Towards Cost Sensitive Decision Making

Yang Li
UNC-Chapel Hill

Junier B. Oliva
UNC-Chapel Hill

Abstract

Many real-world situations allow for the acquisition of additional relevant information when making decisions with limited or uncertain data. However, traditional RL approaches either require all features to be acquired beforehand (e.g. in a MDP) or regard part of them as missing data that cannot be acquired (e.g. in a POMDP). In this work, we consider RL models that may actively acquire features from the environment to improve the decision quality and certainty, while automatically balancing the cost of feature acquisition process and the reward of task decision process. We propose the Active-Acquisition POMDP and identify two types of the acquisition process for different application domains. In order to assist the agent in the actively-acquired partially-observed environment and alleviate the exploration-exploitation dilemma, we develop a model-based approach, where a deep generative model is utilized to capture the dependencies of the features and impute the unobserved features. The imputations essentially represent the beliefs of the agent. Equipped with the dynamics model, we develop hierarchical RL algorithms to resolve both types of the AA-POMDPs. Empirical results demonstrate that our approach achieves considerably better performance than existing POMDP-RL solutions.

1 Introduction

Recently, machine learning models for sequential decision making have made significant progress due to the development of reinforcement learning (RL). These models have achieved remarkable success in many application domains, such as games (Mnih et al., 2015; Silver et al., 2016, 2017), robotics control (Finn et al., 2016; Levine et al., 2016; Polydoros and Nalpantidis, 2017; Haarnoja et al., 2018; Niroui et al., 2019) and medical diagnosis (Ling et al., 2017; Peng et al., 2018; Coronato et al., 2020; Yu et al., 2021). However, the current RL paradigm is incongruous with the expectation of many real-world decision-making systems. First, for fully-observed Markov decision processes (MDPs), the features at each decision step are assumed to be fully observed. In situations like medical diagnosis, some features, such as MRI, might be expensive to obtain; some features might even pose a risk to the patient, such as X-Ray. Furthermore, acquiring all features at each step may create redundancy, as some features will not change within the adjacent time frames. Therefore, the intelligent decision-making systems are expected to automatically balance the cost of feature acquisition and the improvement of decision by acquiring only the necessary information. Second, for partially-observed Markov decision processes (POMDPs), the observation at each step is determined by an unknown observation model of the environment, thus no additional information (features) may be obtained to improve the decision.

In stark contrast to the current RL paradigm, human agents routinely reason over instances with incomplete features and decide when and what additional information to obtain. For example, consider clinicians in intensive care units (ICUs), which have to make sequences of treatment decisions for patients at risk. Typically, all of the (dynamic) patient information is not known, however, and while knowledge of the patient is critical when deciding what treatment decisions to make, due to time/cost/risk constraints the clinician must carefully decide what additional patient attributes (e.g. stemming from a blood sample, or biopsy, etc.) are most worth their cost for better downstream treatment decisions. In order to more closely match the needs of many real-world applications, we propose a active acquisition partially observed Markov decision process (AA-POMDP) and develop several

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

novel RL techniques to solve it. Our agent not only makes decisions with incomplete/missing features, but also dynamically determines the most valuable subset of features to obtain at each decision step.

In this work, we identify two types of AA-POMDPs based on how features may be acquired. First, **Sequential AA-POMDP**, where features are acquired sequentially before a task action is conducted. Here, the later acquisitions will depend on the values of previously acquired features. This type of AA-POMDP is applicable when the feature acquisition actions do not modify the underlying state (such as non-invasive test in medical scenario) and the feature acquisition process takes negligible time compared to the task state changing. Second, **Batch AA-POMDP**, where features are acquired simultaneously in a batch. This type of POMDP is suitable for situations where the feature acquisition actions can modify the state or the state changes so quickly relative to acquisition that there is not enough time for a sequential acquisition.

The agent can only observe part of the underlying features when making a decision for the task. Therefore, our model is essentially partially observed and inherits all the difficulties for solving POMDP (Monahan, 1982; Spaan, 2012). Meanwhile, in contrast to typical POMDP in RL literature, here the observation is controlled by the agent itself, which introduces additional challenges. First, the action space for feature acquisition is exponential to the number of candidate features. That is, for a d-dimensional feature space, there are 2^d possible acquisition actions in total. The large action space makes it difficult for RL agents to explore efficiently. Second, the feature acquisition process and the task decision process are intimately correlated. The feature acquisition process must collect informative features so that appropriate decisions can be made for the task. Moreover, the task decisions should transit the underlying MDP into appropriate states so that acquisitions can be performed effectively.

In order to deal with the aforementioned challenges, we propose a model-based approach in which a generative model is utilized to capture the dependencies between features (see § 3.1). Given a sequence of acquired features and corresponding task actions, the generative model predicts the possible state at the next decision step by imputing the missing features, which represents the beliefs of our agent. The acquisition action and the task action are both determined based on the belief states, which help the agent learn better policy with partial observation. Furthermore, the generative model can assist the agent with an intrinsic reward by assessing the imputation quality of the underlying state, which essentially provides guidance to the acquisition process. In addition, we decompose the acquisition process and the task decision process into a hierarchy, where the high-level task policy takes inputs from the low-level acquisition policy and provides reward signal in return based on its policy uncertainty and value estimates (see § 3.3 for details).

Our contributions are as follows: 1) We propose the active acquisition partially observed Markov decision process (AA-POMDP), which integrates the feature acquisition process with the task decision process to make decisions taking into account the cost of feature acquisition. 2) We identify two types of the AA-POMDP, Sequential AA-POMDP and Batch AA-POMDP, to accommodate different application requirements. 3) We develop a novel generative model for partially observed sequences that captures the dependencies across features and across time steps. The generative model serves as a surrogate of the task state transition model and assists the agent by estimating the beliefs. We then develop model-based RL agents for both types of the AA-POMDP. 4) We formulate the feature acquisition process and the task decision process into a hierarchical structure and propose hierarchical RL approaches that automatically balance the feature acquisition cost and task reward. 5) We demonstrate the effectiveness of our approach on several benchmark environments and achieve state-of-the-art performance compared to baselines. Our code is released at `https://github.com/leao1995/CostRL`.

2 Active Acquisition POMDP

A discounted AA-POMDP is an environment defined by a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{R}, \mathcal{C}, \gamma \rangle$, where $\mathcal{S}$ is the state space and $\mathcal{A} = \mathcal{A}_f \cup \mathcal{A}_c$ is the joint action space of feature acquisition actions $\mathcal{A}_f$ and task control actions $\mathcal{A}_c$. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ represents the state transition kernel, which can be deterministic or stochastic. As in ordinary POMDP, the observation space $\mathcal{O}$ is related to $\mathcal{S}$ by the observation/emission model $p(o \mid s', a)$, which defines the probability of observing o when the agent takes action a resulting in a state s'. In AA-POMDP, however, the observation model is specified as $p(o \mid s', a_f)$. I.e., the observation is controlled only by the feature acquisition action a_f. For a state s' with underlying d-dimensional measurable features x' that are unknown beforehand, the feature acquisition action a_f will result in acquiring a subset of features $x'_v, v \subseteq \{1, \dots, d\}$, where v is decoded from action a_f. The features $x'_u, u = \{1, \dots, d\} \setminus v$ remain unobserved to the agent. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A}_c \to \mathbb{R}$ specifies the reward structure for the original task MDP; the cost function $\mathcal{C} : \mathcal{S} \times \mathcal{A}_f \to \mathbb{R}_{\geq 0}$ defines the cost of acquisitions. $\gamma \in [0, 1]$ is the discount factor.

Given the partial observation history and the action

history, we let b represent a belief distribution over possible states. When the agent takes an action a based on its policy $\pi(a \mid b)$, it receives the immediate reward

$$r = \mathcal{R}(s,a)\mathbb{I}(a \in \mathcal{A}_c) - \mathcal{C}(s,a)\mathbb{I}(a \in \mathcal{A}_f). \quad (1)$$

Our goal is to learn a policy $\pi(a \mid b)$ that maximizes the expected cumulative discounted reward

$$\mathbb{E}_{\pi,\mathcal{T}}\left[\sum_{h=1}^{H} \gamma^h r\right], \quad (2)$$

where H represents the horizon of an episode. Next, we describe two types of the AA-POMDP that acquires features in different manners.

Batch AA-POMDP In the above AA-POMDP formulation, the action space consists of feature acquisition actions $a_f \in \mathcal{A}_f$ and task control actions $a_c \in \mathcal{A}_c$. The policy $\pi(a \mid b)$ can be decomposed into two sub-policies: π_f, which controls what acquisition actions to perform, and π_c, which controls what task-level actions to perform,

$$\pi(a \mid b) = \pi_f(a_f \mid b)\pi_c(a_c \mid b'), \quad (3)$$

where the belief b' is updated after acquiring the features indicated by the acquisition action a_f. This formulation implies that the features are acquired simultaneously in a batch. For the d-dimensional feature space, the acquisition action space $\mathcal{A}_f = 2^{[d]}$ is the powerset of all features, where $[d]$ represents the set $\{1,\ldots,d\}$. Each action $a_f \in \mathcal{A}_f$ indicates the subset of features being acquired. That is, the acquired features are $x_v, v = a_f \subseteq \{1,\ldots,d\}$.

The batch acquisition paradigm is useful when features are so time-critical that all acquisitions need to be performed in parallel to save time. For example, in an emergency, the doctor might need to acquire certain features as soon as possible to decide on a first aid strategy. Another situation where the batch acquisition may help is when the acquisition action can modify the underlying state or the state may change between two acquisitions, such as invasive procedures.

Sequential AA-POMDP In addition to batch acquisition, we also introduce the sequential acquisition scheme, where the features are acquired one-by-one. Each acquisition action will acquire one of the features $\{1,\ldots,d\}$. We also introduce a special acquisition action ϕ to indicate the termination of the acquisition process. Therefore, the acquisition action space becomes $\mathcal{A}_f = \{1,\ldots,d\} \cup \{\phi\}$. Given the sequential acquisition process, we can further decompose (3) to

$$\pi(a \mid b) = \textstyle\prod_{k=1}^{K} \pi_f(a_f^{(k)} \mid b'_{k-1})\pi_c(a_c \mid b'_K), \quad (4)$$

where b'_k is the belief after k acquisition steps ($b'_0 = b$), K is the number of acquired features, and the last acquisition action is always ϕ. Note that the beliefs b'_k are updated based on all the previously acquired features; the updated belief represents a more accurate distribution of the underlying state, and thus enabling better acquisition plan.

The sequential acquisition scheme may further reduce redundancy due to the awareness of the values from previous acquisitions, but at the expense of increased acquisition time due to its sequential nature. Therefore, this formulation is only applicable when the acquisition action is fast relative to the state changing. Another implication of this formulation is that the acquisition action will not change the underlying state, otherwise, the previous observations will be outdated when making the task decisions.

3 Methods

In this section, we introduce a generative model to capture the features dependencies along the state transition trajectory. The sequential generative model is leveraged afterwards to impute the missing features, which represent the agent's belief. We then construct the feature acquisition policy and the task control policy in a hierarchical way based on the belief estimation.

3.1 Partially Observed Sequence Modeling

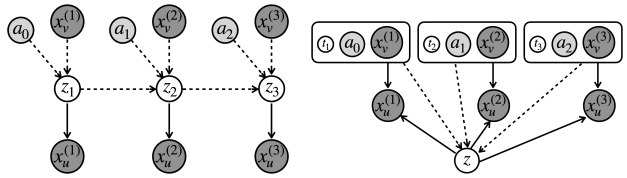
Let $t \in \{1,\ldots,T\}$ denote a time step where state transition happens due to the execution of task action $a_c^{(t)}$ in the environment. At each time step t, the agent observes a subset of features $x_v^{(t)}$ from state $s^{(t)}$, while the features $x_u^{(t)}$ remain unobserved. In order to model the state transitions and estimate the beliefs about the underlying state, we build a generative model to impute the unobserved features conditioned on the observed ones and the action sequence:

$$p(x_u^{(1:T)} \mid x_v^{(1:T)}, a_c^{(0:T-1)}). \quad (5)$$

To simplify notation, we denote $a_c^{(0)}$ as a dummy action that initializes the environment. Note that the conditionals could be evaluated on an arbitrary subset of features since the agent may acquire different features for different instances. The conditionals essentially capture dependencies between the subset of features and across time steps.

One way to model the conditional in (5) is to exploit its sequential nature and factorize it by

$$\begin{aligned} &p(x_u^{(1:T)} \mid x_v^{(1:T)}, a_c^{(0:T-1)}) \\ &= \prod_{t=1}^{T} p(x_u^{(t)} \mid x_u^{(1:t-1)}, x_v^{(1:t)}, a_c^{(0:t-1)}). \end{aligned} \quad (6)$$



(a) Sequential Formulation (b) Set Formulation

Figure 1: Graphical model of a 3-step trajectory, with dashed arrows for inference and solid arrows for generation.

A sequential latent variable $z^{(t)}$ can be introduced to simplify the model as in many sequential VAEs (Chung et al., 2015; Igl et al., 2018; Zhu et al., 2020; Yin et al., 2020). Please see Fig. 1(a) for an illustration. However, the sequential formulation has several drawbacks: First, the latent variable is updated sequentially, which means the latent only depends on previous time steps, therefore the training signals coming from later time steps cannot be leveraged. Second, due to the limitation of recurrent models, the previous time steps might not have significant influence at the current time step, especially when the episode is long. Third, in order to make a prediction at a distant time step, the model has to unroll the latent multiple times, which could make the error accumulated and result in erroneous predictions.

In order to overcome those drawbacks, we draw inspiration from set modeling (Bender et al., 2020; Li et al., 2020b; Li and Oliva, 2021b; Kim et al., 2021; Biloš and Günnemann, 2021) and Transformer (Shan et al., 2021; Fang et al., 2021; Petrovich et al., 2021) literature and formulate our generative modeling task in (5) as a conditional set generation problem. Specifically, we concatenate the time index with the corresponding features and actions as a tuple and then the sequence becomes a permutation invariant set $\{(t, x_v^{(t)}, x_u^{(t)}, a^{(t-1)})\}_{t=1}^T$. We can reformulate (5) as

$$p(\{x_u^{(t)}\}_{t=1}^T \mid \{(t, x_v^{(t)}, a_c^{(t-1)})\}_{t=1}^T) \equiv p(\mathbf{x}_u \mid \mathbf{ax}_v), \quad (7)$$

where we denote $\mathbf{x}_u := \{x_u^{(t)}\}_{t=1}^T$ and $\mathbf{ax}_v := \{(t, x_v^{(t)}, a_c^{(t-1)})\}_{t=1}^T$ for notation simplicity. Our Partially Observed Set models for Sequences (POSS) precisely overcomes the shortcomings of the aforementioned sequential generative models. Based on the set formulation, we can now draw samples at arbitrary time points without having to rolling out the sequence step-by-step. During training, later time steps can propagate gradients to early ones and even distant time points can influence each other. Please see Fig. 1(b) for an illustration.

To learn the conditional distribution over sets, we employ De Finetti's theorem (Diaconis and Freedman, 1980; Kerns and Székely, 2006; Edwards and Storkey, 2017; Li and Oliva, 2021b) and introduce a latent variable z. Given the latent variable, the conditionals can be decomposed:

$$p(\mathbf{x}_u \mid \mathbf{ax}_v) = \int \prod_{t=1}^T p(x_u^{(t)} \mid z, t, x_v^{(t)}, a_c^{(t-1)}) p(z \mid \mathbf{ax}_v) dz. \quad (8)$$

However, optimizing (8) is still intractable due to the high dimensional integration over z. Therefore, we propose to utilize a variational approximation and optimize a lower bound:

$$\log p(\mathbf{x}_u \mid \mathbf{ax}_v) \geq \sum_{t=1}^T \mathbb{E}_{q(z|\mathbf{ax})} \log p(x_u^{(t)} \mid z, t, x_v^{(t)}, a_c^{(t-1)}) - D_{\mathrm{KL}}(q(z \mid \mathbf{ax}) \| p(z \mid \mathbf{ax}_v)), \quad (9)$$

where $\mathbf{ax}$ denotes the set $\{(t, x_u^{(t)}, x_v^{(t)}, a_c^{(t-1)})\}_{t=1}^T$, which includes all of the features in $x^{(t)}$. $q(z \mid \mathbf{ax})$ and $p(z \mid \mathbf{ax}_v)$ are variational posterior and prior respectively, and they are permutation invariant w.r.t. the conditioning set inputs. $p(x_u^{(t)} \mid z, t, x_v^{(t)}, a_c^{(t-1)})$ is the decoder distribution, which could be shared for each time step t. Note that different from typical VAE models, the decoder operates over arbitrary subset of features. That is, the decoder takes in a subset of observed features along with other conditional information and outputs the distribution for the remaining subset of unobserved features. Please see Appendix A for detailed derivations and model illustration.

To deal with the arbitrary dimensionality for feature subsets $x_u^{(t)}$ and $x_v^{(t)}$, we impute the missing features with zeros and introduce a binary mask to indicate whether the corresponding dimensions are observed or not. That is, for the prior and the decoder, $x_v^{(t)}$ is represented as the concatenation of a d-dimensional features and a d-dimensional binary mask, where the missing features are replaced by zeros; while for the posterior, we combine $x_u^{(t)}$ and $x_v^{(t)}$ and regard all features as observed.

Given the complexity of set based inputs and arbitrary dimensionality of observed features, modeling the posterior and prior using a simple distribution family, such as Gaussian, may not be optimal. Therefore, we propose to use normalizing flows for both prior and posterior distributions. Following the best practice in normalizing flow literature (Kingma et al., 2016; Durkan et al., 2019a), we model the posterior using inverse autoregressive flow for its fast sampling speed. The prior is modeled using a coupling flow with spline networks. The base distribution for both distributions are Gaussian conditioned on their corresponding set representations. However, the ELBO in (9) is now not analytically available due to normalizing flow based posterior and prior distributions. We instead using

a Monte Carlo estimation by sampling multiple (M) latent z_m from the posterior:

$$\frac{1}{M}\sum_{t=1}^{T}\sum_{m=1}^{M}\Big[\log p(x_u^{(t)} \mid z_m, t, x_v^{(t)}, a_c^{(t-1)}) - \log q(z_m \mid \mathbf{ax}) + \log p(z_m \mid \mathbf{ax}_v)\Big]. \tag{10}$$

During training, we assume access to both the observed features $x_v^{(t)}$ and the unobserved features $x_u^{(t)}$. Therefore, we can directly optimize the ELBO in (10). During sampling, given a set of observed features $\{x_v^{(t)}\}_{t=1}^T$ and the corresponding actions $\{a_c^{(t-1)}\}_{t=1}^T$, we can impute the unobserved features at any time steps, even at time steps beyond T.

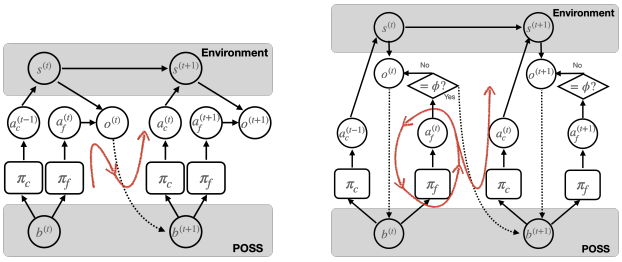
3.2 Belief State Estimation

In order to solve the aforementioned AA-POMDP, our agent will need to determine an optimal acquisition plan and an optimal task action sequence based solely on the partially observed information. Fortunately, the sequential generative model can impute the missing features and thus estimate the belief about the underlying state.

At any specific time step h, suppose the agent has executed task actions $a_c^{(<h)} \equiv \{a_c^{(i-1)}\}_{i=1}^h$ resulting in the underlying state $s^{(h)}$, the agent has access to the observation history $o^{(<h)} \equiv \{x_v^{(i)}\}_{i=1}^{h-1}$ [1]. The acquisition sub-policy will begin with $x_v^{(h)} = \varnothing$. In the sequential setting $x_v^{(h)}$ shall be updated with each acquisition sub-step (with the acquired feature values from state $s^{(h)}$); in the batch acquisition setting, the observation $x_v^{(h)}$ is updated only once after all specified acquisitions are made. Given the available information, we utilize the sequential generative POSS model (§3.1) to predict the unobserved features for state $s^{(h)}$, i.e., $x_u^{(h)}$. We first sample a latent code from the prior $p(z \mid \{(i, x_v^{(i)}, a_c^{(i-1)})\}_{i=1}^h)$, then pass the latent code through the decoder only for state $s^{(h)}$ to obtain the distribution $p(x_u^{(h)} \mid z, t, x_v^{(h)}, a_c^{(h-1)})$, to sample the unobserved features $x_u^{(h)}$.

The distribution over the unobserved features may have multiple modes, therefore, using one sample may not accurately represent the beliefs. We instead perform multiple imputations by sampling multiple latent codes. The belief at time step h can then be represented as a set of imputed features, i.e., $b^{(h)} = \{(x_v^{(h)}, \hat{x}_u^{(h)})_n\}_{n=1}^N$, where N is the number of samples of the unobserved features.

[1] At time $i-1$, the agent might have taken a feature acquisition action, so $a_c^{(i-1)}$ might be undefined. For notation simplicity, here $a_c^{(i-1)}$ actually means the last task action the agent have taken before time $i-1$.



(a) Batch Acquisition (b) Sequential Acquisition

Figure 2: Illustrations of the batch acquisition process and the sequential acquisition process. The dashed lines indicate the update of the belief. The red arrows represent the hierarchical policy execution processes.

3.3 Cost Sensitive Reinforcement Learning

Given the sequential transition model and the belief estimates, we now build the RL agent to solve the AA-POMDP. We decompose the agent into two policies, the feature acquisition policy π_f and the task policy π_c, which are then combined in a hierarchical fashion. Both policies take the belief estimation set $b^{(h)}$ as inputs and derive the action distribution in a permutation invariant manner.

At any underlying state $s^{(h)}$, we first run the feature acquisition policy π_f to collect information. The features are acquired either in a batch or one-by-one depending on the acquisition setting. In the batch acquisition setting, the acquisition policy is ran once to determine the set of features to be acquired, while in the sequential acquisition setting, the acquisition policy is run multiple times sequentially. The belief estimations are updated after acquiring the features. The task policy π_c is then executed based on the updated belief using the acquired features (see Fig. 2 for illustrations).

Our goal in the AA-POMDP is to maximize the task reward while minimizing the feature acquisition cost. In the hierarchical setting, we decompose the goal as well. The high-level task policy aims at achieving as high task reward as possible based on the acquired features, while the low-level acquisition policy aims at providing sufficient information and minimizing the acquisition cost. Therefore, the reward for the task policy at time step h is defined as $r_c^{(h)} = \mathcal{R}(s^{(h)}, a_c^{(h)})$, which is the same as the original task reward. For the acquisition policy, in addition to the acquisition cost, we desire the acquired features to support the task policy. First, the task policy should produce confident action choice based on the acquired features. There-

fore, we use the negative entropy of the task policy as a reward to the acquisition policy, i.e., $-\text{Ent}(\pi_c(b^{(h)}))$. Second, the acquired features as inputs to the task policy should lead to a high value estimation indicating high long-term return of the task policy. Therefore, we employ the task value estimation, $V_c(b^{(h)})$, as an additional reward. Third, the acquired features should be indicative of the unobserved features so that the belief estimation is accurate. We hence use the imputation accuracy as one of the acquisition rewards, i.e., $\text{Acc}(x_u^{(h)}, \hat{x}_u^{(h)})$. For discrete features, the accuracy is evaluated as the average exact mach accuracy of the N belief samples. For continuous features, the accuracy is evaluated as the average negative MSE of the N belief samples. In total, the reward for the acquisition policy at time step h is defined as

$$\begin{aligned} r_f^{(h)} = &- \mathcal{C}(s^{(h)}, a_f^{(h)}) - \omega_e \cdot \text{Ent}(\pi_c(b^{(h)})) \\ &+ \omega_v \cdot V_c(b^{(h)}) + \omega_a \cdot \text{Acc}(x_u^{(h)}, \hat{x}_u^{(h)}), \end{aligned} \tag{11}$$

where ω_e, ω_v and ω_a are hyperparameters for weighting the corresponding terms. In the batch acquisition setting, all features in $x_v^{(h)}$ are acquired simultaneously, thus the rewards are received immediately after the acquisition. In the sequential acquisition setting, however, each acquisition action will only receive its cost as immediate reward, while the other reward terms are granted only when the agent selects the termination action ϕ.

3.4 Implementation

In this section, we describe several important implementation details. We use PPO (Schulman et al., 2017) for both the acquisition policy and the task policy. The actor and the critic networks are implemented as an ensemble over the belief sets, where the action probabilities and values are averaged over the belief set elements. The sequential generative model is implemented as a VAE with normalizing flow based posterior and prior, of which the base distribution are Gaussian conditioned on the corresponding sets. We use Set Transformers (Lee et al., 2019) to extract the set representations. The time indexes are embedded using sinusoidal functions as in other Transformer models (Vaswani et al., 2017). For continuous actions and features, we directly concatenate them. For discrete actions and features, we learn their embeddings jointly. During training, we first train the sequential generative model with trajectories obtained by randomly acquired features and random task actions; then we pre-train the task policy with fixed generative model and random acquisitions; finally we train the generative model and both policies jointly.

4 Related Works

Active Feature Acquisition Active feature acquisition involves acquiring features at a cost to predict a target variable. Prior works (Zubek and Dietterich, 2002; Rückstieß et al., 2011; Shim et al., 2018; Yoon et al., 2019; Chang et al., 2019) formulate AFA as an MDP and develop various RL approaches to optimize the acquisition plan. Li and Oliva (2021a) further propose a model-based solution by leveraging ACFlow (Li et al., 2020a) to model the AFA dynamics. Zannone et al. (2019) propose to learn the acquisition policy using augmented data sampled from a pretrained Partial VAE (Ma et al., 2019). He et al. (2012) and He et al. (2016) instead employ the imitation learning approach guided by a greedy reference policy to learn the acquisition policy. In addition to RL based approaches, Ling et al. (2004), Chai et al. (2004) and Nan et al. (2014) propose decision tree, naive Bayes, and maximum margin based classifiers, respectively, to jointly minimize the misclassification cost and feature acquisition cost. Ma et al. (2019), Gong et al. (2019) and Li and Oliva (2020) propose to acquire features greedily using mutual information as the estimated utility. Unlike previous AFA works, our setting does not contain a specific target variable; instead, we focus on optimizing the cumulative reward of MDP. Furthermore, our agent learns the feature acquisition policy and the task policy simultaneously.

POMDP and Temporal Dynamics Modeling Learning in POMDPs without an environment model is more challenging than in MDPs (Papadimitriou and Tsitsiklis, 1987). Many works focuse on planning in POMDPs with known environment model (Littman et al., 1995; Hauskrecht, 2000; Pineau et al., 2003; Ross et al., 2007a, 2008; Kurniawati et al., 2008; Silver and Veness, 2010; Somani et al., 2013; Bai et al., 2014; Sunberg and Kochenderfer, 2018). Bayes-Adaptive POMDP (Ross et al., 2007b, 2011; Katt et al., 2018) instead learn the environment model in a Bayesian fashion by assuming access to an informative prior over the observation model and plan using posterior belief distributions over states. Instead of planing with an environment model, Deep Recurrent Q-Networks (DRQN) (Hausknecht and Stone, 2015) and its variants (Zhu et al., 2017) parameterize the value function with a recurrent neural network that takes in the action and observation history. Deep Variational Reinforcement Learning (DVRL) (Igl et al., 2018) uses the action and observations history to learns a VAE model, where the latent variable is interpreted as the belief. TD-VAE (Gregor et al., 2018) builds a VAE model to predict the belief state for time points separated by random intervals. Their jumpy state modeling enables the prediction of belief at arbitrary future time

without the step-by-step rollout.

Outside of POMDP literature, there is a number of works that consider jumps when modeling temporal dynamics. Koutnik et al. (2014) and Chung et al. (2016) equip recurrent neural network with skip connections, which makes it easier to bridge distant time steps. Buesing et al. (2018) temporally sub-sample the data with fixed jump interval and build models on the subsampled data. One of the limitations of the subsampling is that the model cannot leverage information contained in the skipped observations. Neitz et al. (2018) and Jayaraman et al. (2018) predict sequences with variable time-skips, by choosing the most predictable future frames as target.

Due to the feature acquisition, our setting makes the agent observe only a subset of features, thus following a POMDP, but with the difference that the observation is controlled by the agent itself rather than the environment. In order to infer the belief and assist policy learning, we develop a VAE model to impute the missing features. In our model, the action and observation history together with their timestamps are treated as a permutation invariant set. The set perspective enables our model to directly predict the belief at arbitrary time step without resorting to the stepwise rollout.

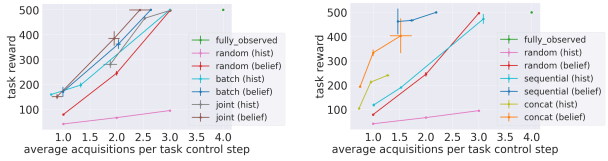
Cost Sensitive Reinforcement Learning Previous works have considered learning agent to decide *what* and *when* to observe when the observation has a cost. Zubek and Dietterich (2000) introduce even-odd POMDP, assuming full observability every other step. They then convert it into an equivalent MDP, whose value function captures the sensing costs of the original POMDP. Zubek et al. (2004) propose the Cost Observable MDP (COMDP), where the actions are partitioned into state-changing actions and pure observation actions. The reward function is modified to incorporate observation costs. The COMDP is conceptually similar to our AA-POMDP, but their algorithm focuses solely on tabular environments and the batch acquisition scenario. Yin et al. (2020) study the batch acquisition scenario in continuous-state POMDP, using sequential VAE model to extract representation form the action and observation history, which is then used to guide the RL policy. Bellinger et al. (2020) propose to learn a policy and a state estimator in parallel, allowing the agent to either pays the cost for full observation or rely on the estimated state for free. Nam et al. (2021) introduce Action-Contingent Noiselessly Observable MDPs (ACNO-MDPs), where the agent can fully observe the state at a cost or act based on past observations. Bellinger et al. (2021) consider a similar intermittently observed scenario and provide a in-depth qualitative analysis of agents' measurement patterns. In contrast, our work generalizes ACNO-MDPs by allowing agents to observe an arbitrary subset of features in both batch and sequential acquisition scenarios.

5 Experiments

We evaluate batch acquisition and sequential acquisition scenarios on several benchmark environments. For context, we provide the rewards stemming from a task policy on `fully_observed` states. Additionally, we also tested a typical POMDP setting where a random subset of features were observed (`random*`). We vary the cost per acquisition to demonstrate the trade-off between acquisition cost and task reward. We evaluate both the batch acquisition setting and sequential acquisition setting with our proposed cost-sensitive hierarchical PPO (CS-HPPO) (§ 3.3), where the belief state is estimated by POSS (§ 3.1). In order to verify the benefits of our belief estimation, we compare to the variants that replace belief with the observation history, which is a typical practice in POMDP literature (McCallum, 1993), i.e., PPO agents select an action at each step based on either the belief estimation (`belief`) or the observation history (`hist`). Inspired by (Nam et al., 2021), we compare our batch acquisition models to a setting where the action space is the Cartesian product of task control actions and feature acquisition actions (`joint*`). In this setting, the acquisition action controls what will be observed in next state. For sequential acquisition setting, we also compare to a setting that concatenates feature acquisition actions and task control actions into a larger action space (`concat*`). We attempted to evaluate a generic POMDP-RL algorithm, DRQN (Hausknecht and Stone, 2015), in the setting of concatenated action space as well, but found that it fails to learn effective acquisition policy. Please find more details in Appendix D.

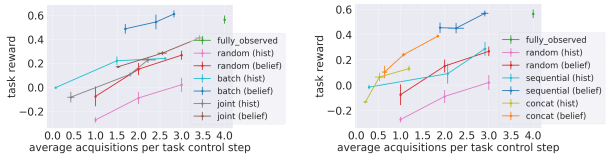
Partially Observed CartPole First, we evaluate on a modified OpenAI gym CartPole-v1 environment, where the features of a state can be dynamically acquired with a cost. In the batch acquisition setting, the action space contains 16 acquisition actions and 2 task control actions, while in the sequential acquisition setting, the acquisition action spaces contains the four measurable features plus a termination action.

Sepsis Simulator This environment simulates a Sepsis patient and the effects of several common treatments (Oberst and Sontag, 2019). The task is to apply three treatment actions, antibiotic, ventilation and vasopressors, to the ICU patients. Therefore, the task action space is the powerset of the three treatments. Each patient has eight features including one indicator of the patent's diabetes condition, three indicators



(a) batch (b) sequential

Figure 3: CartPole results



(a) batch (b) sequential

Figure 4: Sepsis results.

of the current treatment state and four measurable states over heart rate, sysBP rate, percoxyg state and glucose level. We only allow the agent to acquire the four measurable states and regard the rest of features as given. Therefore, in the batch acquisition setting, the acquisition action space contains the powerset of the four measurable features, i.e., 16 acquisition actions in total; in the sequential acquisition setting, the acquisition action space contains the four measurable features plus a termination action. We limit the maximum treatment steps to 30. The patient will be discharged if his/her measurement states all return to normal values, which gives the agent a 1.0 reward. An episode will also terminate upon mortality with a reward -1.0.

Results Figure 3 and 4 demonstrate the results for CartPole and Sepsis respectively. We run each acquisition setting with three acquisition costs and three random seeds. For each acquisition cost, we plot the average task reward for the original control task against the average number of acquired features for each task control action. We can see that in every acquisition setting (either batch acquisition, sequential acquisition, random acquisition, acquisition in concatenated action space, or acquisition in joint action space), the agent equipped with the belief estimation performs much better than the ones using observation history. During training, we also observed the belief estimation help stabilize the training and converges to the optimal policy quickly. Please see Fig. 5 for an example of the training curve. For agents that have access to the belief estimation, our proposed CSHPPO almost always outperforms the non-hierarchical policies in both batch acquisition and sequential acquisition settings. One exception is the batch acquisition setting in CartPole environment, where acquisition in the joint action space performs better. We believe it is due to the relatively small action space (only 32 actions in the joint action space) so that the non-hierarchical policy is good enough to explore the space while the hierarchical one introduces additional complexity.

Table 1: Prediction accuracy for in-distribution (ID) and oud-of-distribution (OOD) time steps.

	ID	OOD
Seq-PO-VAE	93.32	75.12
POSS	**94.49**	**78.54**

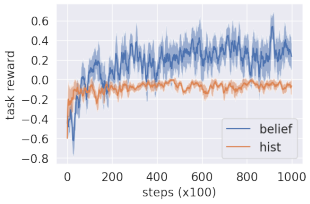


Figure 5: Training curve.

Ablation Studies As an ablation study, we compare the prediction accuracy of our proposed sequential generative model (POSS) to the Seq-PO-VAE proposed in (Yin et al., 2020). For a fair comparison, we augment Seq-PO-VAE with normalizing flow based prior and posterior distributions as in POSS. We train both models on 1000 trajectories collected from Sepsis simulator with random acquisition policy and random task policy and test on the held-out 100 trajectories. To evaluate the generalizability, we train models using only observations from the first 15 trajectory steps and test the prediction accuracy on both in-distribution (ID) time steps (≤ 15) and out-of-distribution (OOD) time steps (> 15). Table 1 shows that POSS outperforms Seq-PO-VAE on both ID and OOD time steps, verifying the superiority of our proposed set based sequence modeling approach. Please see Appendix D.5 for additional ablation studies on the intrinsic rewards.

6 Conclusion

In this work, we study the sequential decision making problems with feature acquisition costs. We present a special MDP named AA-POMDP and identify two types of the feature acquisition settings, batch acquisition and sequential acquisition, which are applicable under different conditions. To help solve the partially observed problem, we develop a sequential generative model to capture the state transitions multiple imputation of the unobserved features. The agent then takes a set of imputed observations as the belief estimation. In order to balance the acquisition cost with the task reward, we propose a hierarchical formulation of the policy, where the low-level policy is respon-

sible for acquiring features and the high-level policy maximizes the task reward based on the acquired feature subsets. The entire framework, including both the generative model and two levels of the policies, is trained jointly. We conduct extensive experiments and demonstrate state-of-the-art performance.

Acknowledgements

This research was partly funded by NSF grants IIS2133595, DMS2324394, and by NIH grants 1R01AA02687901A1, 1OT2OD032581-02-321, 1OT2OD038045-01.

References

Bai, H., Hsu, D., and Lee, W. S. (2014). Integrated perception and planning in the continuous space: A pomdp approach. *The International Journal of Robotics Research*, 33(9):1288–1302.

Bellinger, C., Coles, R., Crowley, M., and Tamblyn, I. (2020). Active measure reinforcement learning for observation cost minimization. *arXiv preprint arXiv:2005.12697*.

Bellinger, C., Drozdyuk, A., Crowley, M., and Tamblyn, I. (2021). Scientific discovery and the cost of measurement–balancing information and cost in reinforcement learning. *arXiv preprint arXiv:2112.07535*.

Bender, C., O'Connor, K., Li, Y., Garcia, J., Oliva, J., and Zaheer, M. (2020). Exchangeable generative models with flow scans. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10053–10060.

Biloš, M. and Günnemann, S. (2021). Scalable normalizing flows for permutation invariant densities. In *International Conference on Machine Learning*, pages 957–967. PMLR.

Buesing, L., Weber, T., Racaniere, S., Eslami, S., Rezende, D., Reichert, D. P., Viola, F., Besse, F., Gregor, K., Hassabis, D., et al. (2018). Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*.

Chai, X., Deng, L., Yang, Q., and Ling, C. X. (2004). Test-cost sensitive naive bayes classification. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 51–58. IEEE.

Chang, C.-H., Mai, M., and Goldenberg, A. (2019). Dynamic measurement scheduling for event forecasting using deep rl. In *International Conference on Machine Learning*, pages 951–960. PMLR.

Chung, J., Ahn, S., and Bengio, Y. (2016). Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28.

Coronato, A., Naeem, M., De Pietro, G., and Paragliola, G. (2020). Reinforcement learning for intelligent healthcare applications: A survey. *Artificial Intelligence in Medicine*, 109:101964.

Diaconis, P. and Freedman, D. (1980). Finite exchangeable sequences. *The Annals of Probability*, pages 745–764.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019a). Neural spline flows. *Advances in neural information processing systems*, 32.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019b). Neural spline flows. *ArXiv*, abs/1906.04032.

Edwards, H. and Storkey, A. (2017). Towards a neural statistician. In *International Conference on Learning Representations*.

Fang, L., Zeng, T., Liu, C., Bo, L., Dong, W., and Chen, C. (2021). Transformer-based conditional variational autoencoder for controllable story generation. *arXiv preprint arXiv:2101.00828*.

Finn, C., Levine, S., and Abbeel, P. (2016). Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR.

Gong, W., Tschiatschek, S., Nowozin, S., Turner, R. E., Hernández-Lobato, J. M., and Zhang, C. (2019). Icebreaker: Element-wise efficient information acquisition with a bayesian deep latent gaussian model. In *Advances in Neural Information Processing Systems*, pages 14820–14831.

Gregor, K., Papamakarios, G., Besse, F., Buesing, L., and Weber, T. (2018). Temporal difference variational auto-encoder. *arXiv preprint arXiv:1806.03107*.

Haarnoja, T., Pong, V., Zhou, A., Dalal, M., Abbeel, P., and Levine, S. (2018). Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6244–6251. IEEE.

Hausknecht, M. and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*.

Hauskrecht, M. (2000). Value-function approximations for partially observable markov decision processes. *Journal of artificial intelligence research*, 13:33–94.

He, H., Eisner, J., and Daume, H. (2012). Imitation learning by coaching. In *Advances in Neural Information Processing Systems*, pages 3149–3157.

He, H., Mineiro, P., and Karampatziakis, N. (2016). Active information acquisition. *arXiv preprint arXiv:1602.02181*.

Igl, M., Zintgraf, L., Le, T. A., Wood, F., and Whiteson, S. (2018). Deep variational reinforcement learning for pomdps. In *International Conference on Machine Learning*, pages 2117–2126. PMLR.

Jayaraman, D., Ebert, F., Efros, A. A., and Levine, S. (2018). Time-agnostic prediction: Predicting predictable video frames. *arXiv preprint arXiv:1808.07784*.

Katt, S., Oliehoek, F., and Amato, C. (2018). Bayesian reinforcement learning in factored pomdps. *arXiv preprint arXiv:1811.05612*.

Kerns, G. J. and Székely, G. J. (2006). Definetti's theorem for abstract finite exchangeable sequences. *Journal of Theoretical Probability*, 19(3):589–608.

Kim, J., Yoo, J., Lee, J., and Hong, S. (2021). Setvae: Learning hierarchical composition for generative modeling of set-structured data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15059–15068.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751.

Koutnik, J., Greff, K., Gomez, F., and Schmidhuber, J. (2014). A clockwork rnn. In *International Conference on Machine Learning*, pages 1863–1871. PMLR.

Kurniawati, H., Hsu, D., and Lee, W. S. (2008). Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Citeseer.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3744–3753.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373.

Li, Y., Akbar, S., and Oliva, J. (2020a). ACFlow: Flow models for arbitrary conditional likelihoods. In *Proceedings of the 37th International Conference on Machine Learning*.

Li, Y. and Oliva, J. (2021a). Active feature acquisition with generative surrogate models. In *International Conference on Machine Learning*, pages 6450–6459. PMLR.

Li, Y. and Oliva, J. (2021b). Partially observed exchangeable modeling. In *International Conference on Machine Learning*, pages 6460–6470. PMLR.

Li, Y. and Oliva, J. B. (2020). Dynamic feature acquisition with arbitrary conditional flows. *arXiv preprint arXiv:2006.07701*.

Li, Y., Yi, H., Bender, C., Shan, S., and Oliva, J. B. (2020b). Exchangeable neural ode for set modeling. *Advances in Neural Information Processing Systems*, 33.

Ling, C. X., Yang, Q., Wang, J., and Zhang, S. (2004). Decision trees with minimal costs. In *Proceedings of the twenty-first international conference on Machine learning*, page 69.

Ling, Y., Hasan, S. A., Datla, V., Qadir, A., Lee, K., Liu, J., and Farri, O. (2017). Learning to diagnose: assimilating clinical narratives using deep reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 895–905.

Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. (1995). Learning policies for partially observable environments: Scaling up. In *Machine Learning Proceedings 1995*, pages 362–370. Elsevier.

Ma, C., Tschiatschek, S., Palla, K., Hernandez-Lobato, J. M., Nowozin, S., and Zhang, C. (2019). Eddi: Efficient dynamic discovery of high-value information with partial vae. In *International Conference on Machine Learning*, pages 4234–4243. PMLR.

McCallum, A. (1993). Overcoming incomplete perception with utile distinction memory. In *International Conference on Machine Learning*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Monahan, G. E. (1982). State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management science*, 28(1):1–16.

Nam, H. A., Fleming, S., and Brunskill, E. (2021). Reinforcement learning with state observation costs in action-contingent noiselessly observable markov decision processes. *Advances in Neural Information Processing Systems*, 34:15650–15666.

Nan, F., Wang, J., Trapeznikov, K., and Saligrama, V. (2014). Fast margin-based cost-sensitive classification. In *2014 IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP), pages 2952–2956. IEEE.

Neitz, A., Parascandolo, G., Bauer, S., and Schölkopf, B. (2018). Adaptive skip intervals: Temporal abstraction for recurrent dynamical models. *Advances in Neural Information Processing Systems*, 31.

Niroui, F., Zhang, K., Kashino, Z., and Nejat, G. (2019). Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. *IEEE Robotics and Automation Letters*, 4(2):610–617.

Oberst, M. and Sontag, D. (2019). Counterfactual off-policy evaluation with gumbel-max structural causal models. In *International Conference on Machine Learning*, pages 4881–4890. PMLR.

Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450.

Peng, Y.-S., Tang, K.-F., Lin, H.-T., and Chang, E. (2018). Refuel: Exploring sparse features in deep reinforcement learning for fast disease diagnosis. *Advances in neural information processing systems*, 31.

Petrovich, M., Black, M. J., and Varol, G. (2021). Action-conditioned 3d human motion synthesis with transformer vae. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10985–10995.

Pineau, J., Gordon, G., Thrun, S., et al. (2003). Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032. Citeseer.

Polydoros, A. S. and Nalpantidis, L. (2017). Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173.

Ross, S., Chaib-Draa, B., et al. (2007a). Aems: An anytime online search algorithm for approximate policy refinement in large pomdps. In *IJCAI*, pages 2592–2598.

Ross, S., Chaib-draa, B., and Pineau, J. (2007b). Bayes-adaptive pomdps. *Advances in neural information processing systems*, 20.

Ross, S., Pineau, J., Chaib-draa, B., and Kreitmann, P. (2011). A bayesian approach for learning and planning in partially observable markov decision processes. *Journal of Machine Learning Research*, 12(5).

Ross, S., Pineau, J., Paquet, S., and Chaib-Draa, B. (2008). Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research*, 32:663–704.

Rückstieß, T., Osendorfer, C., and van der Smagt, P. (2011). Sequential feature selection for classification. In *Australasian Joint Conference on Artificial Intelligence*, pages 132–141. Springer.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shan, S., Li, Y., and Oliva, J. B. (2021). Nrtsi: Non-recurrent time series imputation. *arXiv preprint arXiv:2102.03340*.

Shim, H., Hwang, S. J., and Yang, E. (2018). Joint active feature acquisition and classification with variable-size set encoding. In *Advances in neural information processing systems*, pages 1368–1378.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Silver, D. and Veness, J. (2010). Monte-carlo planning in large pomdps. *Advances in neural information processing systems*, 23.

Somani, A., Ye, N., Hsu, D., and Lee, W. S. (2013). Despot: Online pomdp planning with regularization. *Advances in neural information processing systems*, 26.

Spaan, M. T. (2012). Partially observable markov decision processes. In *Reinforcement Learning*, pages 387–414. Springer.

Sunberg, Z. N. and Kochenderfer, M. J. (2018). Online algorithms for pomdps with continuous state, action, and observation spaces. In *Twenty-Eighth International Conference on Automated Planning and Scheduling*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Yin, H., Li, Y., Pan, S. J., Zhang, C., and Tschiatschek, S. (2020). Reinforcement learning with efficient active feature acquisition. *arXiv preprint arXiv:2011.00825*.

Yoon, J., Jordon, J., and Schaar, M. (2019). Asac: Active sensing using actor-critic models. In *Machine Learning for Healthcare Conference*, pages 451–473. PMLR.

Yu, C., Liu, J., Nemati, S., and Yin, G. (2021). Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36.

Zannone, S., Hernandez Lobato, J. M., Zhang, C., and Palla, K. (2019). Odin: Optimal discovery of high-value information using model-based deep reinforcement learning. In *Real-world Sequential Decision Making Workshop, ICML*.

Zhu, P., Li, X., Poupart, P., and Miao, G. (2017). On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1704.07978*.

Zhu, Y., Min, M. R., Kadav, A., and Graf, H. P. (2020). S3vae: Self-supervised sequential vae for representation disentanglement and data generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6538–6547.

Zubek, V. B. and Dietterich, T. (2000). A pomdp approximation algorithm that anticipates the need to observe. In *Pacific Rim International Conference on Artificial Intelligence*, pages 521–532. Springer.

Zubek, V. B. and Dietterich, T. G. (2002). Pruning improves heuristic search for cost-sensitive learning. In *ICML*.

Zubek, V. B., Dietterich, T. G., et al. (2004). Two heuristics for solving pomdps having a delayed need to observe.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] All code will be released once the paper is accepted.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] All code will be released once the paper is accepted.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure 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, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Partially Observed Set Models for Sequences (POSS)

As discussed in Sec. 3.1, we formulate the partially observed sequence modeling task (5) as a set modeling task (7) for a set $\mathbf{ax} := \{(t, x_v^{(t)}, x_u^{(t)}, a_c^{(t-1)})\}_{t=1}^T$. According to De Finetti's theorem, there exists a latent code z such that the set elements are conditionally independent conditioned on z. See derivation in equation A.1.

The equation (1) applies the De Finetti's theorem again. Since $x_v^{(t)}$ contains a subset of features at time step t, the same latent variable z that factors the set element $(t, x_v^{(t)}, x_u^{(t)}, a_c^{(t-1)})$ conditionally independent will also factor $(t, x_v^{(t)}, a_c^{(t-1)})$ conditionally independent.

Divide both sides with

$$p(\mathbf{ax}_v) := p(\{(t, x_v^{(t)}, a_c^{(t-1)})\}_{t=1}^T),$$

we have

$$\begin{aligned} &p(\mathbf{x}_u \mid \mathbf{ax}_v) \\ &= \int \prod_{t=1}^{T} \left[p(x_u^{(t)} \mid t, x_v^{(t)}, a_c^{(t-1)}, z) \right] p(z \mid \mathbf{ax}_v) dz. \end{aligned} \tag{A.2}$$

To optimize A.2, we resort to the variational approach and optimize a lower bound (9). The prior $p(z \mid \mathbf{ax}_v)$ and posterior $q(z \mid \mathbf{ax})$ are permutation invariant w.r.t. their inputs $\mathbf{ax}_v$ and $\mathbf{ax}$ respectively. To obtain accurate estimations of the prior and posterior, we utilize normalizing flow based distributions where the base distributions are parameterized as Gaussian distributions with mean and variance derived from $\mathbf{ax}_v$ and $\mathbf{ax}$ using Set Transformers. Due to the permutation invariant architecture of Set transformer, the Gaussian base distribution is permutation invariant; and since the transformations are invertible, the ultimate normalizing flow based distributions are permutation invariant as well. Please see Fig. A.1 for an illustration of our proposed POSS model.

B Motivating Examples of Sequential vs Batch Acquisition Settings

To better illustrate the practical differences between sequential and batch acquisition settings, we provide several real-world application examples that highlight when each approach is most appropriate.

In medical diagnosis and treatment scenarios, both acquisition patterns have important use cases. Sequential acquisition is well-suited for non-emergency medical scenarios, where a doctor can order diagnostic tests one after another, analyzing each result before determining the next test to perform. This approach allows physicians to minimize unnecessary tests and their associated costs/risks by adapting their testing strategy based on previously acquired information. However, in emergency situations like trauma cases, batch acquisition becomes necessary - doctors must order multiple tests (blood work, CT scans, X-rays) simultaneously to gather critical information quickly for immediate treatment decisions.

Geological exploration provides another illustrative example of these different approaches. Sequential acquisition is often employed during initial site exploration, where geologists conduct preliminary surveys and then strategically acquire additional data (seismic readings, core samples) based on their findings. This sequential strategy helps optimize resource allocation by focusing detailed investigation on promising areas. In contrast, offshore oil exploration often requires batch acquisition, where multiple types of data (seismic surveys, gravity measurements) must be collected simultaneously across a large area. This batch approach is necessary when the acquisition process itself may modify the environment or when environmental conditions (like ocean currents) are rapidly changing.

In autonomous driving applications, both acquisition patterns also find natural use cases. Sequential acquisition can be appropriate during normal navigation, where a vehicle can methodically process different sensor inputs (camera, LIDAR, radar) one after another, adjusting its acquisition strategy based on previously processed data. However, batch acquisition becomes critical in time-sensitive scenarios like highway merging or complex intersection navigation, where multiple sensor inputs must be processed simultaneously to enable rapid decision-making for safe operation.

These examples demonstrate that sequential acquisition is most appropriate when the feature acquisition process does not modify the underlying state and there is sufficient time to acquire features iteratively. Batch acquisition, on the other hand, becomes necessary in time-critical situations, when the acquisition process itself can alter the state, or when the state changes too rapidly relative to the acquisition time to allow for sequential sampling.

C Algorithm Details

C.1 Integration of POSS with PPO

Our algorithm combines a Partially Observed Set model for Sequences (POSS) with Proximal Policy Optimization (PPO) in a hierarchical framework. At each decision step h, the process proceeds as follows:

$$
\begin{aligned}
p(\mathbf{ax}) &= p(\{(t, x_v^{(t)}, x_u^{(t)}, a_c^{(t-1)})\}_{t=1}^T) \\
&= \int \prod_{t=1}^T \left[p(t, x_v^{(t)}, x_u^{(t)}, a_c^{(t-1)} \mid z) \right] p(z) dz \\
&= \int \prod_{t=1}^T \left[p(x_u^{(t)} \mid t, x_v^{(t)}, a_c^{(t-1)}, z) p(t, x_v^{(t)}, a_c^{(t-1)} \mid z) \right] p(z) dz \\
&= \int \prod_{t=1}^T \left[p(x_u^{(t)} \mid t, x_v^{(t)}, a_c^{(t-1)}, z) \right] \prod_{t=1}^T \left[p(t, x_v^{(t)}, a_c^{(t-1)} \mid z) \right] p(z) dz \\
&\overset{(1)}{=} \int \prod_{t=1}^T \left[p(x_u^{(t)} \mid t, x_v^{(t)}, a_c^{(t-1)}, z) \right] p(\{(t, x_v^{(t)}, a_c^{(t-1)})\}_{t=1}^T \mid z) p(z) dz \\
&= \int \prod_{t=1}^T \left[p(x_u^{(t)} \mid t, x_v^{(t)}, a_c^{(t-1)}, z) \right] p(z \mid \{(t, x_v^{(t)}, a_c^{(t-1)})\}_{t=1}^T) p(\{(t, x_v^{(t)}, a_c^{(t-1)})\}_{t=1}^T) dz \\
&\equiv \int \prod_{t=1}^T \left[p(x_u^{(t)} \mid t, x_v^{(t)}, a_c^{(t-1)}, z) \right] p(z \mid \mathbf{ax}_v) p(\mathbf{ax}_v) dz.
\end{aligned}
\tag{A.1}
$$

1. **Belief State Sampling:** Given the observation history $o^{(<h)}$ and action history $a_c^{(<h)}$, POSS samples N different imputations of the unobserved features to represent the belief state $b^{(h)} = \{(x_v^{(h)}, \hat{x}_u^{(h)})_n\}_{n=1}^N$.
2. **Feature Acquisition:** The acquisition policy π_f takes the belief state $b^{(h)}$ as input and outputs either:
   - A set of features to acquire simultaneously (batch setting)
   - A sequence of features to acquire one by one until termination (sequential setting)
3. **Belief Update:** After each acquisition (or batch of acquisitions), POSS updates the belief state by incorporating the newly acquired features.
4. **Task Execution:** The task policy π_c takes the updated belief state as input and selects the task action $a_c^{(h)}$.

C.2 Computational Complexity

The computational complexity of our approach scales primarily with:

- Trajectory length T
- Number of belief samples N
- Feature dimension d

At each step, POSS generates N belief samples in parallel through a single forward pass. The policy networks process these samples in a permutation-invariant manner, resulting in $O(T)$ complexity with respect to trajectory length. Both POSS and the policy networks use lightweight architectures, enabling efficient practical implementation.

C.3 Belief Utilization in PPO

The algorithm demonstrates how beliefs are maintained and utilized throughout training. Both policies process beliefs through permutation-invariant networks to handle the set-based nature of multiple imputations. The acquisition policy receives additional reward signals based on belief quality and task policy confidence as detailed in Equation 11 of the main paper.

D Experiment

In this section, we evaluate both the batch acquisition setting and the sequential acquisition setting with our proposed cost-sensitive hierarchical PPO (CS-HPPO). The following are a list of settings we experimented:

Fully Observed In this setting, the agent only needs the task policy π_c since all features will be observed at each time step. The task policy takes the full observation as input and no belief estimation is needed either.

Random Acquisition Since there will not be an acquisition policy, we cannot control the number of acquisitions using cost. Instead, we set a fixed budget so that the agent can observe part of the features that is selected at random. We evaluate two variants of this setting where the task policy takes input from either

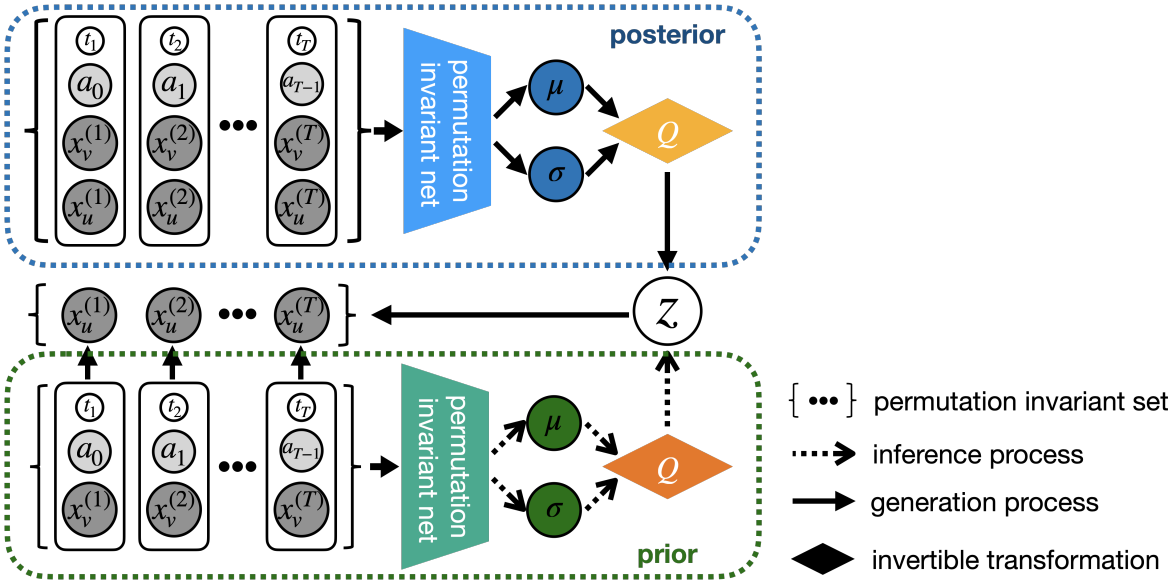


Figure A.1: Model partially observed trajectories via set based VAE.

the previous 8 observations or the belief estimated by POSS.

Batch Acquisition When the agent arrives at state $s^{(h)}$, the agent first runs the acquisition policy π_f to select a set of features to acquire; then, based on the acquired features, the agent runs the task policy π_c to perform the actual task. If the agent is equipped with POSS, the belief is updated right after each acquisition so that the next task action is based on the updated belief; otherwise, the agent takes the previous 8 observations as input for both acquisition policy and task policy.

Joint Action Space Acquisition in joint action space is meant to be a baseline for batch acquisition, where we combine the batch acquisition actions and task actions by their Cartesian product to form a joint action space. At the beginning, the agent observes all features and then selects a joint action where the task action transits the environment to a new state and the acquisition action determines what features to observe for next step. The following actions are selected based only on the acquired features. The agent continues this process until the task is terminated. We similarly evaluate two variants of this setting where the inputs to the policy are either the previous observations or the belief estimation.

Sequential Acquisition In the sequential acquisition setting, the agent first runs the acquisition policy to acquire features from the underlying state $s^{(h)}$ and terminates acquisition until it selects the termination action ϕ. Afterwards, the task policy π_c selects a task action based on the acquired features. The inputs to the policies are either belief estimations or observation histories.

Concatenated Action Space As a baseline for sequential acquisition, we evaluate a setting where the acquisition actions and task acquisitions are concatenated. At each step, the agent selects either an acquisition action or a task action. When the agent selects the task action, it automatically terminates the acquisition process and transits the environment to a new state by executing the task action within the environment. Similarly, the inputs to the policy could be belief estimation or observation histories.

D.1 Benchmark Environments

D.1.1 Partially Observed CartPole

The OpenAI gym CartPole-v1 environment contains 4 features (i.e., cart position, cart velocity, pole angle and pole angular velocity) and 2 discrete actions (i.e., push cart to left and push cart to right). In the batch acquisition setting, the action space contains $2^4 = 16$ acquisition actions and 2 task control actions. In the sequential acquisition setting, the acquisition action space contains the 4 measurable features plus a termination action ϕ, and the task action space contains the 2 original task control actions. We conduct experiments with three different costs per feature (0.005, 0.01, and 0.015), and for each cost we report results from 3 independent runs. Since each run might acquire different number of features and achieve different rewards, we report the mean and standard deviation for

Algorithm 1 Belief-based Hierarchical PPO

Require: POSS model M, acquisition policy π_f, task policy π_c

for each iteration **do**
  for each environment step **do**
    $b^{(h)} \leftarrow$ Sample N imputations from M given history
    if batch_acquisition **then**
      $a_f \leftarrow \pi_f(b^{(h)})$ # Select features to acquire in batch
      Acquire features x_v indicated by a_f
      $b^{(h)} \leftarrow$ Update belief with acquired features
    end if
    if sequential_acquisition **then**
      while not terminated **do**
        $a_f \leftarrow \pi_f(b^{(h)})$ # Sequentially select features
        if $a_f = \phi$ **then**
          break
        end if
        Acquire feature indicated by a_f
        $b^{(h)} \leftarrow$ Update belief with acquired feature
      end while
    end if
    $a_c \leftarrow \pi_c(b^{(h)})$ # Execute task action based on final belief
    Execute a_c in environment
    $r_f \leftarrow$ ComputeAcquisitionReward() # Eq. 11
    $r_c \leftarrow$ ComputeTaskReward()
    Update π_f using r_f
    Update π_c using r_c
    [Optional] Update M using acquired features
  end for
end for

both the acquisitions per task action and the task reward.

D.1.2 Sepsis Simulator

As described in Sec. 5, the Sepsis simulator contains 8 features, in which 4 of them can be acquired, while the rest 4 features are given. The agent can take 8 treatment actions. In the batch acquisition setting, the action space contains $2^4 = 16$ acquisition actions and 8 task actions. In the sequential acquisition setting, the acquisition action space contains the 4 measurable features plus a termination action ϕ, and the task action space contains the 8 treatment actions. We conduct experiments with three different costs per acquisition (0.005, 0.01, and 0.02), and report results from 3 independent runs for each cost.

D.2 POSS Implementation

The POSS model contains a prior network, a posterior network, two invertible transformations for prior and posterior respectively and a decoder network. The prior and posterior networks first use 4 permutation equivariant Set Transformer layers with 128 hidden units to extract set based features; then, an attentive pooling layer squashes the set features into a 128 dimensional permutation invariant feature vector; finally, 2 linear layers with 128 hidden units output the mean and variance for the Gaussian base distribution. we set the latent variable dimension to 64. For prior, we stack 4 rational-quadratic coupling transformations to transform the base distribution; and for posterior, we use 4 rational-quadratic autoregressive transformations (Durkan et al., 2019b). For categorical observations, we learn a set of 16 dimensional embeddings for each feature and a special embedding to represent the missing feature. We also embed the discrete actions with 16 dimensional features. The time steps are represented by sinusoidal functions as 16 dimensional features. The inputs for the prior network contain the time step embeddings, the action embeddings and the embeddings of observed features, while the inputs for posterior network contain the time step embedddings, the action embeddings and the embeddings of all features. The decoder network takes the latent code as well as time step embeddings, action embeddings and embeddings for observed features as inputs and outputs a distribution for the unobserved features. For

Component	HyperParameter	CartPole	Sepsis
PPO	γ	0.99	0.99
PPO	λ	0.95	0.95
PPO	clip ratio	0.2	0.2
PPO	reward weight ω_e	1.0	1.0
PPO	reward weight ω_v	0.01	1.0
PPO	reward weight ω_a	100.0	1.0
Policy	actor	Linear: $64 \to 64$	
Policy	critic	Linear: $64 \to 64$	
POSS	prior	SetTransformer: 128×4 + Linear: $128 \to 64$	
POSS	prior transformations	rational-quadratic coupling: 128×4	
POSS	posterior	SetTransformer: 128×4 + Linear: $128 \to 64$	
POSS	posterior transformations	rational-quadratic autoregressive: 128×4	
POSS	decoder	SetTransformer: 128×4 + Linear: $128 \to 128$	
Training	model learning rate	0.0001	0.0001
Training	actor learning rate	0.0003	0.0003
Training	critic learning rate	0.0003	0.0003
Training	grad norm	1.0	1.0

Table D.1: Hyperparameters

categorical features, the decoder outputs logits of a Categorical distribution; and for continuous features, the decoder outputs mean and variance of a Gaussian distribution.

D.3 Policies

In different settings, the policy network will have different type of inputs. In fully observed setting, the policy takes in a vector representation of the observations. When using observation histories, the policy network takes in a set of partially observed features. When using belief estimations, the policy network takes in multiple imputations of the unobserved features. We use a 2-layer linear network to implement both the acquisition policy and the task policy. If the input is a set (history or belief), we obtain the final action distribution with ensemble. For discrete actions, the actor outputs a categorical distribution where the probabilities are the average probability across set elements. For continuous actions, the actor distribution is a Gaussian distribution where the mean is averaged across set elements.

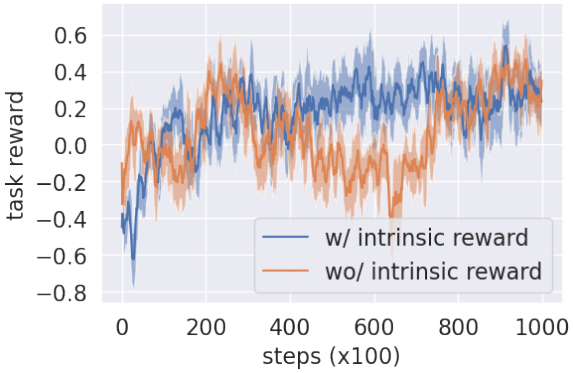
D.4 Hyperparameters

Table D.1 list all the hyperparameters for CartPole and Sepsis environments. Note that we did not conduct any hyperparameter optimization and all hyperparameters are set based on our previous experiences.

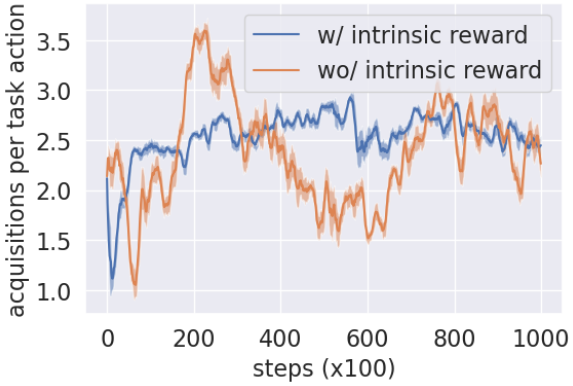
D.5 Additional Ablation Studies

The benefits of our proposed POSS are two folds: First, it provides the agent with an accurate belief estimation so that the agent can make better decisions based solely on the partial observations. Second, the imputation accuracy provides an intrinsic reward to the acquisition policy to guide the agent acquire informative features. We have seen the belief estimation help achieve better reward-cost tread-off compared to observation histories (Sec. 5), and we have verified the advantage of the the set based sequence modeling formulation (Sec. 5). To better understand the benefits of the intrinsic reward, we conduct an ablation study on Sepsis simulator by removing the intrinsic reward (set ω_a to 0).

Figure D.1 compares the training curve for the two agents with and without intrinsic reward. First, we can see the two agent eventually converges to a similar solution (both similar number of acquisitions and similar task reward), which empirically verifies that the intrinsic reward does not affect the optimal policy. Second, we can see the agent trained with intrinsic reward converges much faster and the training is more stable.



(a) task reward



(b) acquisitions per task action

Figure D.1: Compare task reward (a) and average acquisitions per task action (b) for two agents with and without intrinsic reward provided by POSS.