Adapting Interaction Environments to Diverse Users through Online Action Set Selection

Paper No. 1354

Abstract

Interactive interfaces are a common feature of many systems ranging from field robotics to video games. In most applications, these interfaces must be used by a heterogeneous set of users, with substantial variety in effectiveness with the same interface when configured differently. We address the issue of personalizing such an interface, adapting parameters to present the user with an environment that is optimal with respect to their individual traits - enabling that particular user to achieve their personal optimum. We introduce a new class of problems in interface personalization where the task of the adaptive interface is to choose the subset of actions of the full interface to present to the user. In formalizing this problem, we model the user+task as a Markov decision process (MDP), wherein the transition dynamics within a task depends on the *type* (e.g., skill or dexterity) of the user, where the type parametrizes the MDP. The action set of the MDP is divided into subsets, each representing an interface, with different action-sets/interface optimal for different transition dynamics/type. The task of the adaptive interface is then to choose the right action-set. Given this formalization, we present experiments with simulated and human users in a video game domain to show that (a) action set selection is an interesting class of problems (b) adaptively choosing the right action set improves performance over sticking to a fixed action set and (c) immediately applicable approaches such as bandits can be improved upon.

Introduction

Interactive interfaces, such as natural gesture-based user interfaces, have revolutionised the way we interact with video games, robots and many related applications. A key attribute of such an interface is the way in which users' high-dimensional movements are interpreted and correspondingly mapped to the command set within the application. Many modern software applications have similar requirements. For instance, interfaces to search engines (e.g., Google, Bing), or photo-editing software (e.g., Adobe Photoshop), involve numerous configuration choices that, implicitly or explicitly, define the context of the interaction between the human user and the computational process actually being performed.

Almost all of these applications must be deployed with large user populations, with substantial diversity in the performance that results from any given pair of user and configuration setting. For instance, even with a simple interface like joystick-based navigation within a video game or field robotics application, there is variety in dexterity and skill. A mismatch between a user's skill level and settings such as the sensitivity level at which one interprets a particular motion as a signal can have substantial negative impact on the user's ability to perform a task. Sometimes the mismatches can be in assumptions about perceptual ability, e.g., how precisely aware a user is regarding the current computational context. As behavioural studies, e.g., (Valtazanos and Ramamoorthy, 2013), have shown, perceptual limitations can have a disproportionately large effect on user's performance when the task involves interactive decisions in a dynamic environment. Similar effects are observed in software applications, such as in the performance difference between young children and skilled adults when presented with a number of choices in an interface.

These issues of personalisation in diverse applications are unified in the computational problem of automatically shaping the environment so that it is best adapted to the specific user whom the system is currently facing. Our particular interest is in adapting online, to personalise on the fly as the user interacts with the system, without long calibration phases and with only partial knowledge of the underlying traits that induce variability (see, e.g., (Rosman et al., 2014; Hartmann and Schreiber, 2008) for user studies showing evidence for the utility of this kind of adaptation).

The objective of this paper is to concretely define this problem of user environment shaping by presenting a model of this problem. We also present experimental studies in a video game domain with human and simulated users to show that this adaptation is beneficial compared to using some a-priori optimal action set. We also posit some desiderata such adaptation algorithms should satisfy: it must afford - *safety*, i.e., ensuring that individualised adaptation does not lead to worse performance than a pre-configured statically optimal setting, *efficiency*, i.e., learning to obtain settings that enable the user achieve their maximal performance, and *speed*, i.e., the ability to adapt within a few episodes which roughly corresponds to users' patience. We present one simple algorithm that satisfies these desidirata, and show that more established bandit algorithms such as (Auer, 2002) do not. This leaves opens the door for development of more power-

ful algorithms in future work.

We now briefly present our approach to modelling the problem of interface adaptation by environment shaping. We model the user as a sequential decision maker, operating in an Markov decision process (MDP) (Sutton and Barto, 1998), where the MDP models the task. We identify an interface for the task with a *subset* of the set of MDP actions. The adaptation now takes the form of choosing the *action subset* the user is allowed to use when accomplishing the task. We posit that different user *types* (e.g. skilled, expert, risk averse, daredevil etc.) will accomplish the task in different ways, which will manifest as different transition distributions of the MDP. So for different user type/transition distribution different action sets might be optimal. The task of a learning algorithm in this problem will be to combine estimation of observed transition distribution with prior training with users, and then determine online which interface/action-set is the optimal and present that action-set to the user.

In the following, we proceed as follows. In subsection Related Work, we discuss work related to ours. In Section Interface Adaptation as Action Set selection, we present concrete definition of our problem as action set selection over parametrized MDPs. Then in Section Experiments, we motivate and present our video game domain that illustrates this above problem. We also discuss how bandit approach may be used in our case, and also describe our initial approach to this problem. After that we end with a conclusion.

Related work

The problem of learning to adapt to user behaviour is garnering increasing attention in different domains. HCI researchers consider user interface customization. For instance, Gajos, Wobbrock, and Weld (2008) automatically synthesize user interfaces tuned to user (dis)abilities. The process involves off-line calibration using a battery of tasks, based on which one obtains information necessary to pose the interface design problem as one of combinatorial search. Related works on personalisation, e.g., (Seuken et al., 2012; Zhang, Chen, and Parkes, 2009; Mourlas and Germanakos, 2008) parameterize user performance within decision and game theoretic models, to enable data-driven optimization. We take inspiration from these notions of searching for an appropriate user interface, with specific focus on continual online adaptation – which has received less attention in the literature.

A different line of related work is on plan recognition, e.g., see (Charniak and Goldman, 1991), where one uses traces of an agent's actions to infer which one of many possible plans they are attempting to execute. Recent instantiations of this type of problem have included work such as (Fern and Tadepalli, 2010) on hidden-goal MDPs. Our focus is different in that our problem is not really that of trying to predict the user's future state, based on the past actions; instead, we try to optimally shape the user environment and we try to infer latent 'personality traits' or skill/preferences of the user (as in the earlier referenced HCI work) to that end.

Other related work includes (Hoey et al., 2010), who synthesize optimal policies for issuing prompts to dementia patients, with latent 'attitudes', so as to guide them along desired plan sequences. More directly related to user interface adaptation, Hauser et al. (2009) present the concept of website morphing, where the configuration of a web interface is changed based on estimated 'cognitive state' of a user. In such work, the technical approach is to compute optimal policies within a partially observed MDP. The corresponding computational methods are known to be computationally expensive. For instance, Hauser et al. (2009) adopt restricted heuristics due to the difficulty of directly calculating optimal policies. Similarly, Fern and Tadepalli (2010) discuss complexity implications of even restricted versions of such models.

Although we conceptualise our problem in similar terms, in terms of latent states characterising user types, we adopt a different approach to the solution process. Firstly, we adapt the *action set* available to the user rather than intervene state-by-state on the users' actions. Because this gives us more direct control over the users' behaviour, our solution can be one of online model selection - which we implement in our experiments. Such a model is easy to train with limited examples, incurs low computational cost and can be made orders of magnitude faster than alternate solvers in online deployment to detect type. That said, the algorithm presented in the experiment section is merely a first step to explore properties of our model of interface adaptation via action set selection. We expect that more sophisticated solution methods could be brought to bear on this problem in future, including those adapted from works such as the above. This includes consideration, e.g., of users' learning behaviour (Rosman et al., 2014) and other forms of changing dynamics.

Interface Adaptation As Action Set Selection

Our main idea is to model the user+task as a MDP, model interfaces as different subsets of the action set, with the user type determining the transition function of the MDP. In subsection Our Model of Interface Adaption, we justify these choices in detail and define the learning problem. But before proceeding, we start by defining an MDP.

Markov Decision Processes

Markov decision processes are a popular model for capturing sequential decision problems (Puterman, 1994) (Sutton and Barto, 1998). A finite MDP $\mathcal{M}$ is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, T, R, \gamma)$ where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions and $\mathcal{R} \subset \mathbb{R}$ is the set of rewards. $T(s'|s,a)$ is a the state transition distribution for $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$ while and $R(s,a)$, the reward function, is a random variable taking values in $\mathcal{R}$. Finally, $\gamma \in [0,1)$ is the discount rate. A (stationary) policy π for $\mathcal{M}$ is a map $\pi : \mathcal{S} \to \mathcal{A}$. The Q function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ of a policy π is defined by: $Q^\pi(s,a) = \mathbb{E}[R(s,a)] + \gamma \sum_{s'} T(s'|s,a) Q^\pi(s', \pi(s'))$ and gives the total future expected discounted reward obtained by taking action a at the current step, and then following the policy π after that. The value function for π is defined as $V^\pi(s) =$

$Q^\pi(s, \pi(s))$ and is total future expected discounted reward obtained by following policy π. An optimal policy π^* is defined as $\pi^* = \arg\max_\pi V^\pi$ – the Q function is given by $Q^*(s,a) = \mathbb{E}[R(s,a)] + \gamma \sum_{s'} T(s'|s,a) \max_a Q^*(s',a)$. For the optimal policy the value functions is denoted by V^*. In general reinforcement learning, goal of the agent (user in our case) is to estimate Q^* and then choose the action $\arg\max_a Q^*(s,a)$ at state s. In the sequel, we assume, without loss of generality, a fixed starting state s° for a MDP and define $V^\pi \triangleq V^\pi(s^\circ)$.

Our Model of Interface Adaptation

From our discussion in the Introduction, our model of the interface adaptation should allow us to: (i) represent each user as a sequential decision maker trying to optimize some goal; (ii) express that users can be of different types; and (iii) specify interfaces as collection of actions that the user is allowed to choose from at any given point in time. We now give a model that fulfils these criteria. A plausible way to satisfy criteria (i) is to model the user+task as a MDP as it is a model of sequential decision making where the user has a very precise objective to optimize. In this case, the states $\mathcal{S}$ of the MDP represent the different stages of a task, the actions $\mathcal{A}$ represent the choices available from all possible interfaces, and the reward R gives feedback on to what extent the user has accomplished task. The transition T is used to satisfy criteria (ii).

In particular, given the above mapping from MDPs to users+tasks, we can make our model satisfy criteria (ii) by positing a *class* of MDPs parametrized by the user type. We assume that the user of our interface has a type τ from a finite set of types $\boldsymbol{\tau}$, and the transition function T has the form $T(s'|s,a;\tau)$. This models the fact that depending on the user type, the same action will have different effects, i.e., next or end state of the task. For instance, in an interface which is a joystick for teleoperating a robot, the effect of moving the joystick 'forward' with high level of sensitivity will depend on how quick the user's reflexes are (the type, in this instance). Therefore, our user+task space is modelled by a class of MDPs M of the form:

$$\mathsf{M} \triangleq \{((\mathcal{S}, \mathcal{A}, \mathcal{R}, T(\cdot;\tau), R, \gamma) | \tau \in \boldsymbol{\tau}\} \tag{1}$$

We now turn to (iii). To satisfy this criterion, we recall that each interface corresponds to a set of actions that are available to the user. In that case, it makes sense to posit that the action set $\mathcal{A}$ has been divided into a subsets $\alpha^1, \alpha^2, \alpha^n$, where each α^i corresponds to an interface and contains the actions available in that interface. So, if the interface is a website, then each α^i corresponds to a specific view of the site and α^i contains the buttons, text fields, instructions etc. available at that view. If the interface is a video game interface or the joystick for a teleoperated robot, then each may α^i contain motion actions available at a specific level of sensitivity. Putting all this together, this means that the action set $\mathcal{A}$ is a union of subsets $\alpha^1, \alpha^2, \cdots \alpha^n$ where in general we expect each α^i to be significantly smaller than $\mathcal{A}$.

Finally, to model the fact that a user only gets to use one interface at a time, we require that only one set α^i is *live* at any point in time, so that the user is only allowed to choose actions from the live set. Given a live set α_t at time step t, we assume that the user chooses the action $\pi^*_{\alpha_t}(s_t)$ where is s_t is the state at step t and $\pi^*_{\alpha_t}$ is the policy maximizing $V^\pi(s_t)$ among all policies which only chooses actions from α_t. We now define two reference strategies for action set selection. The first is the true optimal action set and the second one is the a-priori optimal action sets:

$$\alpha^* \triangleq \arg\max_\alpha V^\alpha_{\tau^*}, \qquad \alpha_* \triangleq \arg\max_\alpha \mathbb{E}_{W(\tau)}[V^\alpha_\tau] \tag{2}$$

So α^* is the action set/interface that we would have chosen if we knew the true user type τ^* from the beginning. And α_* is the optimal action set based on some prior W over the set of user types.

The Learning Problem. We define the learning problem in the above model as follows. The user solves a sequence of MDPs from the set M in collaboration with the learner, so that their mutual goal is to maximize the future expected discounted reward of the user. Each phase of the problem begins with an action set α_0 as live and a user with a particular type τ^* comes in to use the application. At each step t the user chooses an action a_t and in response the learner (adaptive interface) has to choose the interface α_{t+1} to make live for step $t+1$. The learner cannot observe the true τ^* but knows the value of T for each τ. Once α_{t+1} is made live, the user chooses her next action according to the policy $\pi^*_{\alpha_{t+1}}$. The objective is to make the best sequence of choices, α_i.

Experiments

To motivate our experiment, let us consider the interface adaptation issues faced when we want to maximize the enjoyment of a player playing a game such as Temple Run (Imangi Studios, 2011), available on mobile touchscreen devices. In such games, the objective is to guide a continuously moving character through a series of obstacles by swiping on the screen, tilting the device etc. Typically, the goal is to get the user to play as long as possible, and so the speed of the game-character should be set to match her skill level. If the user has poor reflexes on such devices, and if the speed of the game-character is too fast, then she will quickly get frustrated and not play for long. On the other hand, if she does has good reflexes, then setting the speed too slow will bore the user. The current strategy in such games is often to start the character at a specific speed and keep increasing the speed in fixed pre-determined increments, which can be sub-optimal. The better strategy to maximize play time seems to be to adaptively set the game-character speed by assessing her skill level. This is exactly what our formulation achieves.

Specifically, we frame this adaptation as a simple (possibly the simplest) instance of live action-set selection where each possible speed of the game-character corresponds to an action set. The actions in a particular action set are the same (swipe on screen, tilt device) but the effect is different, depending on the action set. This is also an illustration of a situation where it is obviously better to have distinct action sets rather than having all the actions together simultaneously – having all the swipe and tilt actions at different speeds available at the same time would make the game extremely hard, and reduce its enjoyability significantly. Similar ideas can

be used, in future, to address even more challenging and dynamic problems like the ones we presented as motivation in the introduction.

In the following, we first describe our domain, which distils the essential interface adaptation issues in the above domain. We then describe the algorithms that we use as baseline for comparison and then present our results.

The Red Ball Video Game Domain

The domain we use is a simple video game where the user controls a red ball from a target location to a goal location (see Figure 1). The ball speed can be set to one of several possible values and the player/user can only control the direction of motion. Her goal is to take the ball from a start location to the goal location as quickly as possible without bumping into walls. The user has several possible skill levels (i.e. types) when controlling the ball, and her skill level determines the optimal speed. So, for instance, a highly skilled user finds the highest speed optimal, while a less skilled user prefers a lower speed (otherwise she will collide with the wall too often).

Specifically, the ball moves with a continuous speed p pixels-per-second and taking an action in the each of the cardinal directions N, S, E, W changes the motion of the ball to that direction. The speed p is one of $\{30, 40, 50, 60, 70\}$. Given the simple nature of the game, we explicitly enforce the different user skill levels by giving each user different ball sizes of b pixels where b is one of $\{2, 5, 10, 20, 40, 60\}$. We also add a noise level of $n_b = b/2\%$ to the motion of the ball to further enforce different performance. The size of the ball models limitations of user perception, while noise models limitations of motor ability. So, larger ball+noise imply less skill. In our experiments, none of the adaptive algorithms know the true ball size b or noise n_b during the testing.

MDP Formulation of Interaction Adaptation

To construct our MDP formulation, we need to specify state space $\mathcal{S}$, actions $\mathcal{A}$, rewards $\mathcal{R}$, transition function T, reward function R and the set of action sets α. $\mathcal{S}$ is the location of the ball at each time step (the game field is 1000×1000). Each action set α_p corresponds to all motions at speed p. The actions $a \in \alpha_p$ were motions in each of the cardinal directions N, S, E, W at speed p.

The direction of motion at step t is the most recent direction chosen by the user at $t' \leq t$ (so, the actual play of the user is modelled in the MDP as the user choosing the same action at every step as her last choice of direction). There was a constant amount of noise (on top of noise due to skill/type) with each action, so the ball moves in the given direction followed by (discretized) Gaussian motion (perpendicular to the direction of motion) with mean 0, and $\sigma \in [0.3, 8]$ (chosen randomly for each episode).

The transition function T is defined as follows. There were 5 different types: ball size $b \in \{2, 5, 10, 20, 40, 60\}$, noise levels $n_b = b/2\%$. So for type $\tau = (b, n_b)$, with probability $1-n_b$, the ball moves in the current direction, and with probability n_b, it moves in another random direction. Hence, the type modifies the base transition probabilities (1) by the noise n_b and (2) because the ball sizes determine the locations that result in collisions. Finally for the reward function R, the player receives a reward of -1 for every time step and -5 for every collision.

Experiment Description

We performed two sets of experiments on the above domain, one with simulated users and another with human users. In both cases, we ran a training and test phase. The data collected during the former was used to train our algorithms which were then used during the latter phase. During training, we provided transition distribution information for each $(b, n_b) \times z$ (see below for details). The simulated user policy was always the A^* algorithm, where the direction chosen at each step, at speed p, was toward the lowest cost location, at distance p from the current location. During the test phase we chose the type at random and recorded the loss (see below for details on the amount of data used).

The strategies we used were α^* (choose the optimal action set (2) for the true type), α_* (choose the a-priori/population optimal action set (2)), and the algorithms we used were the EXP-3 algorithm for bandits (Auer, 2002), and our own algorithm that chooses the *Bayes optimal action set with* additional exploration. We describe these two in more detail.

Algorithms

It is very natural to formulate the problem of live action set selection as a non-stochastic multi-armed bandit problem (NSMB). In the bandits problem, there are c *arms* where each arm i has a *payoff process* $x_i(t)$ associated with it. The learner runs for T steps and at each step t needs to *pull*/select one of the arms $f(t)$ and his payoff is $x_{f(t)}(t)$. Additionally, the learner only gets to view the payoff of the arm $f(t)$ it has chosen. The goal of the learner is to minimize its *regret* with respect to the best arm, that is minimize the quantity $\max_i \sum_{t=1}^{T} x_i(t) - \sum_{t=1}^{T} x_{f(t)}(t)$. In general it is not possible to minimize this in any meaningful sense. An optimal algorithm in the general case for minimizing the *expected regret* is the EXP-3 algorithm mentioned above. For live action-set selection, we can identify each action set α as an arm and the payoff received for choosing the arm as the reward in the following time-step. With this identification we can use EXP-3 exactly as described in (Auer, 2002).

The second algorithm is a simple one, which we call BOA, that, at each step t, with probability ϵ makes live an action set at random, and with probability $1 - \epsilon$ makes live the *Bayes optimal action set* (hence the name BOA for the algorithm) . This action set defined as follows. Assume that at time step t of a particular interaction session we have observed a state-action sequence $sa_{0:t} \triangleq s_0 a_0 s_1 a_1 \cdots s_t$ (with $sa_{0:0} \triangleq s_0$). Given this session, the likelihood of a type τ_i is $L(\tau_i | sa_{0:t}) = \prod_{i=0}^{t-1} T(s_{i+1} | s_i, a_i; \tau_i)$ (the function $T(|; \tau_i)$ is learnt during the training phase. The posterior probability of τ_i is $Pr(\tau_i | sa_{0:t}) = L(\tau_i | sa_{0:t}) W(\tau_i)$ where the prior $W(.)$ is the frequency with which we observed τ_i during training. The Bayes optimal action-set α_{BO} is the one that maximizes the expectation of V_τ^α with respect to the poste-

rior over $\boldsymbol{\tau}$:

$$\alpha_{BO}(sa_{0:t}) \triangleq \arg\max_{\alpha} \sum_{i} Pr(\tau_i|sa_{0:t})V^{\alpha}_{\tau_i} \qquad (3)$$

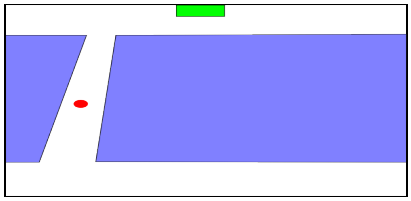

Figure 1: The Red Ball video game domain; green rectangle is the goal location, and the red circle is the ball. The initial location was a random location in the bottom.

Results

Data Used For Experiments. For the simulated user case, during training we generated 1548 trajectories from which we learned the $T(\cdot|\cdot;(b,n_b))$ for each type of ball-size+noise (b,n_b). The trajectories were divided uniformly over the different types. The results we present in graphs below are for the testing phase, which contained 344 trajectories with the types draw uniformly at random before each trajectory. For the human experiments we used three different human users. For the training phase we used 311 different trajectories and for the testing phase we used 51 trajectories and the results below are based on these latter ones.

Results For Simulated User. The results are presented in Figures 2 – 4. Figure 2 shows the loss of each p for three representative (b,n_b) and establishes the baseline performance of action-sets for these types: higher speeds tend to work better for smaller sizes and conversely.

Figure 3 shows that adaptation is more beneficial when we have greater number of types of users in our pool of users. Here, in the x-axis each x corresponds to all possible combinations of x types (so all $\binom{6}{x}$ combinations). This captures different levels of heterogeneity in user populations (i.e. population diversity). In the figure, for the curve α_*, the point $\alpha_*(x)$ gives the loss of best speed averaged across all populations of size x. Hence $EXP\text{-}3(x)$ gives the loss of EXP-3 averaged over all population of size x, while $BOA(x)$ gives the loss of BOA averaged over all population of size x. This curve shows, that averaged across population sizes, BOA outperforms EXP-3 significantly. Furthermore, while for the smaller population size, the α_* beats BOA, for the larger population sizes BOA beats α_* comprehensively, illustrating the need for adaptation. Additionally, note that the curve for α^*, the optimal per type, averaged over all types, is the line $f(x) = \alpha_*(1)$ because $\alpha_*(1)$ gives the 'population optimal' of size 1, which is just α^*.

Finally, Figure 4 shows BOA is able to identify types very quickly. It shows that BOA has trouble identifying the smallest type. We conjecture that this is because the behavior of $b = 5$ and $b = 2$ are nearly indistinguishable. However, other than that, it is able to identify the correct type fairly rapidly, almost always halfway through the first episode.

Results for Human Users. The results for the human experiments are presented in Figures 5 – 7 and they draw similar conclusions as with the simulated user. Figure 5 establishes the baseline as before and shows different speeds are optimal for different types.

Figure 6 shows the performance of BOA and α_* for the human experiments in terms of population diversity. We do not report the performance of EXP-3 because data collection for that algorithm takes time that exceeded the constraints of our human subjects. In these experiments, we see that BOA significantly outperforms α_* for all population sizes, hence demonstrating the benefit of adaptation with real human subjects, corroborating our more extensive simulation results above. *This is a key experimental result of this paper.*

Finally, Figure 7 shows the efficacy of our algorithm at identifying different types. As in simulated user experiments, our algorithm has some trouble distinguishing between the very nearby types $b = 5$ and $b = 2$ but the error is not beyond the threshold of small noise.

Safety, Efficiency and Speed in Experiments

Our stated goal in the introduction was to construct algorithms that have safety, efficiency and speed. The experiments show that our algorithm BOA is safe. For simulated users, BOA does not under-perform w.r.t. any alternatives including EXP-3 and static best speed α_*. In the human experiments, BOA always does better than α_* (Figure 6). In the artificially constrained low-population-diversity setting (1-3) with simulated users, α_* is marginally better than BOA; but BOA outperforms at realistic levels of diversity (Figure 3). Hence, a key conclusion of our experiments is that *this kind of adaptation is indeed necessary in environments with substantial diversity as the regret of alternate approaches shoots up at high diversity*. BOA is also efficient, in particular, Figures 3 and 6 shows that we reach the optimal action-set. Finally, Figures 4 and 7 also shows that BOA can identify the true types fairly rapidly, demonstrating speed. We also notice that EXP-3 satisfies none of the desidarata, showing need for novel algorithms.

Conclusions

In this paper we concretely formulated the problem of adaptation of interactive interfaces by live action set selection, which is a kind of environment shaping. We presented what we hope are convincing simulated and human experiments that demonstrated the need for such adaptation and scope for further research. Currently, we perform model selection in the setting where the user changes her behaviour immediately when the action set changes. While this assumption has not been substantially contradicted in our experiment, it is likely that deviations may be more substantial as task complexity goes up. A direction for future work is to incorporate models of user learning, e.g., (Rosman et al., 2014), into this model selection process. Similarly, we work in a setting where a training corpus corresponding to various types seeds our learning algorithm. Incrementally acquiring data corresponding to heterogeneous types and performing (fully unsupervised) learning in a life-long fashion is another important direction to be explored.

SIMULATED USERS

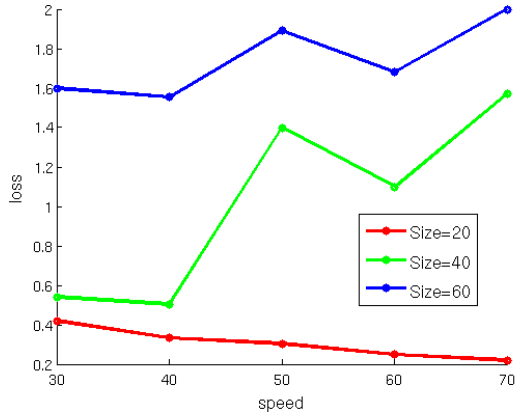

Figure 2: The performance of each speed for each type establishing baseline domain performance.

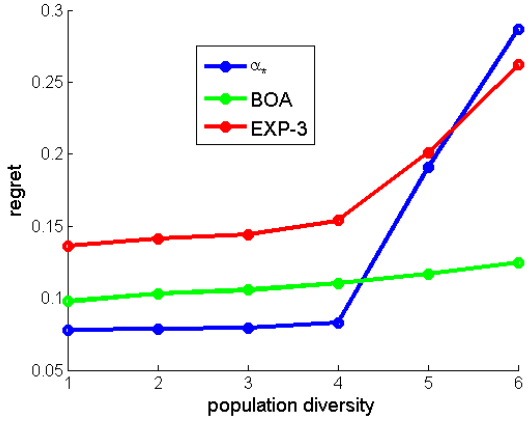

Figure 3: The performance of best static speed α_*, EXP-3 and BOA for each of 6 possible combinations of type populations when the user is simulated.

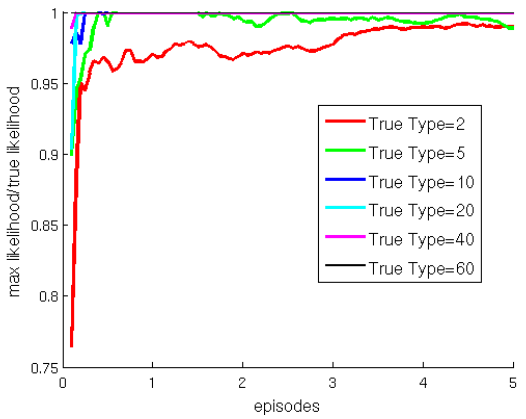

Figure 4: The ratio of the maximum-likelihood type and posterior and the true type in BOA, averaged over all population-diversity trials in our experiments.

HUMAN USERS

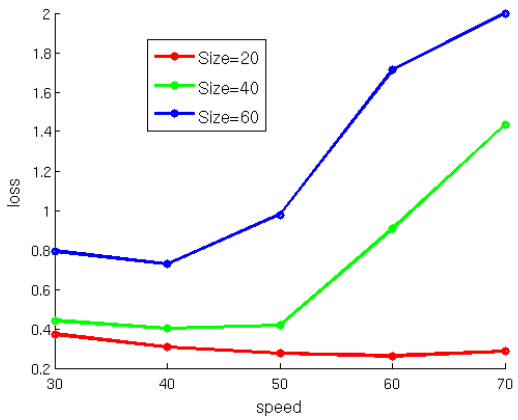

Figure 5: The performance of each speed for each type, establishing baseline performance in domain.

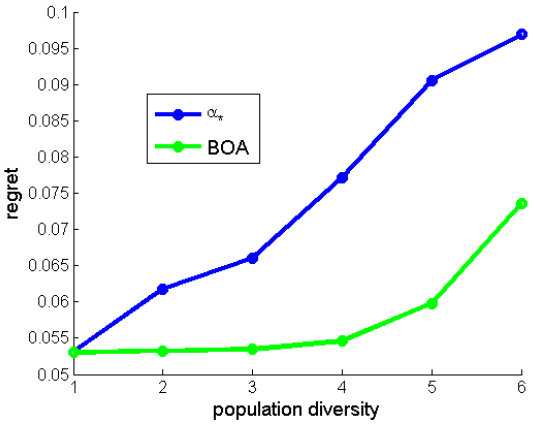

Figure 6: The performance of best static speed α_* and BOA for each of 6 possible combinations of population types.

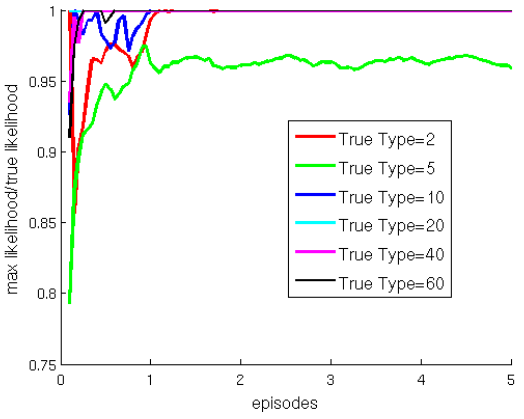

Figure 7: The ratio of the maximum-likelihood type and posterior and the true type in BOA, averaged over all population-diversity trials in our experiments when the user is human.

References

Auer, P. 2002. Using upper confidence bounds for exploitation-exploration tradeoffs. *Journal of Machine Learning Research* 3:397–422.

Charniak, E., and Goldman, R. 1991. A probabilistic model of plan recognition. In *Proceedings of the ninth National conference on Artificial intelligence - Volume 1*, AAAI'91.

Fern, A., and Tadepalli, P. 2010. A computational decision theory for interactive assistants, advances in neural information processing systems. In *Proceedings of the* 23^{rd} *Conference on Neural Information Processing Systems*.

Gajos, K.; Wobbrock, J.; and Weld, D. 2008. Improving the performance of motor-impaired users with automatically-generated, ability-based interfaces. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, 1257–1266.

Hartmann, M., and Schreiber, D. 2008. Proactively adapting interfaces to individual users for mobile devices. In *Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*.

Hauser, J. R.; Urban, G. L.; Liberali, G.; and Braun, M. 2009. Website morphing. *Marketing Science* 28(2):202–223.

Hoey, J.; Poupart, P.; von Bertoldi, A.; Craig, T.; Boutilier, C.; and Mihailidis, A. 2010. Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process. *Computer Vision and Image Understanding* 114(5):503–519.

Imangi Studios. 2011. Temple run. *This game was Developed and Published by Imangi Studios*.

Mourlas, C., and Germanakos, P. 2008. *Intelligent User Interfaces: Adaptation and Personalization Systems and Technologies*. Information Science Reference.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons.

Rosman, B.; Ramamoorthy, S.; Mahmud, M. H.; and Kohli, P. 2014. On user behaviour adaptation under interface change. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*.

Seuken, S.; Parkes, D. C.; Horvitz, E.; Jain, K.; Czerwinski, M.; and Tan, D. S. 2012. Market user interface design. In *ACM Conference on Electronic Commerce*, 898–915.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Valtazanos, A., and Ramamoorthy, S. 2013. Evaluating the effects of limited perception on interactive decisions in mixed robotic environments. In *HRI '13: Proc. ACM/IEEE International Conference on Human-Robot Interaction*.

Zhang, H.; Chen, Y.; and Parkes, D. C. 2009. A general approach to environment design with one agent. In *IJCAI*, 2002–2014.