Validation of Cognitive Models for Collaborative Hybrid Systems with Discrete Human Input

Abraham P. Vinod, Yuqing Tang, Meeko M. K. Oishi, Katia Sycara, Christian Lebiere and Michael Lewis

***Abstract*— We present a method to validate a cognitive model, based on the cognitive architecture ACT-R, in dynamic human-automation systems with discrete human input. We are inspired by a two-choice game in which the human is tasked with maximizing net coverage of a robotic swarm that can operate under rendezvous or deployment dynamics. We model the human as a Markovian controller based on gathered experimental data, that is, a non-deterministic control input with known likelihoods of control actions associated with certain configurations of the state-space. We use reachability analysis to predict outcome of the resulting discrete-time stochastic hybrid system, in which outcome is defined as a function of the system trajectory. We suggest that the resulting expected outcomes can be used to validate the cognitive model against actual human subject data. We apply our method to the two-choice game with data from 10 human subjects. The novelty of this work is 1) a method to compute expected outcome in a hybrid dynamical system with a Markov chain model of the user's discrete choice, and 2) application of this method to validation of cognitive models with a database of actual human subject data.**

***Index Terms*— Reachability analysis, Cognitive models, ACT-R, Human-in-the-loop systems, Markov controller, Hybrid systems, Human-automation interaction**

I. Introduction

Human-automation systems are ubiquitous, not only in commercial products, but also in complex and safety critical systems, such as power grid distribution systems, transportation systems, and biomedical devices. While extensive work has been done on assuring safety and performance of fully automated systems, less is known about how to provide such assurances in human-in-the-loop systems. Indeed, human-automation interaction is often key for the overall performance of safety-critical human-automation systems. Recently, Google announced that 15 out of the 16 accidents involving Google's self-driving autonomous car were due to the other human drivers sharing the road [1]. Predicting expected behavior of human-automation systems is crucial for assuring safety as well as for optimizing performance.

This material is based upon work supported by the National Science Foundation. Oishi and Vinod are supported under Grant Number CMMI-1254990 (CAREER, Oishi) and CNS-1329878. Sycara, Ting, and Lebiere are supported under Grant Number CNS-1329986. Lewis is supported under Grant Number CNS-1329762. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Abraham P. Vinod and Meeko M. K. Oishi are with the Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131 USA; e-mail: aby.vinod@gmail.com, oishi@unm.edu (corresponding author)

Yuqing Tang, Katia Sycara and Christian Lebiere are with Carnegie Mellon University, Pittsburgh, PA 15213 USA; e-mail: {yuqing.tang,katia,cl}@cs.cmu.edu

Michael Lewis is with University of Pittsburgh, Pittsburgh, PA 15260 USA; email: ml@sis.pitt.edu

A variety of methods have been considered to model the human in human-automation systems. Early models [2], [3], [4] for human drivers and pilots were based on transfer functions gathered from experimental data, and enabled characterization of pilot-induced oscillations and other problematic behaviors. However, in modern cyber-physical systems, human inputs may include non-trivial combinations of low-level continuous inputs as well as high-level discrete inputs (e.g., as in supervisory control schemes) [5], [6]. Recent investigations have explored models of high-level human decision making [7], [8], [9] and attention allocation [10] to facilitate design of decision support systems [11], [12] for dynamical systems.

An alternative approach makes use of a cognitive architecture, a method of modeling human behavior based on known psychological phenomena. Computational unified theories of cognition, known as cognitive architectures, have been developed to implement aspects of cognition that do not vary across human subjects, including the mechanisms and structures through which information is processed. They include limitations on attention and working memory, cognitive biases driven by the statistics of the environment such as higher weight given to recent information, as well as statistical processes driving generalization such as similarity-based pattern matching. We have chosen ACT-R [13][14] as our target cognitive architecture because: (1) it has been used to develop a wide range of cognitive models, (2) it incorporates both cognitive capabilities and human limitations relevant to complex system control, (3) it provides constraints on human performance through its mechanisms, e.g., memory retrieval and generalization. Such cognitive models are based on "first-principles" of human cognition and hence can be argued to be more general than e.g., experimentally obtained transfer function models of a specific subject.

In this paper, we use the framework of discrete-time hybrid systems to model human-in-the-loop cyberphysical systems [15], and pose the question of expected outcome in terms of a stochastic reachability problem. We model the human as a non-deterministic Markov controller, generated from an ACT-R based cognitive model. To predict expected behavior of the system with respect to a desired outcome, we formulate a forward reachability problem. The resulting expected outcome can be compared against actual human subject data to validate a given cognitive model (e.g., a given Markov chain controller). We apply this method to a two-choice game, conducted over Amazon Turk [16], in which the human is tasked with maximizing net coverage of a robot swarm that can operate under rendezvous or deployment

dynamics. The novelty of this work is 1) a method to compute expected outcome in a hybrid dynamical system with a nondeterminsitic Markov controller that describes the human input, and 2) application of this method to validation of an ACT-R generated cognitive model with a database of actual human subject data.

The paper is organized as follows: We describe the mathematical formalism of the human-in-the-loop hybrid system and formulate the problem to be solved in Section II. Section III describes the calculation of expected outcome and its comparison to experimental data. In Section IV, we apply this method to a robotic swarm problem and discuss validity of the cognitive model for this experiment. Lastly, we offer conclusions and directions for future work in Section V.

II. Problem Formulation

The following mathematical preliminaries will be used throughout the paper. For $m \in \mathbb{N}$ and a set S, let S^m denote the Cartesian product of S taken with itself m times. As in [17], for a sample space Ω, X is a random variable if X is a Borel-measurable transformation from Ω to $\mathbb{R}$. X is defined on a measurable space $\{\Omega, \mathcal{F}(\Omega)\}$ where $\mathcal{F}(\Omega)$ denotes the sigma-algebra associated with a countable sample space Ω. When $\Omega \subset \mathbb{R}^m$, we denote the sigma-algebra of Ω as $\mathcal{B}(\Omega)$, the Borel-space associated with Ω. A probability measure μ may be assigned to the measurable space to obtain a probability space $\{\Omega, \mathcal{F}(\Omega), \mu\}$. For a m-dimensional random vector $\boldsymbol{X} = \{X_1, X_2, \cdots, X_m\}$ with X_i defined in $\{\Omega, \mathcal{F}(\Omega), \mu\}$ for $1 \leq i \leq m$, $\boldsymbol{X}$ is defined in the product space $\{\Omega^m, \mathcal{F}(\mathcal{F}(\Omega)^m), \mu_p\}$. Here, μ_p is the joint probability of the random variables that make up the random vector $\boldsymbol{X}$.

A. Discrete-time stochastic hybrid dynamical systems with discrete human input

Consider a discrete-time time-invariant hybrid system

$$x[k+1] = \boldsymbol{f}(q[k], x[k], u_a[k]) \tag{1}$$

$$q[k+1] = \boldsymbol{g}(q[k], x[k], u_h[k]). \tag{2}$$

with discrete state $q[k] \in Q \subset \mathbb{N}$, continuous state $x[k] \in \mathbb{X} \subset \mathbb{R}^n$, automation input $u_a[k] \in \mathbb{U}_A \in \mathcal{B}(\mathbb{R}^p)$, and human input $u_h[k] \in \mathbb{U}_H \in \mathcal{F}(\mathbb{N}^m)$. Note that the automation input only affects the continuous state evolution, and the human input only affects the evolution of the discrete state. Further, $\mathbb{U}_H$ is a finite set. We require $\boldsymbol{f}$ and $\boldsymbol{g}$ to be Borel-measurable functions to ensure measurability of the states, as well as Lipschitz continuity of $\boldsymbol{f}$. The hybrid state space is defined as $S \triangleq \bigcup_{q \in Q} \{q\} \times \mathbb{X}$, and $\pi_a : S \to \mathbb{U}_A$ is the deterministic control policy used by the automation.

The human input is modeled as an event (action) which causes the discrete state transitions as described by $\boldsymbol{g}$ (2). We further specify that the human input is modeled using a Markov chain, hence is defined as a stochastic map, $\pi_h : S \to \mathbb{U}_H$, such that

$$u_h[k] = \pi_h(q[k], x[k]). \tag{3}$$

The function $\pi_h(q[k], x[k])$ generates the random variable $u_h[k]$ based on the transition probabilities associated with the Markov chain. We denote the control action sequence of the human and the automation derived from the control laws π_h and π_a as

$$\boldsymbol{u}_h = \{u_h[0], \cdots, u_h[N-1]\} \in \mathbb{U}_H^N, \tag{4}$$

$$\boldsymbol{u}_a = \{u_a[0], \cdots, u_a[N-1]\} \in \mathbb{U}_A^N \tag{5}$$

respectively. Let $u_h[k]$ be defined in the probability space $\{\mathbb{U}_H, \mathcal{F}(\mathbb{U}_H), Pr\}$.

We can equivalently write (1), (2), (3) in standard form as a discrete-time stochastic hybrid system (DTSHS) [15],

$$\mathcal{H} = (Q, \mathbb{X}, \mathbb{U}_A, \mathbb{U}_H, T_x, T_q) \tag{6}$$

with elements defined as follows:

- $Q \subset \mathbb{N}$ is the finite set of discrete states of the system.
- $\mathbb{X} \subset \mathbb{R}^n$ is the set of continuous states, $S \triangleq \bigcup_{q \in Q} \{q\} \times \mathbb{X}$.
- $\mathbb{U}_A \in \mathcal{B}(\mathbb{R}^m)$ is the compact control space for the automation, with deterministic controller, π_a.
- $\mathbb{U}_H \in \mathcal{F}(\mathbb{N}^p)$ is a finite set of choices for the human, generated using π_h (3).
- $T_x : S \times \mathbb{U}_A \times \mathcal{B}(\mathbb{X}) \to [0,1]$ gives a probability measure $T_x(.|s, u_a)$ for each $s \in S$ and $u_a \in \mathbb{U}_A$ in the Borel space $\{S, \mathcal{B}(S)\}$. Since the continuous state evolution is deterministic (1), we define T_x as an impulse function:

  $$T_x(x|s, u_a) = \delta(x - \boldsymbol{f}(s, u_a)) = \delta(x - \boldsymbol{f}(s, \pi_a(s)))$$

  where $\delta(y) : \mathbb{R} \to \{0, 1\}$ is 1 if $y = 0$ and is 0 otherwise.
- $T_q : S \times \mathbb{U}_H \times Q \to [0,1]$ gives a probability distribution $T_q(.|s, u_h)$ over Q for each $s \in S$ and $u_h \in \mathbb{U}_H$. For any $q \in Q$, T_q is a function of the transition probabilities of π_h,

  $$T_q(q|s, u_h) = T_q(q|s, \pi_h(s)) = Pr\{q = \boldsymbol{g}(s, \pi_h(s))\}.$$

  Hence T_q captures the effect of human action.

The initial condition $s_0 = \{q_0, x_0\} \in S$ is either known or may be sampled from a known probability distribution [15], and the time horizon is N. Note that restrictions, for example, to disallow human events in various circumstances, can be modeled in T_q. We present a general framework in (6), although we are primarily interested in systems in which human actions occur infrequently.

The hybrid system execution is $\boldsymbol{\xi}(\cdot; s_0, \boldsymbol{u}_h, \boldsymbol{u}_a) = \{\xi[1], \cdots, \xi[N]; s_0, \boldsymbol{u}_h, \boldsymbol{u}_a\}$ where for $k \in \{0, \cdots, N\}$,

$$\xi[k] = (\xi_q[k], \xi_x[k]). \tag{7}$$

Here, $\xi_q[k]$ is the discrete mode at the k^{th} instant and $\xi_x[k]$ is the continuous state at the k^{th} instant. Note that $\boldsymbol{u}_h$ is a N-dimensional random vector defined by (3). Hence,

$$\boldsymbol{\xi}_q(\cdot; s_0, \boldsymbol{u}_h, \boldsymbol{u}_a) \triangleq \{\xi_q[1], \cdots, \xi_q[N]\}, \tag{8}$$

and $\boldsymbol{\xi}_q(\cdot; s_0, \boldsymbol{u}_h, \boldsymbol{u}_a)$ is a N-dimensional random vector defined in the probability space $\{Q^N, \mathcal{F}(Q^N), \mathbb{P}_{s_0}\}$. Similarly,

$$\boldsymbol{\xi}_x(\cdot; s_0, \boldsymbol{u}_h, \boldsymbol{u}_a) \triangleq \{\xi_x[1], \cdots, \xi_x[N]\}, \tag{9}$$

and $\boldsymbol{\xi}_x(\cdot; s_0, \boldsymbol{u}_h, \boldsymbol{u}_a)$ is a N-dimensional random vector defined in the probability space $\{\mathbb{X}^N, \mathcal{B}(\mathbb{X}^N), \mathbb{P}_{s_0}\}$. Here, $\mathbb{P}_{s_0}$ refers to the joint probability of the discrete states for a given initial state $s_0 \in S$ [17]. From (3), $\boldsymbol{u}_h$ is also defined in the probability space $\{\mathbb{U}_H^N, \mathcal{F}(\mathbb{U}_H^N), \mathbb{P}_{s_0}\}$.

Let $\hat{\boldsymbol{u}}_h$ be an instantiation of the non-deterministic policy $\boldsymbol{u}_h$ modeling the human, i.e, $\hat{\boldsymbol{u}}_h$ is a sequence of actions taken by the human. For a initial condition $s_0 \in S$, we can define instantiations of $\boldsymbol{\xi}_q$ and $\boldsymbol{\xi}_x$ as

$$\boldsymbol{q} = \boldsymbol{\xi}_q(.; s_o, \hat{\boldsymbol{u}}_h, \boldsymbol{u}_a),\ \boldsymbol{x} = \boldsymbol{\xi}_x(.; s_o, \hat{\boldsymbol{u}}_h, \boldsymbol{u}_a). \tag{10}$$

when a particular instantiation of $\boldsymbol{\xi}_q$ is considered as given by (1). The likelihood of a given trajectory $(\boldsymbol{q}, \boldsymbol{x})$ starting from s_0 is

$$\begin{aligned}\mathbb{P}_{s_0}\Big\{\boldsymbol{\xi}(\cdot; s_0, \boldsymbol{u}_h, \boldsymbol{u}_a) = (\boldsymbol{q}, \boldsymbol{x})\Big\} &= \mathbb{P}_{s_0}\Big\{\boldsymbol{\xi}_q = \boldsymbol{q}\Big\} \\ &= \prod_{n=1}^{N} T_q(q_{n-1}, x_{n-1}, q_n).\end{aligned} \tag{11}$$

where $\boldsymbol{x} = \{x_1, \cdots, x_N\}$ and $\boldsymbol{q} = \{q_1, \cdots, q_N\}$. We obtain T_q from a cognitive model [18] .

B. Cognitive models

The cognitive model is implemented in a neurally-inspired cognitive architecture, ACT-R, and follows the instance-based learning (IBL) methodology. Decisions in IBL are made primarily based on experiences of a task and its associated reward function. These experiences are stored as "chunks" of ACT-R's declarative memory, with each chunk corresponding to a relevant experience. Instance chunks typically contain a description of the context in which each decision is made, the decision itself, and the outcome of that decision. Important features influencing the decision are identified to minimally express the context.

The cognitive model estimates the reward function using ACT-R's blending mechanism [19] for interpolation (partial matching) of the chunks representing instances stored in the ACT-R's declarative memory. The interpolation is accomplished using a linear similarity-based pattern matching function between the features, as well as using the activation of retrieval functions of ACT-R declarative memory based on recency and noise. The model then selects the decision that produces the larger reward. Finally, ACT-R compares the resulting reward to an internally generated estimate for the current trial. In short, ACT-R models a human who aims to maximize a reward function for a system given by (1) and (2).

The information stored in the ACT-R cognitive model is characterized by the set of instance chunks and their activation. Environment and system observations change the activation levels of the existing memory chunks as well as add new chunks to the model. Abstracting the information contained in the cognitive model's declarative memory into an analytical model requires Monte Carlo simulations of the cognitive model. The resulting trajectories of activation levels of the memory chunks can be clustered to produce the Markov model; the clustering is left to the modeler's discretion. The transition probabilities in the analytical model are also extracted from the Monte Carlo runs [18], and constitute the Markov chain (3).

C. Problem statement

We define a performance metric as a Borel-measurable reward function, $r : S^{N+1} \to \mathbb{R}$, which assigns a reward to every system execution. We also define a specific reward function, $R : S \times \mathbb{U}_H^N \to \mathbb{R}$, which captures the reward attainable from $s_0 \in S$ for the human action sequence $\boldsymbol{u}_h$. Since $\boldsymbol{u}_a$ is a known, deterministic automation control policy,

$$R(s_0, \boldsymbol{u}_h) = r(s_0, \boldsymbol{\xi}(\cdot; s_0, \boldsymbol{u}_h, \boldsymbol{u}_a)). \tag{12}$$

From (12), $R(s_0, \boldsymbol{u}_h)$ is a random variable defined in the probability space $\{\mathbb{R}, \mathcal{B}(\mathbb{R}), \mathbb{P}_{s_0}\}$, induced from the probability space defined for $\boldsymbol{u}_h$.

The following two problems are addressed in this paper.

Problem 1: Given a discrete-time stochastic hybrid system $\mathcal{H}$ (6) with discrete human input captured by a Markov model (3), and a performance metric R (12), find the expected performance of the system for a typical human subject.

We additionally examine a class of reward functions, functions of the forward reach tube, which provides a simplified solution to Problem 1. Once we have computed expected performance, we can pose the following validation problem with respect to a cognitive model.

Problem 2: Determine the validity of a cognitive model by comparing expected outcome (Problem 1) based on cognitive model with actual outcome from human subject experiments.

III. Computing Expected Outcome via Forward Reachability

A. Expected outcome

The performance for a typical human operator can be defined in terms of the expectation of the reward function (12):

$$\begin{aligned}\mathbb{E}_{s_0}\left[R(s_0, \cdot)\right] \quad = \quad & \sum_{\boldsymbol{q} \in Q^N} \Big[\mathbb{P}_{s_0}\Big\{\boldsymbol{u}_h = \hat{\boldsymbol{u}}_h\Big\} \\ & \times\ r\big(s_0, \boldsymbol{\xi}(\cdot; s_0, \hat{\boldsymbol{u}}_h, \boldsymbol{u}_a)\big)\Big]\end{aligned} \tag{13}$$

While equation (13) theoretically solves Problem 1 for any $R(\cdot)$ that is a function of the system execution, numerical computation may be challenging for the most general case.

We consider the specific case when the reward function is restricted to be a function of the forward reach tube for a known human input,

$$R_T(I, \hat{\boldsymbol{u}}_h) = J\big(ReachTube(I, \hat{\boldsymbol{u}}_h)\big) \tag{14}$$

where $J : \mathcal{B}(S) \to \mathbb{R}$ is a set function defined on S, and the forward reachable tube

$$ReachTube(t, I, \hat{\boldsymbol{u}}_h) = \bigcup_{0 \le \tau \le t} Reach(\tau, I, \hat{\boldsymbol{u}}_h) \subseteq S \quad (15)$$

is the union of forward reach sets over all time steps until $t \le N$. The forward reach set, $Reach(\tau, I, \hat{\boldsymbol{u}}_h)$, is those states that can be reached at exactly time τ.

$$\begin{aligned} Reach(\tau, I, \hat{\boldsymbol{u}}_h) \quad = \quad & \{y \in S : \exists s_0 \in I, \text{ s.t.} \\ & \boldsymbol{\xi}(\tau; s_0, \hat{\boldsymbol{u}}_h, \boldsymbol{u}_a) = y\} \subseteq S \end{aligned} \quad (16)$$

While computation of (15), (16) is in general nontrivial because all instantiations of $\boldsymbol{\xi}_q$ must be explored, the particular form of hybrid system we consider lends itself to realistic computation in specific cases.

First, we note that in the trivial case in which the initial set $\mathcal{I}$ is a singleton; computing (16) can be obtained through simulation. Second, we can exploit the fact that in many systems, human input may occur infrequently.

Since the stochasticity in (6) is due solely to the human input, we consider portions of the trajectories (7) that result from a single human action, that is, that start when one human action occurs but before the next human input occurs. Let $\boldsymbol{v}_h(\tau) = \{v, 0, 0, \cdots, 0\} \in \mathbb{U}_h^\tau$ be an input sequence of length τ following some non-zero human input $v \in \mathbb{U}_H$, with $u_h = 0$ indicating that no human input occurs. Hence the characterizing the reach set and reach tube between human-triggered events becomes a determinsitic reachability problem that can exploit work in developing efficient computational tools [20], [21], [22], [23], [24] for certain classes of dynamics and initial sets.

For ease of notation, we define

$$Reach_{\text{auto}}(\tau, I, v) = Reach(\tau, I, \boldsymbol{v}_h(\tau)) \quad (17)$$
$$ReachTube_{\text{auto}}(t, I, v) = ReachTube(t, I, \boldsymbol{v}_h(t)) \quad (18)$$

Note that this framework can be extended to cost functions which involve more than just the reachable tube, e.g., invariant sets, viable sets, and others.

B. Computing forward reach tube for systems with infrequent human actions

Let $\Delta t(\boldsymbol{u}_h) = \{dt_0, dt_1, \cdots, dt_{N_1}\}$ be the set of time intervals starting with a human input up till the next human input (not included) with $N_1 \le N$. Since there may or may not be a human input at $t = 0$, we define dt_0 as the time interval from $k = 0$ to the first human input (not included). Since $\boldsymbol{u}_h$ is a N-dimensional vector, $\sum_{i=0}^{N_1} dt_i = N$. We elucidate the definition of Δt with this example: Consider $\mathbb{U}_H = \{0, 1, 2\}$, $N = 6$ and $\hat{\boldsymbol{u}}_h = \{0, 2, 0, 0, 0, 1\}$. For this particular $\hat{\boldsymbol{u}}_h$, $\Delta t(\hat{\boldsymbol{u}}_h) = \{1, 4, 1\}$ where $dt_i = \text{len}(TimeInterval)$. Hence, $dt_0 = \text{len}([0])$, $dt_1 = \text{len}([1, 2, 3, 4])$ and $dt_2 = \text{len}([5])$. Computation of the corresponding $ReachTube(N, I, \hat{\boldsymbol{u}}_h) = \cup_{0 \le i \le 2} RT_i$ where RT_i is defined as:

$$RT_0 = ReachTube_{\text{auto}}(dt_0, I, 0), \quad (19)$$
$$RT_1 = ReachTube_{\text{auto}}(dt_1, RS_0, 2) \quad (20)$$
$$RT_2 = ReachTube_{\text{auto}}(dt_2, RS_1, 1) \quad (21)$$

where $RS_0 = Reach_{\text{auto}}(dt_0, I, 0)$ and $RS_1 = Reach_{\text{auto}}(dt_1, RS_0, 2)$. Note that, by (15), $RT_0 = Reach_{\text{auto}}(0, I, 0) \cup Reach_{\text{auto}}(1, I, 0)$. In general, the $ReachTube(\cdot)$ for (6) with infrequent human inputs can be computed using Algorithm 1.

Algorithm 1 Computing the reach tube (15) for DTSHS with discrete human input (6) and occasional events

Input: $I \subseteq S$, Δt, $\hat{\boldsymbol{u}}_h$, $ReachTube_{\text{auto}}(dt, I, v)$
Output: $RTube$ – the forward reach tube as defined in (15)

```
procedure ReachTubeCompute(I, û_h, Δt)
    i, t ← 0                          ▷ Initialize iteration variables
    RTube ← ∅
    I_i ← I
    while i ≤ N_1 do
        RT_i ← ReachTube_auto(Δt[i], I_i, û_h[t])
        I_i ← Reach_auto(Δt[i], I_i, û_h[t])
        RTube ← RTube ∪ RT_i          ▷ Using (15) and (18)
        t ← t + Δt[i]                 ▷ Go to next decision epoch
        i ← i + 1                     ▷ Update iteration variable
    end while
end procedure
```

For a given Δt, Algorithm 1 can be used to compute the reward function (14) for systems with occasional events efficiently using existing tools.

C. Validation of the cognitive model

The cognitive model that provides π_h enables calculation of the expected outcome (13). We solve Problem 2 by comparing the expected outcome with the experimental data. We compute the sample mean of the rewards obtained during human trials,

$$SampleMean[R(s_0)] = \frac{1}{N_{obs}} \sum_{i=1}^{N_{obs}} R(s_0) \quad (22)$$

where N_{obs} is the number of experiments associated with each initial configuration s_0.

For each initial configuration in the experiment, we compare the expected outcome to the observed typical outcome, and analyze the difference.

$$\epsilon(s_0) \triangleq \left(\mathbb{E}_{s_0}[R(s_0)] - SampleMean[R(s_0)]\right). \quad (23)$$

The validation of the cognitive model is done by computing the mean (bias), the variance, and the maximum absolute value of $\epsilon(\cdot)$ over all initial configurations in the experiment. Evaluating whether these derived metrics are within the specifications of the problem completes the validation.

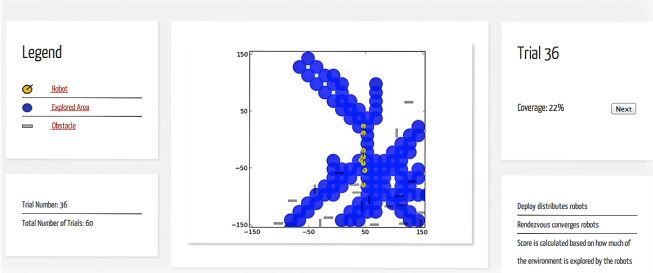


Fig. 1: User interface for the robotic swarm experiment [18]

IV. Robotic swarm two-choice game

A. Experimental Setup

The robotic swarm two-choice game involves a swarm of twenty simulated point-mass robots [18]. The human controls the swarm indirectly by choosing between two strategies, which correspond to different robot dynamics: rendezvous or deploy. The rendezvous strategy causes the robots to converge to a common location, and the deploy strategy causes the robots to diverge to different locations in the game space. The simulated environment also contains a set of twenty fixed obstacles. Each robot has an associated sensing area, that is, a circular area (due to, e.g., omni-directional sensors) that the robot can 'cover'. The participant's goal in each trial is to select the strategy that maximizes the accumulated area sensed. The only information that the participant has to make this decision is the initial configuration of the robots and obstacles.

The game interface is shown in Figure 1, and additional details can be found in [18]. Ten participants were recruited from Amazon Turk [16]. Each participant was presented with sixty trials. In each trial, the participant was shown the initial configuration and asked to choose between the two strategies. In the first ten trials (training trials), the outcomes using both the strategies were shown. In the remaining fifty trials, the participants were only shown the outcome for the strategy they selected.

B. Hybrid human-in-the-loop system

We model the human-swarm system as the tuple $\mathcal{H} = (Q, \mathbb{X}, \mathbb{U}_A, \mathbb{U}_H, T_x, T_q)$, with:

- $Q = \{\text{Rendezvous}, \text{Deploy}\}$
- $\mathbb{X} = \mathcal{A}^{20}$, with $\mathcal{A} = [-150, 150] \times [-150, 150]$ the simulated environment with 20 robots.
- $\mathbb{U}_A \in \mathcal{B}(\mathbb{R}^{40})$, with feedback-based controller $\pi_a : S \to \mathbb{U}_A$.
- $\mathbb{U}_H = \{D, R\}$ corresponds to selection of deploy or rendezvous. The participant's choice is described by $\pi_h : \mathbb{X} \to \mathbb{U}_H$.
- $T_x : S \times \mathbb{U}_A \times \mathcal{B}(\mathbb{X}) \to [0, 1]$ is given by

$$T_x(x|s, u_a) = \delta(x - \boldsymbol{f}(s, \pi_a(s)))$$

- $T_q(q|s) = \delta(q - q[0])$.

Note that T_q permits human input only at the start of the trial; once the participant has made this one decision, no further human input is allowed. The index set of the robots is $\Gamma = \{1, 2, \cdots, 20\}$ and $\mathcal{O} \in \mathbb{X}$ is the set of the locations of the obstacles.

A cognitive model was constructed using ACT-R, based on the data from the 10 training trials. The mean of the initial position of the robots (eccentricity) and its variance (dispersion) were used to characterize the context of each trial [18]. ACT-R's blending mechanism is used to estimate the area coverage under either strategies [19]. The decision of the model is made based on the mode which produced the largest accumulated coverage estimate. As discussed in the experimental setup, the model receives feedback about the true accumulated coverage and uses it to update its estimates. Upon completion of the trial, the representation of the problem is added as a new chunk in ACT-R's declarative memory. The cognitive model is initiated with 10 training instances and is built with up to 60 instances by the conclusion of the experiment. A Markov model π_h was extracted from the data generated by the cognitive model via standard Markov model training [18]. Hence for $x[0] \in \mathbb{X}$, $\pi_h(x[0])$ determines the discrete state, which remains unchanged throughout the execution.

The continuous state is $x[k] \triangleq [\boldsymbol{p}_1[k], \boldsymbol{p}_2[k], \cdots, \boldsymbol{p}_{20}[k]] \in \mathbb{R}^{40}$, with position $\boldsymbol{p}_i[k] = [x_i[k]\ y_i[k]]^\top$ for the i^{th} robot. The dynamics for the i^{th} robot in Deploy mode are described by

$$\begin{aligned}\boldsymbol{p}_i[k+1] &= \boldsymbol{f}_{nl,i}(\text{Deploy}, x[k]) && (24)\\ &= \boldsymbol{p}_i[k] + \alpha_1(\boldsymbol{c}_i[k] - \boldsymbol{p}_i[k]) && (25)\\ &\quad + \alpha_2 \sum_{\boldsymbol{p}_{obs} \in \mathcal{O}} \frac{\boldsymbol{p}_i[k] - \boldsymbol{p}_{obs}[k]}{\|\boldsymbol{p}_i[k] - \boldsymbol{p}_{obs}[k]\|^2} && (26)\end{aligned}$$

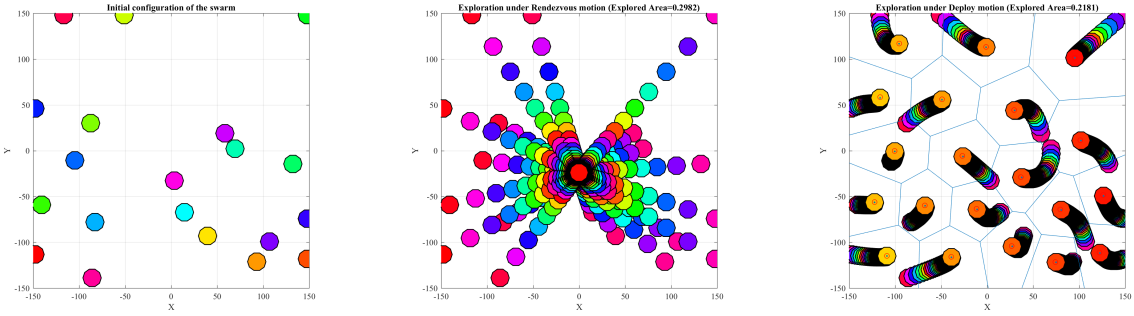
with $\boldsymbol{c}_i[k] \in \mathcal{A}$ the centroid of the Voronoi cell associated with i^{th} robot [25]. The dynamics in Rendezvous mode are

$$\begin{aligned}\boldsymbol{p}_i[k+1] &= \boldsymbol{f}_{nl,i}(\text{Rendezvous}, x[k]) && (27)\\ &= \boldsymbol{p}_i[k] + \frac{\alpha_3}{20}\left(\sum_{j \in (\Gamma \setminus \{i\})} (\boldsymbol{p}_j[k] - \boldsymbol{p}_i[k])\right)\\ &\quad + \alpha_4 \sum_{\boldsymbol{p}_{obs} \in \mathcal{O}} \frac{\boldsymbol{p}_i[k] - \boldsymbol{p}_{obs}[k]}{\|\boldsymbol{p}_i[k] - \boldsymbol{p}_{obs}[k]\|^2} && (28)\end{aligned}$$

with controller gains $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\} \in \mathbb{R}^4$. However, since obstacle locations are not shown to not influence the participant's mode selection [18], we ignore the obstacles (by setting $\alpha_2 = \alpha_4 = 0$) to obtain the simplified dynamics

$$\begin{aligned}\boldsymbol{f}_i(\text{Deploy}, x[k]) &= \boldsymbol{p}_i[k] + \alpha_1(\boldsymbol{c}_i[k] - \boldsymbol{p}_i[k]) && (29)\\ \boldsymbol{f}_i(\text{Rendezvous}, x[k]) &= \boldsymbol{p}_i[k]\\ &\quad + \frac{\alpha_3}{20}\left(\sum_{j \in (\Gamma \setminus \{i\})} (\boldsymbol{p}_j[k] - \boldsymbol{p}_i[k])\right) && (30)\end{aligned}$$

where $\boldsymbol{f}_i : Q \times \mathbb{X} \to \mathcal{A}$ and $i \in [1, 20]$. Note that $\boldsymbol{c}_i[k]$ makes $\boldsymbol{f}$ nonlinear in Deploy mode.



(a) Initial configuration of the robot swarm marked without obstacles ($s_0 \in S$)

(b) Robot swarm execution in $q[0] = \text{Rendezvous}$

(c) Robot swarm execution in $q[0] = \text{Deploy}$ (Centroidal Voronoi tessellation of the $x[N]$ is also marked)

Fig. 2: Obstacle-free simulation of the robot swarm in Rendezvous and Deploy modes.

The performance metric of interest is the *accumulated* area coverage. Unlike the typical area coverage metric (see [25, Equation 2.3.1]), accumulated area coverage is the net area covered throughout the entire system execution (not just at a given instant). We define the performance metric

$$R(s_0) = \frac{\mathrm{m}(SensedS)}{\mathrm{m}(\mathcal{A})} \tag{31}$$

with Lebesgue measure $\mathrm{m}(\cdot)$ and the set of states

$$\begin{aligned} SensedS(s_0) \quad = \quad & \{p \in \mathcal{A} : \exists\ h \in ReachTube(s_0) \\ & \text{s.t. } \|p - h\|_2 \leq c_s\}\} \end{aligned} \tag{32}$$

sensed by the swarm. The omni-directional sensor has radius $c_s > 0$. Since (32) is a function a single initial configuration, the calculation of $ReachTube(s_0)$ can be accomplished through simulation.

Figure 2 shows the evolution of the system under Deploy and Rendezvous modes, generated using MPT [21], for a fixed initial condition. As shown in Figure 3, for 38 configurations out of 50, the model performs very close to the typical human performance, $|\epsilon(\cdot)| \leq 0.01$. The bias, variance, and maximum absolute difference of $\epsilon(\cdot)$ were used for validation since they characterize properties of the cognitive model: unbiasedness, mean square error, and the magnitude of error in the outliers, respectively. The bias and variance of $\epsilon(\cdot)$ across the configurations tested is very low, as seen in Table I, indicating that that the cognitive model is unbiased with respect to the configurations considered, and has very low mean square error. While these metrics should be an indication that the cognitive model performs very close to a typical human performance in general, the maximum absolute difference of approximately 0.026 suggests that there might be certain configurations where the cognitive model may not predict the typical human performance within the tolerance limits.

Mean (Bias)	Variance	Maximum absolute difference
2.42×10^{-4}	8.61×10^{-5}	0.0257

TABLE I: Characterization of prediction error $\epsilon(\cdot)$ (23) across 50 configurations with 10 participants.

V. Conclusions and Future Work

We model human-automation systems using a discrete-time stochastic hybrid systems with stochastic discrete human inputs. We suggest a method to compute the expected outcome for reachable set-based reward functions, and apply this method to the validation of a cognitive model for a robotic swarm problem.

In future work, we will consider reward functions requiring safety (viability) and safety with guarantees (reach-avoid), and to use the reachable set calculations to predict conflicts in collaborative human-automation systems.

References

[1] M. Richtel and C. Dougherty, "Google's driverless cars run into problem: Cars with drivers," September 2015. [Online]. Available: http://www.nytimes.com/2015/09/02/technology/personaltech/google-says-its-not-the-driverless-cars-fault-its-other-drivers.html?_r=0

[2] P. Fitts, *Human engineering for an effective air navigation and traffic control system*. Columbus, OH: Ohio State University Research Foundation, 1951.

[3] D. McRuer, "Human dynamics in man-machine systems," *Automatica*, vol. 16, no. 3, pp. 237–253, 1980.

[4] R. Hess, "Human-in-the-loop control," in *The Control Handbook*, W. Levine, Ed. CRC Press, Inc., 1996, pp. 1497–1505.

[5] J. Bailleiul, N. E. Leonard, and K. Morgansen, "Interaction dynamics: The interface of humans and smart machines," *Proceedings of the IEEE*, vol. 100, no. 3, pp. 567–570, 2012.

[6] M. Oishi, D. Tilbury, and C. J. Tomlin, "Guest editorial: Special section on human-centered automation," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 4–6, January 2016.

[7] K. Savla and E. Frazzoli, "A dynamical queue approach to intelligent task management for human operators," *Proceedings of the IEEE*, vol. 100, no. 3, pp. 672–686, 2012.

[8] A. Stewart, M. Cao, A. Nedic, and N. E. Leonard, "Towards human-robot teams: Model based analysis of human decision making in two-alternative choice tasks with social feedback," *Proceedings of the IEEE*, vol. 100, no. 3, pp. 751–775, 2012.

[9] P. Reverdy and N. E. Leonard, "Parameter estimation in softmax decision-making models with linear objective functions," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 54–67, January 2016.

[10] V. Srivastava, A. Surana, and F. Bullo, "Adaptive attention allocation in human-robot systems," Montreal, Canada, June 2012, pp. 2767–2774.

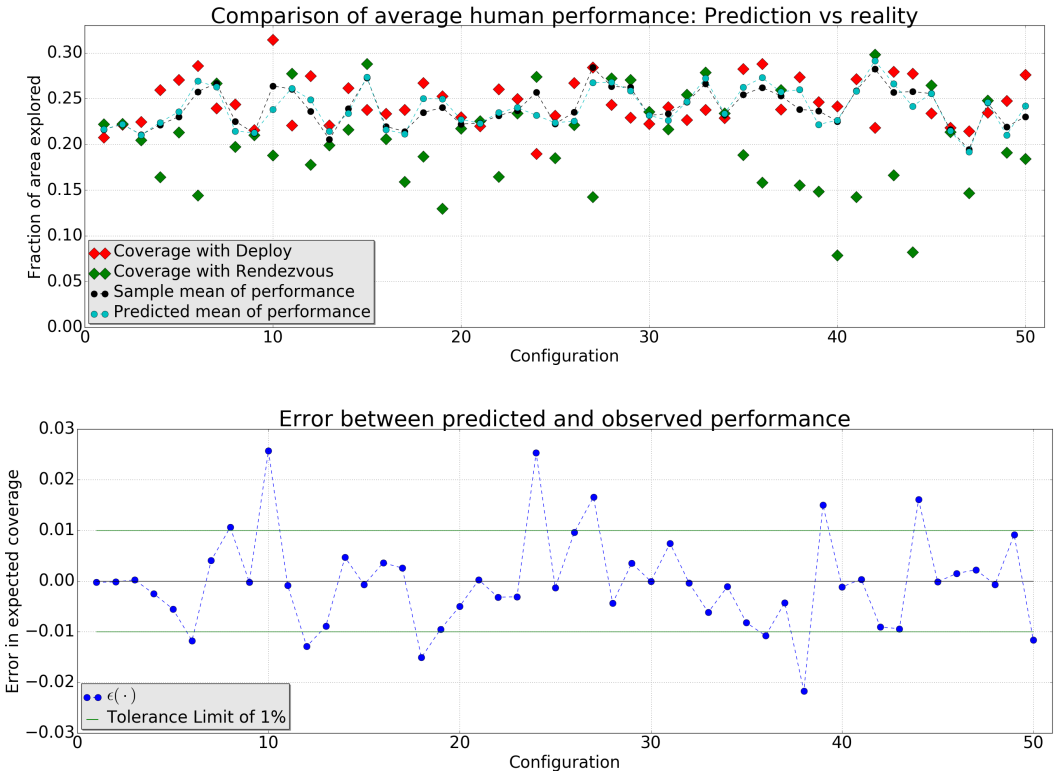


Fig. 3: Validation of the human cognitive model

[11] M. Forghani, J. M. McNew, D. Hoehener, and D. D. Vecchio, "Design of driver-assist systems under probabilistic safety specifications near stop signs," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 1, pp. 43–53, January 2016.

[12] X. Ding, M. Powers, M. Egerstedt, R. Young, , and T. Balch, "Executive decision support," *IEEE Robotics and Automation Magazine*, vol. 16, no. 2, pp. 73–81, June 2009.

[13] J. R. Anderson and C. Lebiere, *The Atomic Components of Thought*. Lawrence Erlbaum Associates, Mahwah, NJ, 1998.

[14] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, "An integrated theory of the mind." *Psychological review*, vol. 111, no. 4, p. 1036, 2004.

[15] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.

[16] A. Inc, "Amazon mechanical turk." [Online]. Available: https://www.mturk.com/mturk/welcome

[17] Y. S. Chow and H. Teicher, *Probability theory*. Springer, 1997.

[18] K. Sycara, C. Lebiere, Y. Pei, D. Morrison, Y. Tang, and M. Lewis, "Abstraction of analytical models from cognitive models of human control of robotic swarms," *International Conference on Cognitive Modeling*, 2015.

[19] C. Lebiere, "The dynamics of cognitive arithmetic," *Kognitionswissenschaft*, vol. 8, no. 1, pp. 5–19, 1999.

[20] M. Althoff, O. Stursberg, and M. Buss, "Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes," *Nonlinear analysis: hybrid systems*, vol. 4, no. 2, pp. 233–249, 2010.

[21] M. Kvasnica, P. Grieder, and M. Baotić, "Multi-Parametric Toolbox 2.6.3 (MPT)," 2004. [Online]. Available: http://control.ee.ethz.ch/ mpt/

[22] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis." London, UK: Springer-Verlag, 2000, pp. 202–214.

[23] C. Le Guernic and A. Girard, "Reachability analysis of linear systems using support functions," *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 250–262, 2010.

[24] I. M. Mitchell, "A toolbox of level set methods," *UBC Department of Computer Science Technical Report TR-2007-11*, 2007. [Online]. Available: http://www.cs.ubc.ca/%7Emitchell/ToolboxLS

[25] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, ser. Applied Mathematics Series. Princeton University Press, 2009.