribution is constant over time then we say that the user's preferences is stationary. Conversely, if the distribution of the bandit changes over time then the user's preferences is said to be non-stationary. We model this with three parameters: (θ, f, η), the period between changes, the probability of change and magnitude of change in $Q^*(a_i)$ respectively.

There exist a number of solutions for solving the optimal policy for the bandit problems, including dynamic-programming, Gittins allocation indices and learning automata (Kaelbling, Littman, & Moore 1996; Sutton & Barto 2002). However, although optimal and instructive these methods are known not to scale well to complex problems (Kaelbling, Littman, & Moore 1996). Because our goal is to incrementally increase the complexity of the problems, and also because the "forgiveness" of the user to the suboptimal decisions, we instead concentrate on heuristic action selection techniques that, although are not provably optimal, are nonetheless tractable and approximate optimal solutions. Furthermore, as mentioned above, optimal techniques compute optimal policies *given* a model (model of the process that generate $Q^*(a_i)$). In the absence of this information the agent must form estimates over and update the value of each action. Therefore, the problem of action selection requires both a learning phase followed by action selection phase.

One popular method for updating the values of $Q_{k+1}(a_i)$, estimates for $Q^*(a_i)$ for action i after k rewards is the exponential recency-weighted average:

$$Q_{k+1}(a_i) = Q_k(a_i) + \alpha_k(a_i)[r_{k+1}(a_i) - Q_k(a_i)] \quad (2)$$

where $0 < \alpha_k(a_i) \leq 1$ is the step-size, or learning, parameter for action a_i after k selection. If $\alpha_k(a_i) = 1/k$ then the learning rate varies at each time step. Under this condition the update rule 2 implements a sample average method (Sutton & Barto 2002).

The next step in solving the bandit problem is to select an action given an estimate of value of actions. We compare the behaviour of three action selection strategies given $Q_{k+1}(a_i)$: greedy, $\epsilon-$greedy and softmax. The first strategy exploits the current agent knowledge by selecting that action with the highest current value estimate: $a_i^* = arg\ max_a Q_t(a_i)$. Conversely, as the horizon of interaction increases then it may be more beneficial to explore the action space since higher valued longer term rewards may be biased against by lower valued shorter term rewards (expressed as non-linearity in the optimization objective function). Exploration, or probability of selecting action a at time t, $(P_t(a_i)$ may be at some constant rate, ϵ or given by a Gibbs, or Boltzmann, distribution:

$$P_t(a_i) = \frac{e^{Q_t(a)/T}}{\sum_{a' \in A}^{n} e^{Q_t(a')/T}} \quad (3)$$

where the temperature T cools at rate μ with time t according to the equation $T_t = T_0 (1 - \mu)^t$. Action selection strategies with constant and variable values of ϵ are referred to as ϵ-greedy and softmax strategies respectively.

Learning the best action can also be achieved not by maintaining estimates of action value but rather an overall reward level, called the reference reward, that can be used as a decision criteria. Techniques based on this method are known as Reinforcement Comparison (RC) methods, precursors to actor-critic methods (Sutton & Barto 2002). In RC a separate measure of action preference for each action at t play of the bandit, $\rho_t(a)$, is kept that are used to determine the action-selection probabilities according to softmax rule 3. The preferences are updated as follows. After each play of the bandit the preference for the action selected on that play, a_t, is incremented by the error signal between the reward r_t and the reference reward $\bar{r}_t$, by:

$$\rho_{t+1}(a_t) = p_t(a_t) + \alpha[r_t - \bar{r}_t] \quad (4)$$

where α is a positive step-size parameter. Unlike action-value updates, the reference reward is an incremental average of all recently recieved rewards independently of the action taken:

$$\bar{r}_{t+1} = \bar{r}_t + \alpha[r_t - \bar{r}] \quad (5)$$

where $0 < \alpha \leq 1$ is again some learning rate.

Finally, in Pursuit methods (PM) both action-value and action-preferences are maintained where the action preferences "pursue" the action that is greedy according to the current action-value estimate (Sutton & Barto 2002). If $\pi_t(a)$ represents the probability of selecting action a at time t, determined through softmax, and $a^*_{t+1} = arg\ max_a Q_{t+1}(a)$ represents the greedy action at play $t+1$ then the probability of selecting $a_{t+1} = a^*_{t+1}$ is incremented by a fraction β toward 1:

$$\pi_{t+1}(a^*_{t+1}) = \pi_t(a^*_{t+1}) + \alpha[1 - \pi_t(a^*_{t+1})] \quad (6)$$

Then the probabilities of selecting other actions are decremented towards zero: $\pi_{t+1}(a) = \pi_t(a) + \alpha[0 - \pi_t(a)]$ for all $a \neq a^*_{t+1}$.

Multi-State PR Problem

As the next step, we consider a multistate model of the PR problem. In the single state bandit problem described above, the expected reward received depends solely upon the current agent action. In our multistate model, the amount of reward is based upon both the agent action and the current service profile, allowing us to model the agent's attempts to learn the user's utility function over the space of service profiles.

Formally, we model the PR as a deterministic MDP with states $s \subset P^\delta$, the set of currently available service profiles. In order to model the trade-off between quality and cost, we define a service profile as a vector of two features (b, h), where b represents bandwidth and h cost. For simplicity, we constrain quality and cost to the set of nonnegative integers.

The set of possible agent actions A^{PR} remains the same as before, but in the multistate model the current service profile may change after every agent action. This transition function is deterministic and assumes that the agent gets the service profile it requests if it is available. For instance, If the state s at time t is (b_t, h_t), the agent selects action LBW, and $(b_t - 1, h_t) \in P^\delta$, then $s_{t+1} = (b_t - 1, h_t)$. If the desired service profile is not in P^δ, then the state remains unchanged. By limiting agent actions to the fixed set A^{PR}, we reduce the complexity of the agent while still enabling the exploration of the full set of services available.

As in the single state model, at each time step the agent receives a reward $r \in \{1, -1\}$ depending on the user action A^u. The user action probability distribution $p(s_i)$ is based on the utility $U(s_i)$ of the current service profile s_i. We model user utility with the linear function $U(q, h) = w_q q + w_h h$, where $w_q > 0$ and $w_h < 0$, expressing a desire for high bandwidth and low cost. This utility function is easy to compute while still allowing the description of a wide range of user preferences.

This multistate model is a natural extension of the single state MDP model described earlier. Though the service profiles and user utility functions in this model have been chosen for ease of computation, the multistate model provides a substantially more detailed view of the interactions between the agent and its environment, capturing the relationship between user utility and user actions as well as the effect of agent actions on the current service profile.

Due to the multistate nature of this model, however, the approaches for solving the single state bandit problem cannot accurately learn the optimal agent actions. The bandit solutions can learn which action yields the greatest reward for any given state, but in order to maximize the total return the agent must take into account the value of other states as well.

Possible solutions for this problem include dynamic programming, Monte Carlo methods, and TD learning (Sutton & Barto 2002). Dynamic programming is not appropriate because the rewards at each state are not known a priori. Monte Carlo approaches are inadequate because they learn policies off-line; the non-episodic nature of the PR problem requires an on-line solution that can learn a policy as it interacts with the user. In contrast to the other two approaches, TD learning works well for non-episodic tasks with unknown rewards. Of the TD(λ) solutions, we choose to examine the 1-step backup methods as an initial approach.

Many 1-step backup TD control methods exist, including Sarsa, Q-learning, and actor-critic methods. Q-learning is an off-policy method that learns the optimal policy regardless of the policy used. In contrast to Q-learning, Sarsa is an on-policy method that takes the current policy into account in its action-value estimates. It operates according to the following update rule:

$$\begin{aligned} Q(s_t, a_t) \quad \leftarrow \quad & (1 - \alpha)\, Q(s_t, a_t) \\ & + \alpha \left[r_{t+1} + \rho Q(s_{t+1}, a_{t+1}) \right] \end{aligned} \quad (7)$$

where $s_t \in P^\delta$ is the current state, s_{t+1} is the next state, $a_t \in A^{PR}$ is the current agent action, a_{t+1} is the next agent action according to the policy, α is a constant weight, r_{t+1} is the reward received in the next time step, and ρ is the discount factor.

Both learning methods work well with a wide range of policies, including ϵ-greedy methods and Gibbs softmax algorithms. The advantage of Sarsa is that its action-value estimates accurately reflect the action values of the policy used, whereas Q-learning always learns estimates for the optimal policy. If the policy converges to the greedy policy, however, then Sarsa will eventually learn the optimal policy.

Simulations

Single State Bandit Model

In order to demonstrate the agent's ability to learn user preferences using reinforcement learning, we simulated the single state bandit solutions described in section . To illustrate a wide range of different approaches, we have selected an ϵ-greedy method, two Gibbs softmax methods with initial temperature $T_0 = 10$ and cooling rates $\mu = \{0.05, 0.01\}$, a pursuit method, and a reinforcement comparison approach. All of these implemented exponential averaging with $\alpha = 0.1$ except for the $\mu = 0.01$ softmax which uses $\alpha = 1/k$, giving equal weight to all samples. For clarity, some plots have been smoothed using cubic spline interpolation.

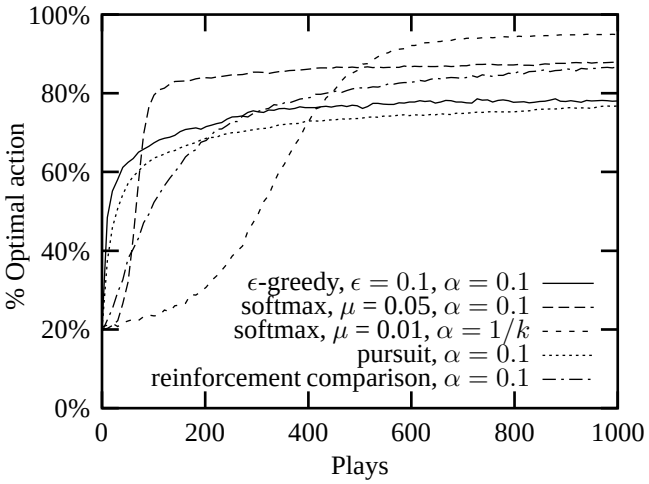

Figure 1: Stationary Bandit User Model

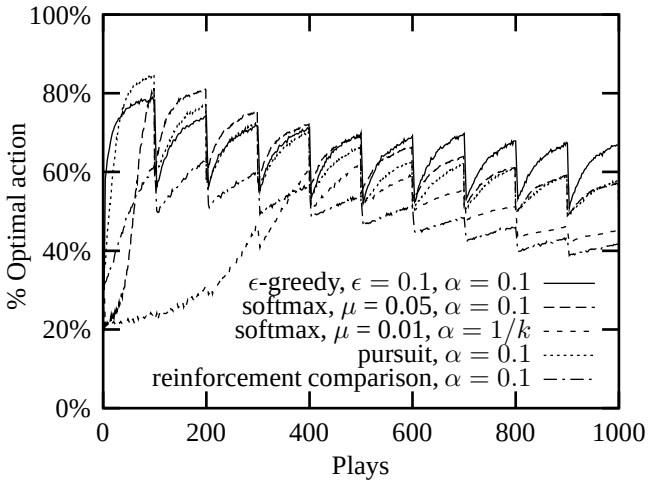

Figure 2: Periodic User Model, $\theta = 100$, $f = 1$, $\eta = 0.4$

Figure 1 shows the observed results for stationary user preferences. The plot shows the average of 10,000 tasks, each consisting of 1000 plays, or time steps. At the start of each task, the reward probabilities $Q^*(a_i)$ are initialized with random values between -1 and 1 for each of the five agent actions $a_i \in A^{PR}$. The plot shows the percentage of optimal actions for each play, where the optimal action is defined as the action providing the greatest expected reward.

The figure shows how the choice of learning method affects the speed at which the optimal action is learned as well as the long term average reward. The ϵ-greedy and pursuit methods improve very rapidly initially, but soon reach asymptotic levels at approximately 80%. In contrast, the Gibbs softmax method with $\mu = 0.01$ makes poor selections in the first few hundred plays while it explores, but eventually selects the optimal action 95% of the time. The other algorithms achieve optimality at various speeds in between these two extremes. In summary, the data shows the trade-off involved between exploration and exploitation; the more time the agent spends exploring, the better the policy it can learn in the long run.

Figure 2 and 3 show the observed behavior for nonstationary preferences that change occasionally. In these simulations, the agent begins with knowledge of the user's preferences and must relearn them as they change. Figure 2 shows a periodic user model in which the user's preferences randomly increase or decrease by $\eta = 0.4$ every 100 plays. The periodic user model clearly illustrates the behavior of the learning models when user preferences change regularly and infrequently. The ϵ-greedy method has the best performance because it can rapidly relearn the new user preferences after every change. The Gibbs softmax with $\alpha = 1/k$ performs most poorly because it uses a sample average method and cannot adapt to change.

In figure 3, user preference changes follow a Poisson distribution: at each play, there is a $f = 0.1$ probability of randomly increasing or decreasing $Q^*(a_i)$ for each action by a

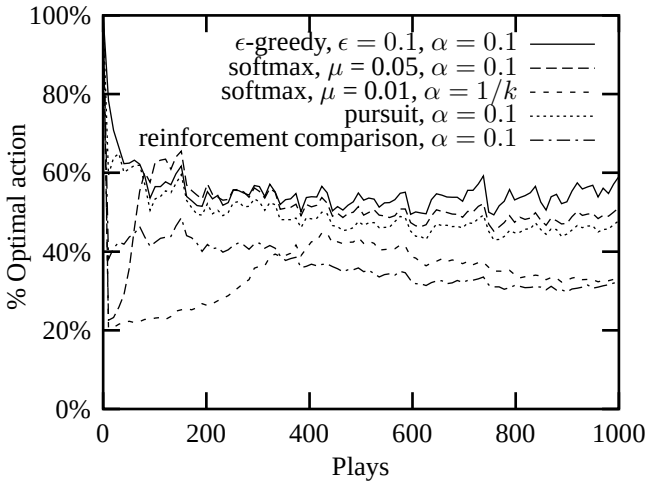

Figure 3: Poisson User Model, $\theta = 1$, $f = 0.1$, $\eta = 0.2$

constant magnitude of $\eta = 0.2$ (see section). In the steady state, the ϵ-greedy method performs the best, selecting the optimal action about 58% of the time. The nonstationary data shows that when the user's preferences change the agent must use recent observations in its estimates and does not have much time to explore. With nonstationary user preferences, the ϵ-greedy approach quickly finds a good action and exploits it, but the reinforcement comparison method spends too much time exploring. The $\alpha = 1/k$ Gibbs softmax algorithm yielded the best results in the stationary case, but performs poorly in both nonstationary cases because it fails to discount old observations.

One would expect that as the rate of user preference change increases it becomes more difficult for the agent to learn their preferences. Figure 4 confirms this expectation by showing the effect of f (frequency of change—see sec-

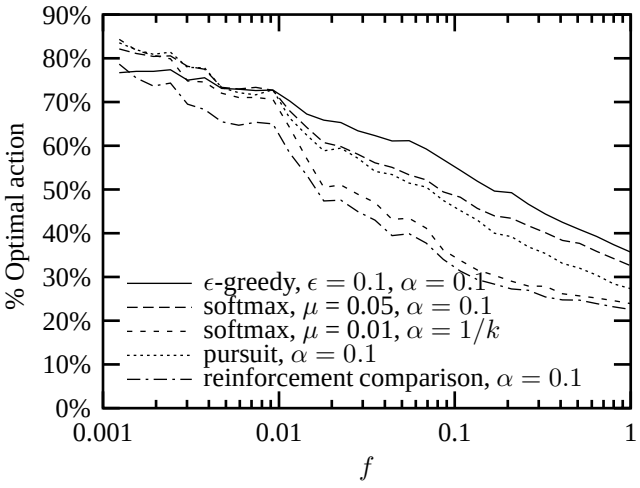

Figure 4: Poisson User Model, $\eta = 0.2$

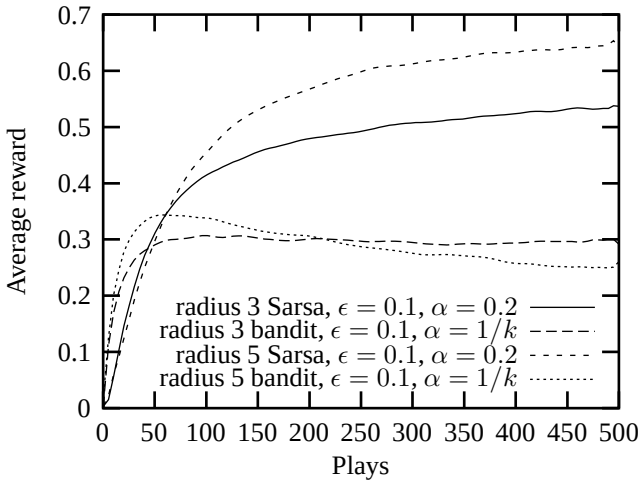

Figure 5: Multistate Stationary User Model

tion) on performance. For chosen values of f between 0 and 1, we simulated each agent model for 5000 tasks and 1000 plays using $\eta = 0.2$. For those values of f, the plot shows the average percentage of optimal actions over the last 300 plays. At the left when preferences change infrequently, it is similar to the stationary case: softmax with $\mu = 0.01$ performs the best while ϵ-greedy performs more poorly. As preferences change more frequently, the relative performance of the ϵ-greedy method improves while the accuracy of the the Gibbs softmax method degrades.

Multistate MDP Model

We have seen that in the single state case, the agent can learn user preferences for stationary and nonstationary user preference models. The reinforcement learning methods described in allow us to learn these preferences in multistate models as well. Figure 5 contrasts the performance of a Sarsa TD learning approach with a single state bandit method in a simulation over 10,000 tasks. In this simulation, the set of service profiles P^{δ} consists of all integer bandwidth/cost pairs (b, h) within a 3 unit radius of the initial service profile $s_0 = (5, 5)$. At the start of each task, the user utility function $U(q, h) = w_q q + w_h h$ is initialized randomly with $0 < w_q < 1$ and $-1 < w_h < 0$. The expected reward is given by the function $r(s_i) = 1 - \frac{2}{1+e^{-U(s_i)}}$. Though the ϵ-greedy bandit method learns the rewards for each action, it does not accurately compute the long term value as well as the Sarsa method. The advantage of TD learning becomes even more apparent as the state space increases; when the radius is 5, the Sarsa approach obtains 2.5 times the reward of the bandit method, illustrating the effectiveness of a TD learning approach over a bandit method in a multistate PR model.

Related Work

The MDP model presented above is similar to solutions such as fast polyhedral conjoint analysis that extend classical solutions to a sequential decision making paradigm (O. Toubia & Hauser 2002). However, motivated by solving the combinatorial size of the elicitation space, fast polyhedral techniques still require additional a priori information over the utility of elicitation question at each stage of decision making. We are, on the other hand, interested in dynamic decision contexts, such as dynamic access to wireless networks, where there is little or no a priori information and the user cannot be engaged in costly elicitation process similar to traditional solutions. Minimal user interaction during preference elicitation is also sought by Chajewska *et.al.* where the UM problem is viewed as a classification problem (Chajewska, Koller, & Parr 2000). A myopically optimal elicitation strategy is constructed to ask the single query with the greatest expected value of information with respect to a distribution of clusters of utility functions. The uncertainty over this distribution is then refined as users answer queries. However, the hillclimbing nature of myopic strategy can fail to ask appropriate questions because *future* values are neglected when determining the value of current questions. Boutilier's extension of the above model to a Partially Observable Markov Process (POMDP) (Boutilier 2002), implements a sequential decision problems with multistage lookahead. However, although also modeling the elicitation process as a sequential decision making problem this model assumes a priori belief model (although any arbitary model) for optimization.

Conclusions and Future Work

In this paper we described a user-modeling problem for the domain of wireless services. An agent, called a Personal Router, was proposed as a solution to this problem. We showed how the nature of the problem bounds the information set of the agent. We then presented a formal model of

the service selection problem and showed how it can be captured in an MDP representation. Heuristic solutions from reinforcement learning were then empirically evaluated for two simpler MDP models of the PR problem.

There are a number of future directions. Our first goal is to continue to empirically evaluate different learning algorithms for increasingly more complex MDPs. The performance of the resulting agents will then be evaluated for ecological validity in controlled user experiments, the outcomes of which will be used for further (re)design-simulation-user experiment development, followed by field testing. Finally, our longer term goal after achieving a satisfactory level of performance of the agent is to extend the single agent decision mechanism to multi-agent systems (MAS) . In particular, we envisage two direction of future MAS research. On the one hand, MAS mechanisms (such as distributed reputation, gossiping, epidemic and/or collaborative filtering mechanisms) can be useful information sources for parameters of the agent decision model. For example, some of the model updating mechanisms presented above are biased by their initial $Q_0(a)$ values meaning that the behaviour of model is dependent on the initial belief states that must be specified as a parameter of the model. In absence of domain expertise another strategy for deriving $Q_0(a)$ is to deduce its value from estimates of a group of other user agents. Finally, the single agent decision mechanism must be extended to represent and optimize over other the states and actions of rational decision makers (or service providers) that the agent interacts with during negotiation of services. Markov games have been proposed as one potential solution to this computational mechanism design problem.

Acknowledgments

This work is supported by the NSF grant ANI-0082503 on Protocols for Open Access Wireless Deployment.

References

Bellman, R. 1957. *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Bertsekas, D. 1987. *Dynamic Programming: Deterministic and Stochastic Models*. NY: Prentice-Hall.

Boutilier, G.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11:1–94.

Boutilier, G. 2002. A POMDP formulation of preference elicitation problems. In *Proceedings of American Association of Artificial Intelligence*, 239–246.

Chajewska, U.; Koller, D.; and Parr, R. 2000. Making rational decisions during adaptive utility elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 363–369. Austin, Texas: AAAI Press.

Clark, D., and Wroclawski, J. 2000. The personal router whitepaper. Technical report, Laboratory for Computer Science, MIT. http://ana.lcs.mit.edu/anaweb/papers.html.

Faratin, P.; Wroclawski, J.; Lee, G.; and Parsons, S. 2002. The personal router: An agent for wireless access. In *Proceedings of American Association of Artificial Intelligence Fall Symposium on Personal Agents, N. Falmouth, Massachusetts, US*, 13–21.

Kaelbling, L.; Littman, M.; and Moore, A. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.

Keeney, R. L., and Raiffa, H. 1976. *Decisions with Multiple Objectives*. New York: John Wiley and Sons.

O. Toubia, D. S., and Hauser, J. 2002. Fast polyhedral adaptive conjoint estimation. Technical report, MIT Sloan School of Management. http://mitsloan.mit.edu/facstaff/index.html.

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter*. Cambridge, USA: The MIT Press.

Sutton, R., and Barto, A. 2002. *Reinforcement Learning*. Cambridge, MA: MIT Press.

Urban, G. L., and Hauser, J. 1980. *Design and Marketing of new products*. New York: Prentice-Hall.