A Direct Approximation of AIXI Using Logical State Abstractions

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose a practical integration of logical state abstraction with AIXI, a Bayesian optimality notion for reinforcement learning agents, to significantly expand the model class that AIXI agents can be approximated over to complex history-dependent and structured environments. The state representation and reasoning framework is based on higher-order logic, which can be used to define and enumerate complex features on non-Markovian and structured environments. We address the problem of selecting the right subset of features to form state abstractions by adapting the Φ-MDP optimisation criterion from state abstraction theory. Exact Bayesian model learning is then achieved using a suitable generalisation of Context Tree Weighting over abstract state sequences. The resultant architecture can be integrated with different planning algorithms. Experimental results on controlling epidemics on large-scale contact networks validates the agent's performance.

1 Introduction

The AIXI agent [1] is a mathematical solution to the general reinforcement learning problem that combines Solomonoff induction [2] with sequential decision theory. At time t, after observing the observation-reward sequence $or_{1:t-1}$ and performing previous actions $a_{1:t-1}$ the AIXI agent chooses action a_t given by

$$a_t = \arg\max_{a_t} \sum_{or_t} \ldots \max_{a_{t+m}} \sum_{or_{t+m}} [r_t + \ldots + r_{t+m}] \sum_{q:U(q,a_{1:t+m})=or_{1:t+m}} 2^{-l(q)}, \tag{1}$$

where U is a universal Turing machine, $U(q, a_{1:n})$ is the output of U given program q and input $a_{1:n}$, $m \in \mathbb{N}$ is the lookahead horizon, and $l(q)$ is the length of q in bits. [1] shows that AIXI's environment model converges rapidly to the true environment and its policy is pareto optimal and self-optimising.

While mathematically elegant, a fundamental issue with the AIXI agent is that it is only asymptotically computable. The question of whether a good practical approximation of AIXI can be constructed remains an open problem twenty years after AIXI was first introduced [3]. The most direct approximation of AIXI given so far is [4], in which the Bayesian mixture over all Turing machines is approximated using the Context Tree Weighting algorithm [5] for maintaining a Bayesian mixture over all n-Markov environments, and the expectimax search is approximated using ρUCT, a Monte Carlo planning algorithm [6]. This paper extends [4] in several ways. To move beyond n-Markov environments and simultaneously make it easy for agent designers to incorporate background knowledge, we adopt a formal knowledge presentation and reasoning framework based on higher-order logic that allows complex features on non-Markovian and structured environments to be defined and enumerated. We then adapt the Feature Reinforcement Learning framework [7] to define the optimisation problem associated with picking the right set of features to form state abstractions and present an algorithm based on Binary Decision Diagrams to approximate the solution. Running binarised CTW [8] and ρUCT, both suitably generalized, over the resultant abstract

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

state-action-reward sequences then gives us the richest, direct approximation of AIXI known to date. The practicality of the resultant agent is shown on epidemic control on non-trivial contact networks.

2 Background and Related Work

2.1 The General Reinforcement Learning Setting

We consider finite action, observation and reward spaces denoted by $\mathcal{A}, \mathcal{O}, \mathcal{R}$ respectively. The agent interacts with the environment in cycles: at any time, the agent chooses an action from $\mathcal{A}$ and the environment returns an observation and reward from $\mathcal{O}$ and $\mathcal{R}$. Frequently we will be considering observations and rewards together, and will denote $x \in \mathcal{O} \times \mathcal{R}$ as a percept x from the percept space $\mathcal{O} \times \mathcal{R}$. We will denote a string $x_1 x_2 \ldots x_n$ of length n by $x_{1:n}$ and its length $n-1$ prefix as $x_{<n}$. An action, observation and reward from the same time step will be denoted aor_t. After interacting for n cycles, the interaction string $a_1 o_1 r_1 \ldots a_n o_n r_n$ (denoted $aor_{1:n}$ from here on) is generated. We define the history space $\mathcal{H}$ to be an interaction string of any length.

Definition 1. A history h is an element of the history space $\mathcal{H} := (\mathcal{A} \times \mathcal{O} \times \mathcal{R})^*$.

Definition 2. An environment ρ is a sequence of probability distributions $\{\rho_0, \rho_1, \rho_2, \ldots\}$, where $\rho_n : \mathcal{A}^n \to \mathcal{D}((\mathcal{O} \times \mathcal{R})^n)$, that satisfies $\forall a_{1:n}\ \forall or_{<n}\ \rho_{n-1}(or_{<n}|a_{<n}) = \sum_{or \in \mathcal{O}\times\mathcal{R}} \rho_n(or_{1:n}|a_{1:n})$.

We will drop the subscript on ρ_n when the context is clear.

The predictive probability of the next percept given history and a current action is given by $\rho(or_n|aor_{<n}, a_n) = \rho(or_n|h_{n-1}a_n) := \frac{\rho(or_{1:n}|a_{1:n})}{\rho(or_{<n}|a_{<n})}$ for all $aor_{1:n}$ such that $\rho(or_{<n}|a_{<n}) > 0$.

The general reinforcement learning problem is for the agent to learn a *policy* $\pi : \mathcal{H} \to \mathcal{D}(A)$ mapping histories to a distribution on possible actions that will allow it to maximise its future expected reward. In this paper, we consider the future expected reward up to a finite horizon $m \in \mathbb{N}$.

Definition 3. Given a history h and a policy π the value function with respect to the environment ρ is given by $V_\rho^\pi(h_t) := \mathbb{E}_\rho^\pi \left[\sum_{i=t+1}^{t+m} R_i \middle| h_t\right]$, where R_i denotes the random variable distributed according to ρ for the reward at time i. The action value function is defined similarly as $V_\rho^\pi(h_t, a_{t+1}) := \mathbb{E}_\rho^\pi \left[\sum_{i=t+1}^{t+m} R_i \middle| h_t, a_{t+1}\right]$.

The agent's goal is to learn the optimal policy π^*, which is the policy that results in the value function with the maximum reward for any given history.

2.2 AIXI agent and its Monte-Carlo approximation

The following equation can be shown to be formally equivalent to Equation (1). At time t, the AIXI agent chooses action a_t^* according to

$$a_t^* = \arg\max_{a_t} \sum_{or_t} \ldots \max_{a_{t+m}} \sum_{or_{t+m}} [r_t + \ldots r_{t+m}] \sum_{\rho \in \mathcal{M}_U} 2^{-K(\rho)} \rho(or_{1:t+m}|a_{1:t+m}), \tag{2}$$

where $m \in \mathbb{N}$ is a finite lookahead horizon, $\mathcal{M}_U$ is the set of all enumerable chronological semimeasures [1], $\rho(or_{1:t+m}|a_{1:t+m})$ is the probability of observing $or_{1:t+m}$ given the action sequence $a_{1:t+m}$, and $K(\rho)$ denotes the Kolmogorov complexity [9] of ρ.

The Bayesian mixture $\xi_U := \sum_{\rho \in \mathcal{M}_U} 2^{-K(\rho)} \rho(or_{1:t}|a_{1:t})$ in (2) is a mixture environment model, with $2^{-K(\rho)}$ as the prior for ρ. A mixture environment model enjoys the property that it converges rapidly to the true environment if there exists a 'good' model in the model class.

Theorem 1. *[1] Let μ be the true environment and ξ be the mixture environment model over a model class $\mathcal{M}$. For all $n \in \mathbb{N}$ and for all $a_{1:n}$,*

$$\sum_{k=1}^{n} \sum_{or_{1:k}} \mu(or_{<k}|a_{<k}) \left(\mu(or_k|aor_{<k}a_k) - \xi(or_k|aor_{<k}a_k)\right)^2 \leq \min_{\rho \in \mathcal{M}} \left\{ \ln \frac{1}{w_0^\rho} + D_n(\mu||\rho) \right\} \tag{3}$$

where $D_n(\mu||\rho)$ is the KL divergence of $\mu(\cdot|a_{1:n})$ and $\rho(\cdot|a_{1:n})$.

To see the rapid convergence of ξ to μ, take the limit $n \to \infty$ on the l.h.s of (3) and observe that in the case where $\min_{\rho \in \mathcal{M}} \sup_n D_n(\mu||\rho)$ is bounded, the l.h.s. can only be finite if $\xi(or_k|aor_{<k}a_k)$ converges sufficiently fast to $\mu(or_k|aor_{<k}a_k)$. From Theorem 1, it can be shown that AIXI's environment mixture model ξ_U also enjoys this property with any computable environment.

The first practical approximation of AIXI is given in [4], in which the expectimax search is approximated using Monte Carlo Tree Search [6] and Bayesian mixture learning is performed using Context Tree Weighting (CTW) [5]. There is a body of work on extending the Bayesian mixture learning to larger classes of history-based models [10, 11, 12, 13]. In this paper, we generalise the Bayesian mixture learning in an orthogonal direction by adapting binarised CTW [8] to run on sequences defined by logical state abstractions, thus significantly improving the ease with which symbolic and statistical background knowledge can be incorporated into the AIXI framework.

2.3 State Abstraction and Feature Reinforcement Learning

In the general reinforcement learning setting, the state abstraction framework is concerned with finding a function $\phi : \mathcal{H} \to \mathcal{S}_\phi$ that maps every history string to an element of an abstract state space $\mathcal{S}_\phi$. Given a state abstraction ϕ, the abstract environment is a process that generates a state s' and a reward r given a history and action according to the distribution. $\rho_\phi(s', r|h, a) := \sum_{o:\phi(haor)=s'} \rho(or|ha)$. The action-observation-reward sequence of the original process thus maps to a state-action-reward sequence $sar_{1:t}$ under the mapping ϕ.

A key reason for considering abstractions is that the space $\mathcal{H}$ is too large and the underlying process may be highly irregular and difficult to learn on. The aim is thus to select a mapping ϕ such that the induced process can facilitate learning without severely compromising performance. Theoretical approaches to this question have typically focused on providing conditions that minimise the error in the action-value function between the abstract and original process [14, 15]. The ΦMDP framework instead provides an optimisation criteria for ranking candidate mappings ϕ based on how well the state-action-reward sequence generated by ϕ can be modelled as an MDP whilst still being predictive of the rewards [7]. A good ϕ results in a model that is Markovian and predictive of future rewards, facilitating the efficient learning of good actions that maximise the expected long-term reward.

Optimisation criteria for selecting mappings The ΦMDP framework provides an optimisation criteria to rank state abstractions based on the code length of the state-reward sequence they produce. For the agent to achieve high expected long-term reward, it is imperative that any mapping ϕ is able to produce a model that can predict the reward sequence well. The code length of the reward sequence captures how well the current model encodes the rewards seen so far and a mapping ϕ can be evaluated on this basis. The reward however is typically a function of the state as well as the action and so the state sequence needs to be encoded too. For more details on how MDP sequences can be coded see [7]. This gives rise to an optimisation objective.

Definition 4. Let $CL(x_{1:n})$ denote the code length of a sequence $x_{1:n}$ and $CL(x_{1:n}|a_{1:n})$ denote the code length of sequence $x_{1:n}$ conditioned on $a_{1:n}$. At time n, the $Cost_M$ objective is defined as

$$Cost_M(n, \phi) := CL(s_{1:n}|a_{1:n}) + CL(r_{1:n}|s_{1:n}, a_{1:n}) + CL(\phi) \tag{4}$$

where $\phi(h_t) = s_t$ and $h_t = aor_{1:t}$.

The objective is then to look for $\phi_n^{best} := \arg\min_\phi Cost_M(n, \phi)$. Note that $CL(\phi)$ is a term that imposes a Occam's razor-like penalty on the candidate mapping; if different candidate abstractions result in the same $Cost_M$ value, the simplest model is preferred.

The equal weighting given to the code lengths of the reward sequence and the state sequence can be suboptimal, given the state sequence is only incidentally useful in so far as it is required for predicting the reward sequence. It is not hard to construct simple environment models where $Cost_M$ gives unintuitive results. In [16], this issue is alleviated by reformulating $Cost_M$ as a convex combination of $CL(s_{1:n}|a_{1:n})$ and $CL(r_{1:n}|s_{1:n}, a_{1:n})$

$$Cost_M^\alpha(n, \phi) := \alpha CL(s_{1:n}|a_{1:n}) + (1 - \alpha)CL(r_{1:n}|s_{1:n}, a_{1:n}) + CL(\phi)$$

where $\alpha \in [0, 1]$. For our purposes, we consider the case where $\alpha = 0$.

2.4 Knowledge Representation and Reasoning Formalism

The formal agent knowledge representation and reasoning language is the one described in [17, 18]. The syntax of the language are the terms of the λ-calculus [19], extended with type polymorphism to increase its expressiveness. The inference engine has two interacting components: an equational-reasoning engine and a tableau theorem prover [20]. Thus, a predicate evaluation at the internal node of a Context Tree takes the general form $\Gamma \vdash p(h)$, where Γ is a theory consisting of equations and implications that reflects the agent's current belief base, h is the history of agent-environment interacts, and p is a predicate mapping histories to boolean values. There is also a predicate rewrite system for defining and enumerating a set of predicates.

Suitable alternatives to our logical formalism are studied in the Statistical Relational Artificial Intelligence literature [21, 22], including a few that caters specifically for relational reinforcement learning [23, 24] and Symbolic MDP/POMDPs [25, 26]. Our choice is informed by the standard arguments given in [27, 28] and the practical convenience of working with (the functional subset of) a language like Python. Our thesis that formal logic can be used to incorporate domain knowledge and broaden the model class of AIXI approximations remains unchanged with an alternate formal logic.

3 The Φ-AIXI-CTW Agent

We now present the Φ-AIXI-CTW agent. The actions and rewards are considered in their binary representation. The agent has access to a large set $\mathcal{P}$ of predicates on histories, and each subset of $\mathcal{P}$ constitutes a candidate state abstraction. In theory, a Φ-BCTW data structure constructed with the entire set $\mathcal{P}$ will converge in the limit to the distribution under the smallest subset of predicates that best models the underlying environment. However, this is computationally infeasible in practice if $\mathcal{P}$ is large. Our approach is to instead use feature selection based on the $Cost_M$ objective to first select a smaller subset of predicates for Φ-BCTW to perform a Bayesian mixture over. Finally, the agent selects actions according to the ρUCT algorithm.

3.1 Shrinking the Model Class via Feature Selection

Our method for selecting candidate state abstractions is centred around the detection of predicate features that result in low entropy conditional distributions over the rewards since lower entropy results in smaller code lengths. A simple heuristic that we employ is to look for candidate state abstractions that contains states that maximise the probability of receiving a given reward. We call such abstractions sharp mappings.

Definition 5. Let $p_\phi(r^c|s,a) = \sum_{r' \neq r \in \mathcal{R}} p_\phi(r'|s,a)$ and $p_\phi(r^c) = \sum_{r \neq r \in \mathcal{R}} p_\phi(r')$ for all $r \in \mathcal{R}$. For a $D > p_\phi(r)/p_\phi(r^c)$, an MDP mapping ϕ is *D-sharp* if for all $r \in \mathcal{R}$, there exists $s \in S_\phi$, $a \in \mathcal{A}$ such that $\frac{p_\phi(r|s,a)}{p_\phi(r^c|s,a)} \geq D$.

The reason for defining sharp mappings in this manner is that we wish to select candidate state abstractions that will be predictive of all rewards. It is often the case in reinforcement learning environments that informative reward signals are sparse during learning. Treating rewards as class labels, the problem of predicting rewards can be viewed as an unbalanced data classification task. If the unbalanced nature of the data is not taken into account, feature selection methods will tend to ignore features that may be predictive of rewards that occur infrequently but are important for performance. Thus, we adjust for this fact by ensuring that the threshold chosen for each reward is bounded from below by $\frac{p_\phi(r)}{p_\phi(r^c)}$; if a reward r occurs infrequently relative to the frequency of all other rewards, then the lower bound on D will be small. Our approach is to select predicate features that lead to sharp mappings by setting the sharpness thresholds appropriately. Increasing the sharpness threshold imposes a constraint that increasingly limits the number of satisfying state abstractions.

Feature Selection Using Decision Boundaries Using the sharpness threshold, we frame the problem as a binary classification task. For each reward $r \in R$, the sharpness of the reward distribution at a state-action pair is used to determine the class of the state by a decision rule defined by $F^r_\phi(s,a) := \mathbb{I}\left[\frac{p_\phi(r|s,a)}{p_\phi(r^c|s,a)} > D\right]$, where $\mathbb{I}$ is the indicator function. Feature selection on the binary classification task can then be performed by eliminating redundant features, defined next.

Definition 6. Let $\phi = \{p_1, \ldots, p_n\}$ be a set of predicates and let $\mathcal{S}_\phi$ be the set of length n binary strings that can be generated by ϕ. Let $F_\phi : \mathcal{S}_\phi \times \mathcal{A} \to \{0, 1\}$ be a decision rule. A predicate feature p_j is redundant if for all $x, y \in \mathcal{S}_\phi \times \mathcal{A}$ such that $x_j \neq y_j$ and $x_i = y_i$ for all $i \neq j$, we have $F_\phi(x) = F_\phi(y)$. A predicate feature is informative iff it is not redundant.

A redundant feature is a feature that does not affect the output of the decision rule, thus making no contribution to discriminating between different classes. An informative feature, in contrast, can change the output of the decision rule and shift observations across the decision boundary. To select candidate predicate features, we set up a decision rule for each reward $r \in \mathcal{R}$ and eliminate the predicate features that are redundant across all rewards.

Removing Redundant Features Using Binary Decision Diagrams Binary decision diagrams (BDD) are a family of data structures that have become an important tool and the method of choice for representing boolean functions [29], with many applications in AI [30, 31]. BDDs represent boolean functions of n variables as a directed acyclical graph. The graph consists of two types of nodes: internal nodes and terminal nodes. Every internal node is labelled by a variable and has a path to two successors that are traversed depending upon the value of the variable. Terminal nodes are labelled either 1 or 0 to denote true or false respectively.

A BDD is *reduced* if it satisfies two properties: **(a)** an internal node's left and right path must never lead to the same node and **(b)** no two nodes are allowed to be equivalent. Two nodes are equivalent if they have the same label and their subtrees are equivalent. Since the decision rules $F^r_\phi : \mathcal{S}_\phi \times \mathcal{A} \to \{0, 1\}$ we use for feature selection are Boolean functions, we can represent them as BDDs. A key property of the reduced BDD representation of F_ϕ is that only informative features remain.

Theorem 2. *Let $F^r_\phi : \mathcal{S}_\phi \times \mathcal{A} \to \{0, 1\}$ be the decision rule for reward $r \in \mathcal{R}$. Then the reduced BDD representation of F^r_ϕ contains only informative features.*

In practice there exists simple procedures to reduce a BDD. For each reward, we determine a decision rule and perform BDD reduction to get a set of informative predicates. The union of the sets of selected predicates is then chosen.

One issue with BDD reduction is that whether a feature is kept can depend on the given variable ordering. In general, BDD reduction will tend to keep more features than necessary since the variable ordering is likely to be sub-optimal. Our method for alleviating this issue is to perform multiple BDD reductions on randomly selected subspaces of the feature space, corresponding to randomly selected subsets of predicates. We name our approach Random Forest-BDD (RF-BDD). For each selected subset of predicates $\mathcal{P}_i$, let ϕ_i be the corresponding MDP mapping. We perform BDD reduction on the decision rule constructed from $\mathcal{P}_i$ given by $F^r_{\mathcal{P}_i}(s, a) := \mathbb{I}\left[\frac{p_{\phi_i}(r|s,a)}{p_{\phi_i}(r^c|s,a)} > D\right]$. Once BDD reduction has been performed for all chosen subsets, a choice has to be made on which predicates to keep. We employ a voting scheme where a predicate receives a vote if it is kept by a BDD reduction. The votes are then normalised by the number of times the predicate was selected. The predicates that have a retention rate above a certain threshold will then be kept. The kept predicates for each percept are then combined as in standard BDD selection.

The set of predicates selected by the RF-BDD procedure above, when used as state abstractions to define contexts in binarised CTW, leads to a model that approximately minimises $Cost^0_M$.

3.2 Mixture Environment Model Using Φ-Binarized Context Tree Weighting

The binarized context-tree weighting (BCTW) algorithm is a universal data compression scheme that extends the context-tree weighting algorithm [5] to non-binary alphabets. The approach is to binarize symbols to bit strings and predict each element using a separate context tree. We first define some notation to help count frequencies of substrings.

Definition 7. Let $\mathcal{A}$ be a finite alphabet and $k \in \mathbb{N}$. Define $\mathcal{A}^k$ to be all strings of length k and $\mathcal{A}^{\leq k} = \mathcal{A}^0 \cup \mathcal{A} \cup \mathcal{A}^2 \cup \ldots \cup \mathcal{A}^k$. Given a sequence $z_{1:n}$ where each $z_i \in \mathcal{A}^k$, a string $a \in \mathcal{A}^{\leq k}$ and an $m \geq |a|$, define

$$\#_m a := \{i \in \{1, \ldots, n\} \,:\, z_i = bac, b \in \mathcal{A}^{m-|a|}, c \in \mathcal{A}^{k-m}\}.$$

If $m = k$ and $|a| < m$, then $\#_m a$ counts the number of elements of $z_{1:n}$ that have a as a suffix. If $m < k$ and $|a| = m$, then $\#_m a$ counts the number of elements of $z_{1:n}$ that have a as a prefix. Finally, if $m = |a| = k$, $\#_m a$ counts the number of occurrences of a in $z_{1:n}$. Given a sequence $z_{1:n} = ((b_{11}, \ldots, b_{1k}), \ldots, (b_{n1}, \ldots, b_{nk}))$ of n binarized symbols of length k, the BCTW estimator estimates the stationary distribution of the source sequentially using k conditional CTW estimators. For the first bit, the distribution is estimated as: $\hat{P}^{(1)}(B_l = \beta; z_{1:n}) := \frac{\#_1\beta + \frac{1}{2}}{n+1}$. Let $\mathcal{T}_k$ denote the set of all context trees of depth at most k. For the l-th bit, the distribution is modelled by a depth $l-1$ CTW taking the previous $l-1$ bits as context:

$$\hat{P}^{(l)}(B_l = \beta; B_{1:l-1} = \beta_{1:l-1}, z_{1:n}) = \sum_{T \in \mathcal{T}_{l-1}} \Lambda_T^{(l)} \left(\frac{\#_l \xi_T(\beta_{1:l-1})\beta + \frac{1}{2}}{\#_{l-1}\xi_T(\beta_{1:l-1}) + 1} \right).$$

Here $\xi_T(\beta_{1:l-1})$ is the unique element of T (i.e. path down to leaf node) that is also a suffix of $\beta_{1:l-1}$ and $\Lambda_T^{(l)}$ is a normalised weight corresponding to the tree T assigned by the binary CTW estimator. (See [8] for more details on $\Lambda_T^{(l)}$.) Piecing together $\hat{P}^{(l)}$ for $l = 1, \ldots, k$ gives the overall prediction for a new symbol $z = \beta_{1:k}$ as $\hat{P}(\beta_{1:k}; z_{1:n}) = \prod_{l=1}^{k} \hat{P}^{(l)}(\beta_l; \beta_{1:l-1}, z_{1:n})$.

To model the abstract MDP environment under a set of predicates, we adapt the BCTW structure to condition on an initial context of the state and action. Let $[x]$ denote the binary representation of x and let the length of the binary representation of the state and action be d and the state and reward be k. Given the sequence $sr_{1:n}$ and current action a_n, the distribution of the first bit is given by

$$\hat{P}^{(1)}(B_1 = \beta | [s_n][a_n]; sr_{1:n}) = \sum_{T \in \mathcal{T}_d} \Lambda_T^{(d+1)} \left(\frac{\#_{d+1}\xi_T([s_n][a_n])\beta + \frac{1}{2}}{\#_d \xi_T([s_n][a_n]) + 1} \right).$$

For the l-th bit, we have

$$\hat{P}^{(l)}(B_l = \beta | [s_n][a_n]\beta_{1:l-1}; sr_{1:n}) = \sum_{T \in \mathcal{T}_{d+l-1}} \Lambda_T^{(d+l)} \left(\frac{\#_{d+l}\xi_T([s_n][a_n]\beta_{1:l-1})\beta + \frac{1}{2}}{\#_{d+l-1}\xi_T([s_n][a_n]\beta_{1:l-1}) + 1} \right).$$

Together, the overall prediction for a state-reward symbol $[sr] = \beta_{1:k}$ is given by

$$\hat{P}(\beta_{1:k} | [s_n][a_n]; sr_{1:n}) = \prod_{j=1}^{k} \hat{P}^{(j)}(\beta_j | [s_n][a_n]\beta_{1:j-1}; sr_{1:n}) \ .$$

The Φ-BCTW data structure is structurally equivalent to a Factored Action-Conditional CTW [4] (FAC-CTW) that is 'state-action' conditional instead of action-conditional. Hence, Φ-BCTW inherits the fast convergence properties that FAC-CTW possesses and will converge to the best MDP structure in its given set of predicates.

3.3 Expectimax Approximation With Modified ρUCT

We use Monte-Carlo Tree Search to perform an approximation of the finite horizon expectimax operation in AIXI. In particular, we employ the ρUCT method for selecting actions during search [4]. During expansion, a state reward pair is sampled from the environment model ρ and added as a node to the tree. Action selection is performed according to the UCB policy criteria [32]:

$$a_{UCB}(s) = \arg\max_{a \in \mathcal{A}} \begin{cases} \hat{Q}(s,a) + C\sqrt{\frac{\log(N(s))}{N(s,a)}} & \text{if } N(s,a) > 0 \\ \infty & \text{otherwise.} \end{cases} \tag{5}$$

where $C \in \mathbb{R}$ and $N(s), N(s,a)$ are the visit counts of s and s, a respectively. The parameter C controls the ratio of exploration to exploitation, and and ties are broken randomly. Suppose during the search routine of ρUCT the following sequence of states and actions $sar_{1:m}$ are encountered. An unbiased approximation of the finite horizon return updates the Q value estimate for actions nodes according to the Monte-Carlo return:

$$\hat{Q}(s_j, a_j) \leftarrow \frac{N(s_j, a_j)}{N(s_j, a_j) + 1} \hat{Q}(s_j, a_j) + \frac{1}{N(s_j, a_j) + 1} \sum_{i=j}^{m} r_i \quad , \ j = 1, \ldots, m \tag{6}$$

One can also consider computing a biased estimate via a n-step temporal difference update:

$$\hat{Q}(s_j, a_j) \leftarrow \frac{N(s_j, a_j)}{N(s_j, a_j) + 1}\hat{Q}(s_j, a_j) + \frac{1}{N(s_j, a_j) + 1}\left(\sum_{i=j}^{n-1} \gamma^i r_i + \gamma^n \hat{Q}(s_j, a_j)\right) \tag{7}$$

where $\gamma \in [0, 1)$ is a discount factor that geometrically decreases as the horizon increases. In theory, discounting transforms the Q estimate to estimating the infinite horizon, discounted return. However in practice, geometric discounting results in a finite effective horizon which can be treated as the search horizon m in AIXI's expectimax operation.

Using Equation 7 requires a few modifications to ρUCT and offers a few advantages. In the traditional setting, ρUCT generates estimates for nodes corresponding to history strings. In general, history sequences will not repeat and thus the Q value estimate must be computed from scratch at each time step. This requires multiple searches through the tree to generate return samples and the number of samples is typically large for nontrivial problems. By performing feature selection and modelling the environment as a Φ-BCTW, the environment model reduces to an MDP with a 'small' state space. We choose to store the search nodes of the tree and maintain the value estimates for the duration of the agent's life, thus reducing the number of samples that need to be generated per-step.

4 Experiments

In this section we evaluate Φ-AIXI-CTW on the novel task of learning to control an epidemic process over large-scale contact networks, a highly structured environment that exhibits an exponential state space, partial observability and history dependency. We first introduce the epidemiological model before discussing the setup and performance of our agent.

4.1 Epidemic Processes on Contact Networks

In this paper epidemic processes are modelled on contact networks [33, 34, 35]. A contact network is an undirected graph where the nodes represent individuals and the edges represent interactions between individuals. Each individual node in the graph is labelled by one of four states corresponding to their infection status: Susceptible (S), Exposed (E), Infectious (I), and Recovered (R). Every node also maintains an immunity level ω. The model evolves as a POMDP. At time t, the temporal graph is given by $G_t = (V, \mathcal{E}_t)$ with V as the set of nodes and $\mathcal{E}_t$ as the set of edges at time t. A function $\zeta_t : V \to \{\mathsf{S}, \mathsf{E}, \mathsf{I}, \mathsf{R}\} \times \{1, \eta_1, \eta_2\}$ maps each node to its label and one of three immunity levels where $\eta_1, \eta_2 \in \mathbb{R}_+$. Together, (G_t, ζ_t) constitute the state. At time t, a Susceptible node with k_t Infectious neighbours becomes Exposed with probability $\frac{1-(1-\beta)^{k_t}}{\omega}$, where $\beta \in [0, 1]$ is the rate of transmission. An Exposed node becomes Infectious at rate σ. Similarly, an Infectious node becomes Recovered at a rate γ and becomes Susceptible again at a rate ρ.

At every time step, the process emits an observation on each node from $\{+, -, ?\}$ corresponding to whether a node tests positive, negative or is unknown/untested. The observation model is parametrised in such a way that positive tests are more likely if the underlying node is Infectious, corresponding to the realistic scenario where sick individuals are more likely to test and also test positive.

The agent can perform an action at each time step. We consider a set of 11 possible actions $\{DoNothing, Vaccinate(i, j), Quarantine(i)\}$, where $i \in [0, 0.2, 0.4, 0.6, 0.8, 1.0]$ and $j = i + 0.2$. A $Vaccinate(i, j)$ action increases the immunity level of the top ith to jth percent of nodes as ranked by betweenness centrality by one level, up to the maximum value. Note that a conferred immunity level lasts for the entire length of an episode unless increased. A $Quarantine(i)$ action quarantines the top ith percent of nodes by removing all edges incident on those nodes for one time step. The reward function is parametrised to evaluate the agent's ability to balance the cost of its actions vs the cost of the epidemic. At each time step, the instantaneous reward is given by

$$r_t(o_t, a_{t-1}) \coloneqq -\lambda Positives(o_t) - Action_Cost(a_{t-1})$$

where $\lambda \in R_+$, $Positives(o_t)$ counts the number of positive tests in the observation o_t and $Action_Cost(a_{t-1})$ is a function determining the cost of each action. Thus changing λ changes the ratio of contributions to the reward received at each time step. If the agent successfully terminates the epidemic, i.e. there are no more Exposed or Infectious nodes, the agent receives a positive reward of

Experiment	n-step horizon	Discount factor (γ)	Explore bias (C)	ρUCT Simulations	ϵ	α	Number of predicates
1	4	0.99	1000	10	0.9999	0.999995	36
2	4	0.99	1000	10	0.9999	0.999995	32

Table 1: Φ-AIXI-CTW model learning configuration

2 per node. During a run, it is possible the environment never reaches an absorbing state. If the agent has not successfully stopped the epidemic process after 1000 steps, we terminate the episode.

Epidemic control is a topical subject given the prevalance of COVID-19 and recent approaches to this question vary wildly in terms of the how the problem is modelled. Most approaches vary in the different ways they model the dynamics of an epidemic and the action space considered is usually fairly small [36, 37, 38, 39, 40, 41, 42]. Ultimately, the differences in the modelling of epidemic processes makes comparison between results difficult. As such, comparisons to heuristic baselines are common [43, 42]. With no consensus on the appropriate model to use, our model is chosen as it is sufficiently complex to demonstrate the efficacy and behaviour of our agent.

4.2 Experimental Setup

We analyse the agent's performance and behaviour on epidemic processes under four different settings. All experiments were performed on a 12-Core AMD Ryzen Threadripper 1920x processor and 32 gigabytes of memory. We use an email network dataset, licensed under a Creative Commons Attribution-ShareAlike License, containing 1133 nodes and 5451 edges as the underlying contact network in all four cases [44, 45]. The dynamics and observation models are parametrised in the same way across all 4 experiments (see Appendix). The $Action_Cost(a_t)$ is also defined the same way across the four settings. A $Quarantine(i)$ action imparts a cost of 1 per node that is quarantined at the given time step. A $Vaccinate(i, j)$ action imparts a lower cost of 0.5 per node. We generate a set of 1489 predicate functions to serve as the initial set of candidate state abstractions. The set generated consists of predicates on top of functions such as the percentage each action was selected in the last N steps, the observed infection rate on different subsets of nodes, and also the infection rate on different subset of nodes as computed by particle filtering. A full description of how the predicates were generated is given in the Appendix.

The Φ-AIXI-CTW agent is trained in an online fashion. The agent explores with probability $\epsilon\alpha^t$ at each step t until $\epsilon\alpha^t < 0.03$, where the agent performs in an ϵ-greedy way with exploration rate 0.03. RF-BDD was performed with a threshold value of 0.9 across all rewards. We describe the set of predicates chosen in each experiment by RF-BDD in the Appendix. Table 1 describes the agent configurations in our experiments. In general, RF-BDD tended to select predicates representing bits in the binary representation of the observed infection rate and the rate of change of the observed infection rate on different subsets of the betweenness centrality nodes. A random selection of different action sequence indicator functions were also kept. Note that predicates on the infection rate as computed by particle filtering were not selected and this can be attributed to the particle degeneracy issue that occurs in high dimensional problems [46]. These choices by RF-BDD highlight its ability to select a small set of predicates that provided useful information for epidemic control. The inclusion of a random selection of different action sequence indicator functions also demonstrates that it is not perfect. A more thorough investigation is the topic of future work.

4.3 Results

We evaluate the performance of the agent online. For each experiment, we plot four graphs. The first graph plots the learning curve computed as a moving average over 5000 steps of the reward per cycle. The remaining graphs provide insight into the learnt policy. The second graph plots the percentage each type of action is performed over a moving average of 500 steps as well as a moving average of the infection rate over the same window. The remaining two graphs provide an example of the agent's behaviour and its impact on the infection rate over an episode collected toward the end of training.

Figure 1 depicts the agents behaviour over two experiments with differing vaccination efficacy and an infection rate weighting of $\lambda = 10$. In both cases, the agent learns to be more aggressive in terminating the epidemic. In Experiment 3 the agent's behaviour is to quarantine almost always, and occasionally vaccinate. This is reflected in the plots of the agent's episode behaviour. Quarantine actions are performed the entire time except once, resulting in a fast termination of the epidemic

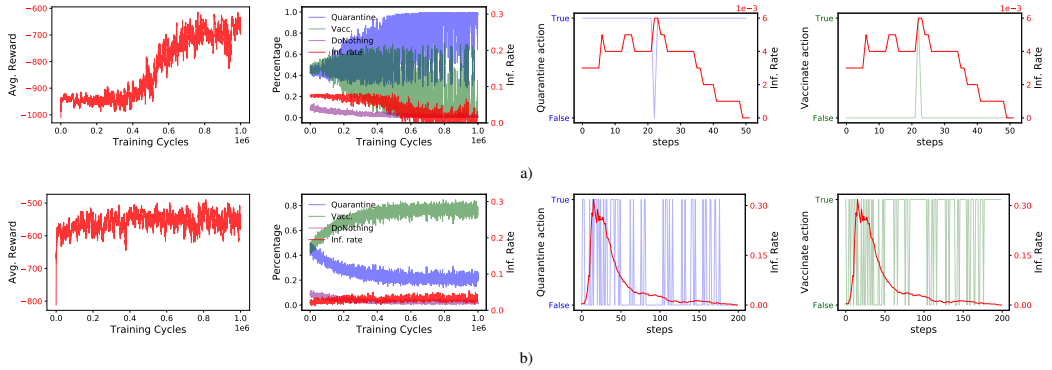

Figure 1: a) Experiment 1, $\lambda = 10, \eta_1 = 2, \eta_2 = 4$. b) Experiment 4=2, $\lambda = 10, \eta_1 = 10, \eta_2 = 20$.

process. In contrast, Experiment 4 shows that the agent prefers to vaccinate more often. This preference can be attributed to the added effectiveness that a stronger vaccination action imparts to lowering the infection rate. When analysing the agent's behaviour over an episode, it is clear that the agent chooses to vaccinate almost always whilst only quarantining at timely moments. This results in the agent also learning to terminate the episode early, albeit not as quickly. In comparison to Experiment 3, the average reward per cycle is also higher at the end of learning.

Overall, the results demonstrate that the agent is able to learn plausible behaviours under the different environment settings. This demonstrates the potential of our approach in reducing a complex environment to a simpler problem whilst still maintaining performance. Nevertheless, it is also quite clear that the agent did not learn optimal behaviour except potentially in the case of Experiment 3. In Experiment 4, the agent did not learn that vaccinating a node more than twice does not further reduce the cost received. However, this is not surprising as this is a difficult causal relationship to learn from data alone. Furthermore, the full set of generated predicates do not contain enough information to isolate this particular history dependency. In fact, the predicates selected generally capture different statistics about the infection rate at the current time step. If the value estimate of vaccinating in a highly visited state is high due to it lowering future received costs, the agent will learn to perform the vaccination action every time the state is encountered, even if it has done so many times before. We expect that importing better domain knowledge into the original set of predicates would solve this issue for the Φ-AIXI-CTW agent.

5 Conclusion

We have introduced the Φ-AIXI-CTW agent model, an agent that expands the model class that AIXI may be approximated over to structured and non-Markovian domains via logical state abstractions. We show that our approach is sufficiently powerful to handle control of epidemic processes on contact networks, a class of problems with a highly structured exponential state space, partial observability and history dependency. In doing so, our result provides another strong evidence that AIXI is not just a mathematically interesting theory, but that it can eventually lead to practically useful agents.

Whilst RF-BDD has demonstrated its usefulness empirically in our experiments, a better understanding of how it works theoretically and how it behaves in other contexts is important. In [43] the authors identify the epidemic control problem within a class of problems they call the control of diffusion processes over networks through nodal interventions. Our experimental model constitutes an instance of this class of problems and it would be interesting to extend our agent setup to other problems within this domain.

A key concern for the real-world application of reinforcement learning on contact networks for control is the preservation of privacy for individual nodes. Our agent setup does not restrict the usage of any form of domain knowledge on the complete graph such as an individual node's contact network and extended neighbourhood. Investigating the performance of our agent under privacy preserving constraints such as differential privacy [47] is an interesting topic for future research.

References

[1] Marcus Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2005.

[2] Ray J. Solomonoff. The discovery of algorithmic probability. *J. Comput. Syst. Sci.*, 55(1):73–88, 1997.

[3] Marcus Hutter. One decade of universal artificial intelligence. *CoRR*, abs/1202.6153, 2012.

[4] Joel Veness, Kee Siong Ng, Marcus Hutter, William T. B. Uther, and David Silver. A monte-carlo AIXI approximation. *J. Artif. Intell. Res.*, 40:95–142, 2011.

[5] F.M.J. Willems, Y.M. Shtarkov, and T.J. Tjalkens. The context-tree weighting method: basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

[6] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *ECML 2006*, pages 282–293. Springer Berlin Heidelberg, 2006.

[7] Marcus Hutter. Feature reinforcement learning: Part I. Unstructured MDPs. In *J. Artif. Gen. Intell.*, 2009.

[8] Badri N. Vellambi and Marcus Hutter. Convergence of binarized context-tree weighting for estimating distributions of stationary sources. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 731–735, 2018.

[9] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 1997.

[10] Joel Veness, Kee Siong Ng, Marcus Hutter, and Michael H. Bowling. Context tree switching. In James A. Storer and Michael W. Marcellin, editors, *2012 Data Compression Conference*, pages 327–336. IEEE Computer Society, 2012.

[11] Joel Veness, Martha White, Michael Bowling, and András György. Partition tree weighting. In Ali Bilgin, Michael W. Marcellin, Joan Serra-Sagristà, and James A. Storer, editors, *2013 Data Compression Conference*, pages 321–330. IEEE, 2013.

[12] Marc G. Bellemare, Joel Veness, and Michael Bowling. Bayesian learning of recursively factored environments. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1211–1219. JMLR.org, 2013.

[13] Marc G. Bellemare, Joel Veness, and Erik Talvitie. Skip context tree switching. In *Proceedings of the 31th International Conference on Machine Learning*, pages 1458–1466. JMLR.org, 2014.

[14] David Abel, Dilip Arumugam, Lucas Lehnert, and Michael L. Littman. State abstractions for lifelong reinforcement learning. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, pages 10–19. PMLR, 2018.

[15] Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics*, 2006.

[16] Phuong Minh Nguyen, Peter Sunehag, and Marcus Hutter. Feature reinforcement learning in practice. In Scott Sanner and Marcus Hutter, editors, *Recent Advances in Reinforcement Learning - 9th European Workshop, EWRL 2011, Revised Selected Papers*, volume 7188 of *Lecture Notes in Computer Science*, pages 66–77. Springer, 2011.

[17] John W. Lloyd and Kee Siong Ng. Declarative programming for agent applications. *Autonomous Agents Multi Agent Systems*, 23(2):224–272, 2011.

[18] John W. Lloyd. *Logic for Learning: Learning Comprehensible Theories from Structured Data*. Springer, 2003.

[19] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[20] Luis Fariñas del Cerro and Olivier Gasquet. A general framework for pattern-driven modal tableaux. *Logic Journal of the IGPL*, 10(1):51–83, 2002.

[21] L. Getoor and B. Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[22] Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2016.

[23] Saso Dzeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. *Mach. Learn.*, 43(1/2):7–52, 2001.

[24] Kurt Driessens. Relational reinforcement learning. In Claude Sammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning and Data Mining*, pages 1096–1103. Springer, 2017.

[25] Kristian Kersting, Martijn van Otterlo, and Luc De Raedt. Bellman goes relational. In Carla E. Brodley, editor, *Proceedings of the Twenty-first International Conference on Machine Learning*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

[26] Scott Sanner and Kristian Kersting. Symbolic dynamic programming for first-order POMDPs. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*. AAAI Press, 2010.

[27] W.M. Farmer. The seven virtues of simple type theory. *Journal of Applied Logic*, 6(3):267–286, 2008.

[28] John Hughes. Why functional programming matters. *The Computer Journal*, 32(2):98–107, 1989.

[29] Donald E. Knuth. *The Art of Computer Programming, Pre-Fascicle 1B*, volume 4. Addison-Wesley, 2008.

[30] Jesse Hoey, Robert St-Aubin, Alan J. Hu, and Craig Boutilier. SPUDD: stochastic planning using decision diagrams. *CoRR*, abs/1301.6704, 2013.

[31] Jinbo Huang and Adnan Darwiche. The language of search. *J. Artif. Intell. Res.*, 29:191–219, 2007.

[32] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3:397–422, 2003.

[33] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. Epidemic processes in complex networks. *Reviews of Modern Physics*, 87(3), 2015.

[34] Cameron Nowzari, Victor M Preciado, and George J Pappas. Analysis and control of epidemics: A survey of spreading processes on complex networks. *IEEE Control Systems Magazine*, 36(1):26–46, 2016.

[35] Mark Newman. *Networks*. Oxford University Press, 2018.

[36] Mauricio Arango and Lyudmil Pelov. COVID-19 pandemic cyclic lockdown optimization using reinforcement learning. *CoRR*, abs/2009.04647, 2020.

[37] Arthur Charpentier, Romuald Elie, Mathieu Laurière, and Viet Chi Tran. Covid-19 pandemic control: balancing detection policy and lockdown intervention under ICU sustainability, 2020.

[38] Cédric Colas, Boris Hejblum, Sébastien Rouillon, Rodolphe Thiébaut, Pierre-Yves Oudeyer, Clément Moulin-Frier, and Mélanie Prague. Epidemioptim: A toolbox for the optimization of control policies in epidemiological models, 2020.

[39] Fred Brauer and Carlos Castillo-Chávez. *Mathematical Models in Population Biology and Epidemiology*. Springer, 2012.

[40] Roy M Anderson. *The population dynamics of infectious diseases: theory and applications*. Springer, 2013.

[41] Konstantin Berestizshevsky, Koffi-Emmanuel Sadzi, Guy Even, and Moni Shahar. Optimization of resource-constrained policies for covid-19 testing and quarantining. *Journal of Communications and Networks*, 23(5):326–339, 2021.

[42] Varun Kompella, Roberto Capobianco, Stacy Jong, Jonathan Browne, Spencer Fox, Lauren Meyers, Peter Wurman, and Peter Stone. Reinforcement learning for optimization of covid-19 mitigation policies, 2020.

[43] Eli A. Meirom, Haggai Maron, Shie Mannor, and Gal Chechik. Controlling graph dynamics with reinforcement learning and graph neural networks, 2020.

[44] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015.

[45] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68:065103, Dec 2003.

[46] Chris Snyder, Thomas Bengtsson, and Mathias Morzfeld. Performance bounds for particle filters using the optimal proposal. *Monthly Weather Review*, 143(11):4750 – 4761, 2015.

[47] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See sections 3 for our main claims and section 4 for our empirical results.
   (b) Did you describe the limitations of your work? [Yes] See the conclusion section.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the conclusion section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [No] . The results included have been proved in other papers. These papers are referenced to.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See table 1. Some more will be reported in the Appendix due to space constraints.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] . The results presented deal with the analysis of the agent's behaviour in a single run/episode. This type of result is not conducive to averaging over multiple runs.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] .
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]