

Coordinators Autonomy Module

Technical Report

Janusz Marecki, Zvi Topol, Sven Koenig and Milind Tambe
University of Southern California
3737 Watt Way, Los Angeles, CA 90089
{marecki,topol,skoenig,tambe}@usc.edu

Vu Ha
Honeywell Aerospace
3660 Technology Dr.
Minneapolis, MN 55418
vuha14@gmail.com

ABSTRACT

Coordination of a team of agents assigned to a time critical task has proven to be a challenge for the AI community. If team members include humans, this task becomes even more complex due to human cognitive limitations. To remedy this, humans are usually provided with personal assistants, called proxies, which handle all team level coordination. Since real tasks are actually performed by humans, the interaction between a human and its proxy, called adjustable autonomy, has been an intensive area of research. Three adjustable autonomy challenges have emerged to be particularly important: (i) learning optimal transfer of control strategy between human and its proxy, (ii) learning the human preference model and (iii) learning the human responsiveness model. In this paper we describe AutonMod, a novel adjustable autonomy module which successfully addresses the above-mentioned challenges. AutonMod uses Markov Decision Process to learn the different transfer of control sequences; to learn the human preference and responsiveness model it leverages the concept of Neural Networks guided by rules extracted from a given Knowledge Base. We test AutonMod in the scope of DARPA Coordinators project. Our experiments show that AutonMod learns human preference model and increases its confidence in autonomous decision making. We also demonstrate that AutonMod transfer of control strategies outperform naïve adjustable autonomy strategies.

1 Introduction

This document presents the current state of our evaluation plans for the Honeywell team’s Coordination Autonomy Module (AutonMod). The AutonMod will interact with the human users to select changes to the mission schedule. We view this as a sequential decision-making problem in which the AutonMod must decide whether the human or the AutonMod will select amongst a set of alternatives developed by the other COORDINATOR modules. We explore solutions to this problem by leveraging and extending state-of-the-art methods for Markov Decision Processes and modern classifier systems.

This document includes a brief overview of our baseline approach, our hypotheses about system performance, and the experimental variables and metrics we use to evaluate the performance of our evolving system.

2 Approach Overview

In our COORDINATOR architecture, the TaskMod and CoordMod may develop one or more alternative schedules for some or all of the mission tasks, in response to various changes (e.g., task outcomes, new task arrivals). The AutonMod is responsible for working with the human user to select the best of those alternative schedules (which are represented as options in AutonMod). In some situations, the AutonMod may decide by itself; in others the AutonMod may offer the decision to the human or it may defer a decision and “play for time” by interacting with other COORDINATOR modules or selecting a schedule that delays commitments.

As illustrated in Figure 1, we view the AutonMod problem as a sequential decision-making problem in which the AutonMod must consider:

- Whether to make the decision autonomously or consult the human or try to “play for time” to delay some or all of the decisions.
- If deciding autonomously, how to select among the alternatives.
- If consulting the human, how long to wait before asking the human.

The problem is sequential because the system faces a receding horizon of future choices, and time is important: humans take significant time to decide, mission deadlines approach, and multiple problems with various decision deadlines must be handled. The AutonMod also needs to reason about both uncertainties (e.g., uncertainty in user responses), and rewards (e.g., effects on mission outcome). MDPs provide a natural framework to address these issues.

There are two fundamentally different objectives the AutonMod might be aiming at: it might aim to maximally support its human user’s individual performance and preferences or it might aim to help the overall team of COORDINATOR-supported humans achieve the maximum possible mission performance quality. If the AutonMod is trying to make its user happy, it must ask the user to make the scheduling decision when the user wants to be asked, and when the AutonMod makes the decision autonomously, it must be as similar as possible to what the human would have decided. If

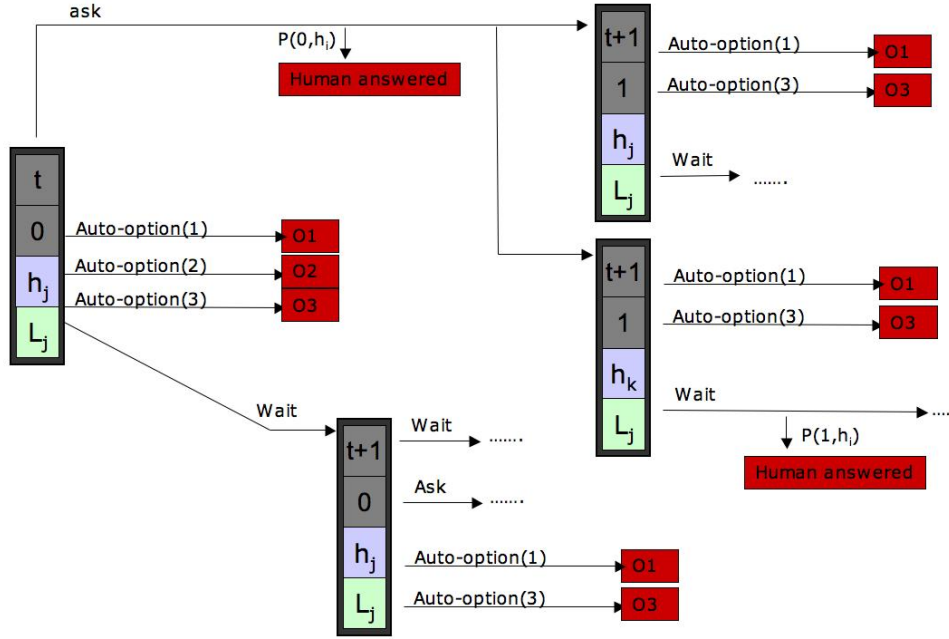


Figure 1: AutonMod must solve a sequential decision problem that can be viewed as an MDP. Possible actions are to choose an option autonomously, ask a human or wait. The states are defined by t (time elapsed since MDP execution has started), d (time elapsed since human was asked – second parameter in the state description), h_j (current human cognitive load) and L_i (list of still available options).

the AutonMod is maximizing overall mission performance, those factors are unimportant, and the AutonMod should pick team-optimal choices (even at the cost of locally sub-optimal performance) and only ask the human to decide when its own information is likely to be insufficient to make the optimal choice (e.g., when it is in a highly dynamic domain that the human might have a better ability to predict).

We hope to develop a unified AutonMod approach that will support both of these objectives, and potentially also intermediate gradations, without algorithmic changes. By casting these different AutonMod roles as objective or reward functions, the same mechanisms can be used to achieve either type of performance emphasis. For now our evaluation of AutonMod focuses on the first item above, i.e, support its human user’s individual performance.

2.1 Concrete System Design

The system architecture as depicted in Figure 2 is composed of the following modules:

- User Module which consists of:

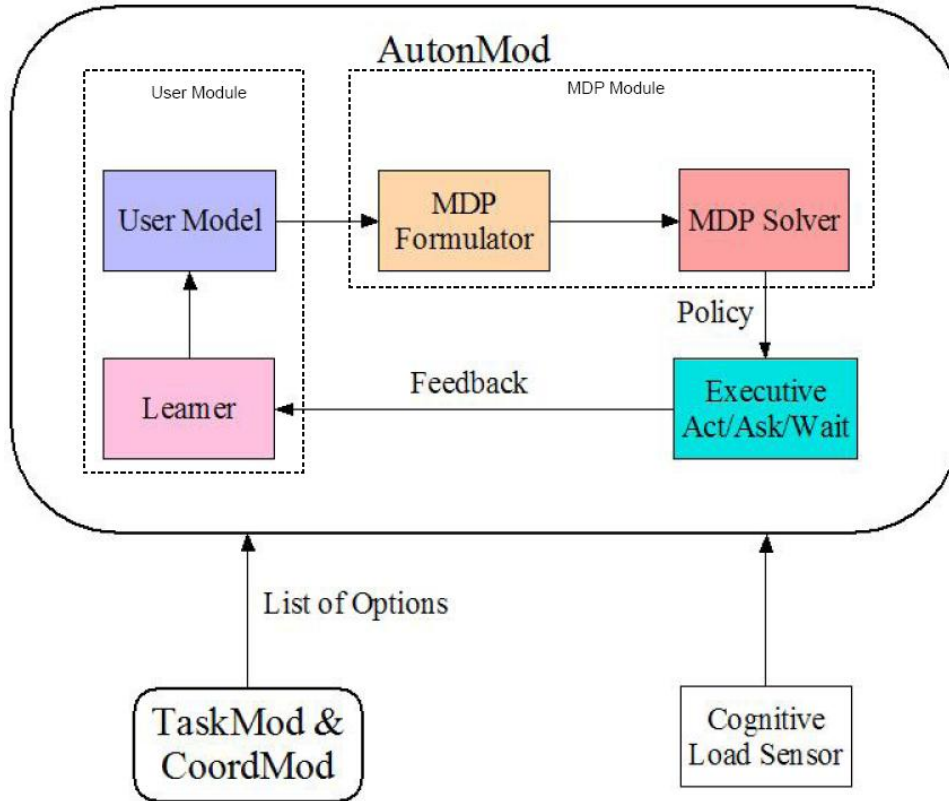


Figure 2: AutonMod system architecture.

- *Preference model of the user:* This model is a representation of the ranking of options according to their preference by the user. Options are characterized by attributes such as deadline, quality, probability of success, etc. For example, the attribute “deadline” represents a time deadline after which the option is no longer available. Ranking of options is learned over time using a Knowledge Based Artificial Neural Network (KBANN). A KBANN is an artificial neural network used to learn a theory (in our case, the theory is the preference model of the user). Accurate theory greatly speeds up learning of the KBANN.
- *Responsiveness model of the user:* This is a probabilistic model describing the probability over time that the user responds. We use the Weibull distribution for this purpose. This distribution belongs to the exponential family of distribution functions, and is used due to its desirable properties to model probability of firing events over time (in our case, the event is that the user has responded). Some of the parameters of this distribution are affected by the cognitive load dynamics of the user, whereas others are assumed to be fixed (representing a-priori knowledge about the user). The cognitive load dynamics of the user is represented by a transition matrix over different levels of cognitive load levels. We use 3 levels of cognitive load: high, medium and low. Thus, the transition matrix is a 3 X 3 matrix. The transition matrix is learned over time.

Hence, the User Module subsumes the User Model and the Learner described in Figure 2.

- MDP Module which consists of:
 - *MDP Formulator*: The formulator constructs: MDP states, transition and reward models, and 3 possible actions: ACT, ASK and WAIT. The ACT action chooses an option autonomously out of the set of currently available options. ASK asks the user which option to choose, and WAIT waits for user response. The ASK and ACT actions can only be executed once, whereas the WAIT action can be taken multiple times. Rewards for the ACT action depend on the confidence level, where confidence level is an internal estimate which measures the confidence AutonMod has in its learning capabilities. The AutonMod confidence depends on the accuracy percentage and estimated learning accuracy of the right cognitive load level.
 - *MDP Solver*: This component solves the underlying MDP and outputs the optimal policy.

3 Evaluation

To evaluate the performance of AutonMod, we will conduct a series of experiments that examine the module’s behavior both in isolation and embedded in the context of a single COORDINATOR system. In the first year, as the rest of our COORDINATOR architecture was evolving rapidly, we have primarily evaluated the AutonMod in isolation. When performing such evaluation in isolation, we have assessed its ability to emulate human performance, i. e. matching human preferences when making autonomous decisions. Later, when evaluating the AutonMod in its COORDINATOR context, we will assess its ability to improve overall system performance according to varying metrics, in comparison to simpler alternative (“straw man”) approaches.

3.1 Isolated User Simulation Framework

In our experimental setup, we use a simulated user composed of preference and responsiveness models, as illustrated in Figure 3 and described earlier. We assume that the preference model of the user and hence the options chosen by the user are fixed. Therefore, we are concerned with noise in the responsiveness model.

Given the two features of the responsiveness model discussed earlier (probability distribution over time that the user responds and cognitive load dynamics), we hereby explain how these features are affected by noise:

- *Probability over time that the user responds*: This probability is currently modeled as a fixed Weibull probability distribution. We currently model the noise as a difference between the Weibull parameters of the simulated user and the Weibull parameters of the human model AutonMod uses.
- *Cognitive load dynamics*: Noise in this model affects the learning of the user cognitive load transition matrix. The training data for this matrix is a sequence of observed cognitive load

changes. The accuracy at which each element of this sequence is observed, is a function of the noise level (higher levels of noise imply lower accuracy).

Our current experimental plan includes experiments with noise affecting the second feature. We fix the simulated user’s transition probability matrix and experiment with “perturbed” transition matrices in AutonMod’s model.

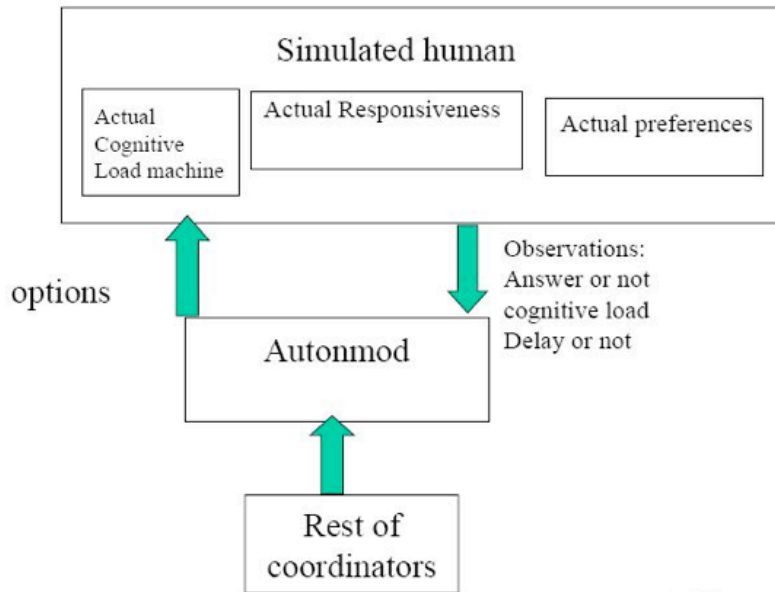


Figure 3: Experimental Setting Diagram

The simulation then runs both the simulated user and AutonMod over different episodes, allowing AutonMod to learn from feedback generated by the simulated user over time; whenever the user responds and chooses an option, AutonMod compares the user’s choice to its own choice (the option that AutonMod would have chosen had it been required to act autonomously) and learns accordingly.

An episode is characterized by the following:

- AutonMod is presented with several options (as specified earlier in the Concrete Design section), which are characterized by attributes such as deadline, quality, probability of success, etc). In our current experimental setting, options are characterized by deadline.
- At every time step AutonMod can decide whether to ask the user if not done already (taking the ASK action), wait for a response of the user if the user was asked or just take no action

(this means taking the WAIT action), or make a decision autonomously (taking the ACT action).

- Once the deadline of an option expires, the option is no longer available.
- The episode ends once the user has chosen an option, AutonMod decides to act autonomously and picks an option, or only one option remains available. In the latter case, AutonMod will choose this option autonomously.
- Response time of AutonMod is measured in number of time steps from the beginning of the episode until its end.
- An episode starts with a random cognitive load level that might change every time step according to the cognitive load dynamics matrix (as mentioned earlier there are 3 different cognitive load levels: high, medium and low, and hence the cognitive load dynamics matrix is 3 X 3).

Each run is composed of 500 episodes, and each outcome of an experiment is produced over the average of 50 runs. Experimental measurements are generated in the following way: we divide the 500 episodes of each run into 20 bundles of 25 episodes, and measure the progress of the average value of the relevant experimental metric from bundle to bundle. This approach allows us to demonstrate the evolution of AutonMod behavior over time. Note that we do not learn from run to run, but only within a run.

3.2 Performance Metrics

We define *episode utility* as follows:

- Each ASK action has a fixed cost $C * h_i$, where C is a constant and h_i is the current cognitive load level.
- Each WAIT action does not incur any cost.
- When the user responds, the episode utility is increased by the maximal possible reward R_{max} over all currently available options (assuming that the user chooses the optimal option).
- Each ACT action adds to the episode utility $confidence * R_{max}$.

To assess the system performance and provide data for assessing the hypotheses defined in the next sub section, we will collect at least the following metrics:

- *Response time of AutonMod*: Measure of time elapsed until AutonMod decides to ask or make choice itself, and time to output the choice.
- *Utility of an experiment*: *Utility* of an experiment is defined as the average of utilities for a single run, over all the runs. Utility for a single run is the sum of all episode utilities over the different episodes of the run divided by the average AutonMod response time during that run (hence preferring faster action on behalf of AutonMod).

- *Accuracy*: We look at accuracy of match between AutonMod decisions and the decisions that the nominal “simulated user” would have made. We calculate accuracy according to the following formula: $\frac{|L|-pos(Op)}{|L|}$, where $|L|$ is the length of currently available options and $pos(Op)$ is the position of option chosen by AutonMod in the option list (option list is sorted according to “goodness” of options, i. e. first option is the best option to be chosen, second is the best second, etc).

3.3 Hypotheses

We evaluate the following hypotheses:

- (i) **Online generated policies outperform static policies** — At a cost of negligible overhead, policies generated online using the MDP solver are better than the following static policies:
 - The “always ask the user” policy.
 - The “always autonomously decide” policy.
 To evaluate this hypothesis, we will compare the utility achieved by using the optimal policy generated given our Markov decision process model (whose look-ahead level is 10 time steps) against the utilities achieved by using the static policies.
- (ii) **Increasing the look-ahead of the MDP formulation produces better online policies** — Considering future decisions can improve AutonMod performance. To evaluate this hypothesis, we will compare the utility achieved by using the optimal policy generated given our Markov decision process model (whose look-ahead level is 10 time steps) against the utilities achieved by using the optimal policies generated for the same Markov decision process models, but with look-aheads (i. e. time horizon) of one and two time steps.
- (iii) **AutonMod learns to improve its user’s responsiveness model** — AutonMod will improve its performance by refining its user’s responsiveness model. To evaluate this hypothesis, we will demonstrate the increase in utility when the cognitive load dynamics transition matrix is learned rather than fixed in advance.
- (iv) **AutonMod learns user preferences** — When start with a random or approximate model of user preferences (initial preference knowledge base) and given consistent training on how the actual user would choose, AutonMod’s decision-making will converge to match the human preference with high probability. Consequently, this will result in better policies.

4 Experimental Results

Each experiment corresponds to each of the above mentioned hypothesis.

Experiment testing hypothesis (i)

- *Overview*: We compare the utility of three different policies. First policy always asks its user which option to choose; second policy always chooses an option autonomously, whereas the third policy is the optimal AutonMod policy generated by solving its corresponding MDP.

- *Parameters:* We experiment with each one of the different policies separately.
- *Metrics:* We measure the average utility for each policy over 50 different independent runs.
- *Results:* In Figure 4 we plot a graph where the y-axis indicates average utilities of the three different policies, and the x-axis indicates 50 different independent runs where the average utilities were attained. The graph indicates that the policy solved by our MDP (full step look-ahead policy) gets higher average utility than both first two policies (always ASK and always ACT) over all 50 runs.
- *Conclusions:* Policy generated by AutonMod outperforms fixed policies generated online.

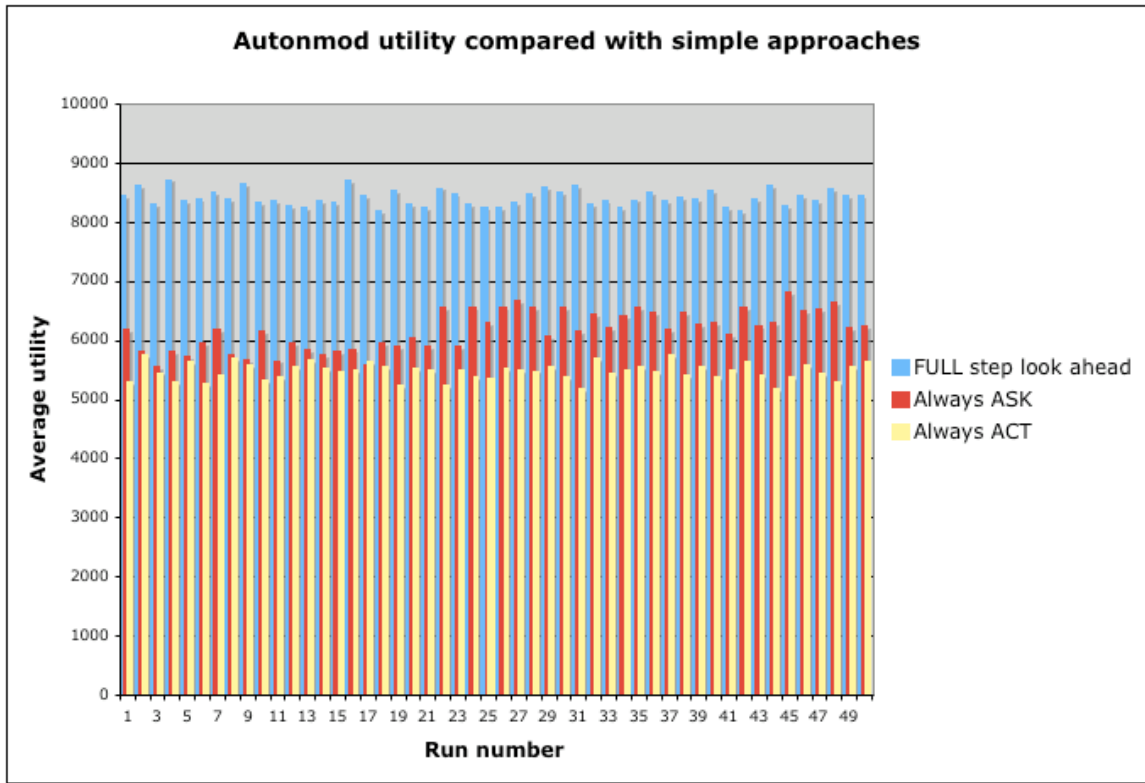


Figure 4: Experiment results for testing hypothesis (i)

Experiment testing hypothesis (ii)

- *Overview:* We compare the utility of three different AutonMod policies obtained by solving three corresponding MDPs with different look-ahead levels: one, two and 10.
- *Parameters:* We vary the MDP look-ahead level.
- *Metrics:* We measure the average utility for each policy.

- *Results:* In Figure 5 we plot a graph where the y-axis indicates average utilities of the three different policies, and the x-axis indicates 50 different independent runs in which the average utilities were attained. The figure indicates that the full look-ahead level policy (10 look-ahead level) obtains better average utility than both one and two look-ahead levels.
- *Conclusions:* Increased look-ahead of AutonMod’s MDP transfers into higher quality decisions.

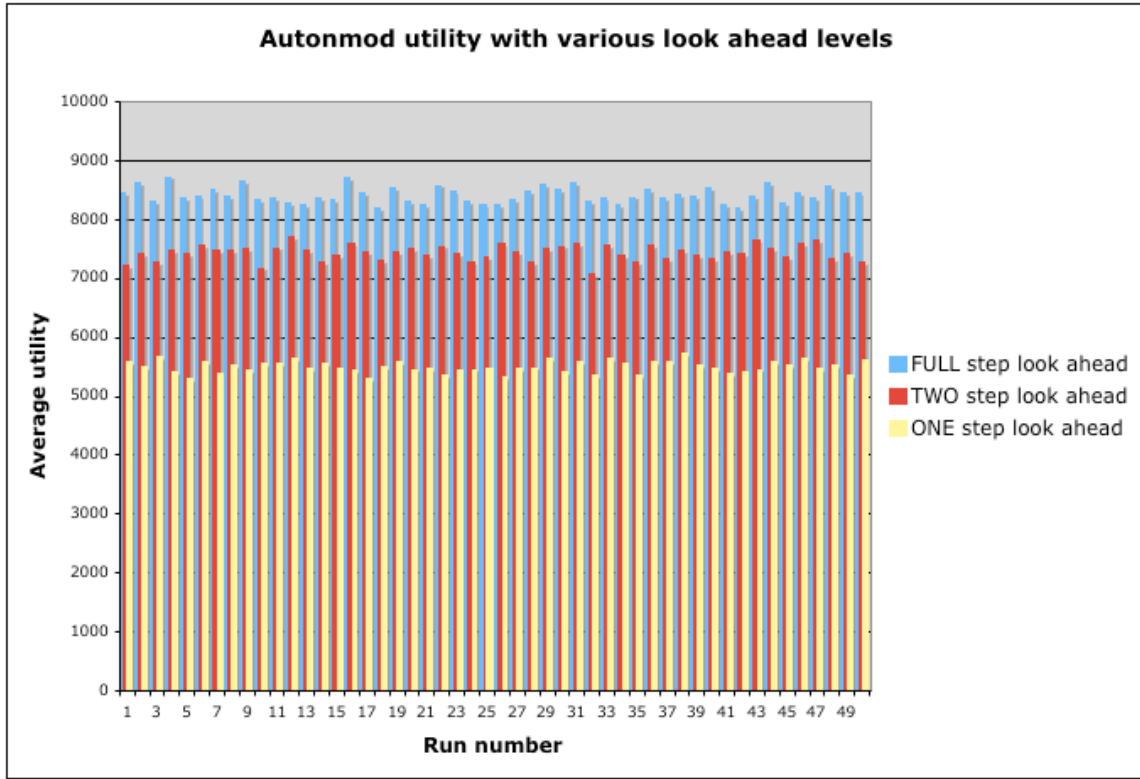


Figure 5: Experiment results for testing hypothesis (ii)

Experiment testing hypothesis (iii)

- *Overview:* Learning the responsiveness model implies predicting correctly the user cognitive load level dynamics; Better model of user cognitive load dynamics (which is represented by a transition matrix) translates into higher confidence of AutonMod, which in turn yields better response time.
- *Parameters:* We perturb the observed history of cognitive load levels which is a sequence of observed cognitive load changes. Each element of this sequence is observed accurately with a certain probability related to the noise level.
- *Metrics:* We measure the response time of AutonMod over time.
- *Results:* In Figure 6 we plot a graph where the y-axis indicates the average response time in each bundle (each bundle contains 25 episodes), and the x-axis indicates episodic bundle

number (i. e. the sequential number of the bundle in the run). The trend of the graph indicates that AutonMod delivers options quicker over time.

- *Conclusions:* AutonMod response time improves over time as a better model of user cognitive load dynamics is learned.

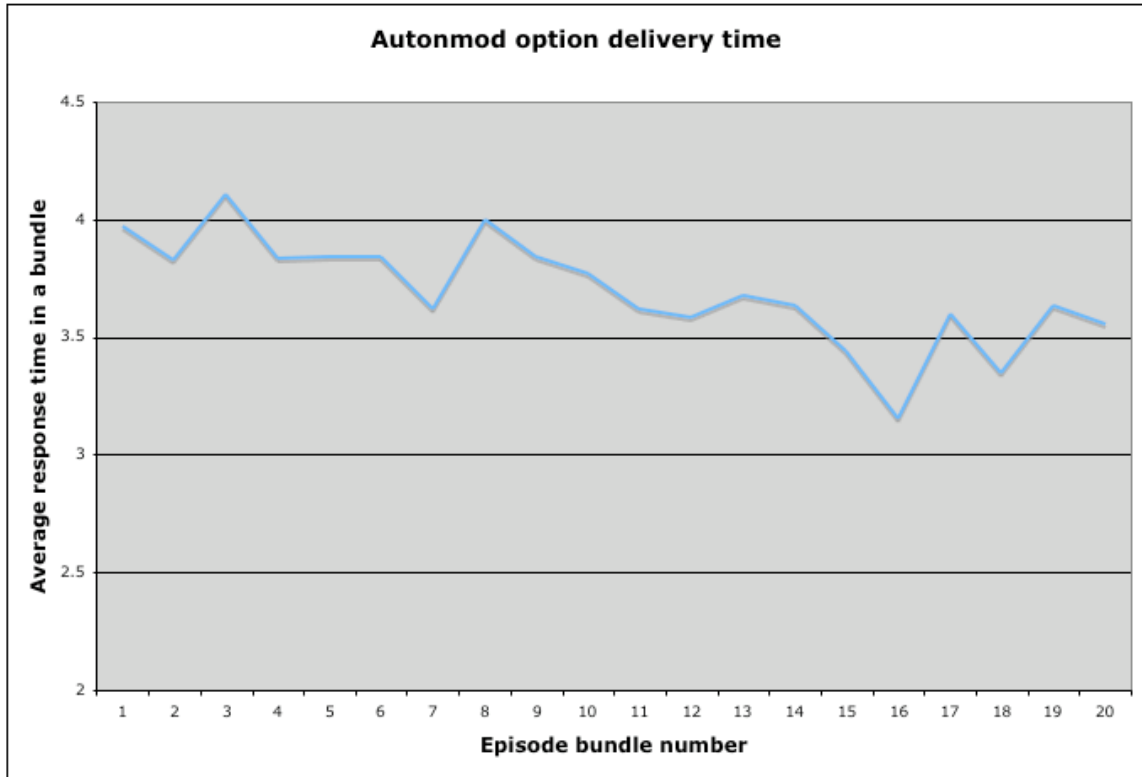


Figure 6: Experiment results for testing hypothesis (iii)

Experiment testing hypothesis (iv)

- *Overview:* Learning the user preferences implies predicting correctly which option would the user choose.
- *Parameters:* We test the learning ratio with / without the initial knowledge base
- *Metrics:* We measure the average accuracy (the third metric) for subsequent episodes
- *Results:* We plot on the x-axis the subsequent episode bundle number and on the y-axis the percentage of autonmod option choices which matched user option choices.
- *Conclusions:* Autonmod learned the user preference model both with and without using the initial knowledge base. With the knowledge base Autonmod learned to predict the user option choice with 99/

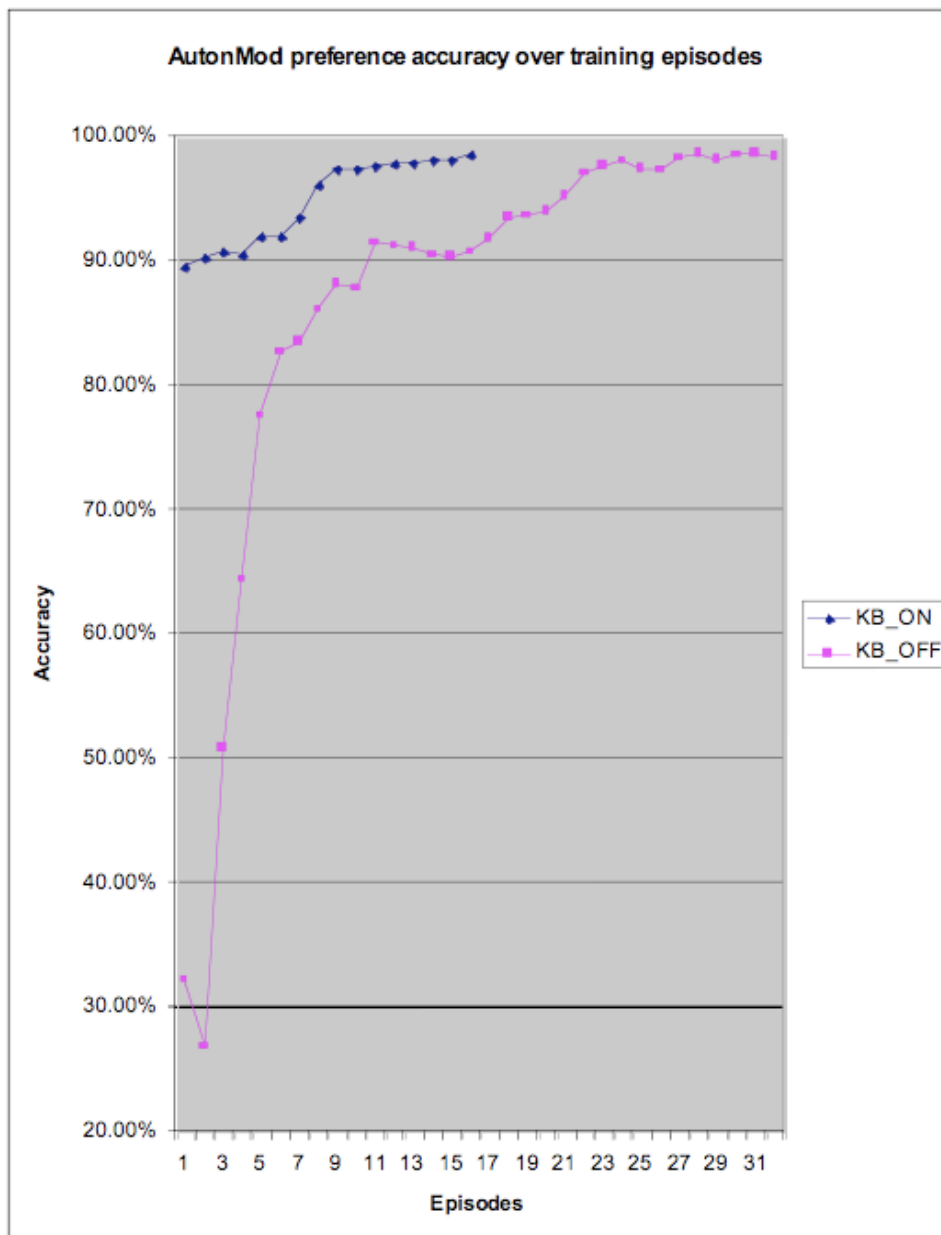


Figure 7: Experiment results for testing hypothesis (iv)

Acknowledgements

This material is based upon work supported by the DARPA/IPTO COORDINATORS program and the Air Force Research Laboratory under Contract No. FA8750-05-C-0030. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.