Efficient Reward Identification In Max Entropy Reinforcement Learning with Sparsity and Rank Priors

Mohamad Louai Shehab* Alperen Tercan* Necmiye Ozay

Abstract— **In this paper, we consider the problem of recovering time-varying reward functions from either optimal policies or demonstrations coming from a max entropy reinforcement learning problem. This problem is highly ill-posed without additional assumptions on the underlying rewards. However, in many applications, the rewards are indeed parsimonious, and some prior information is available. We consider two such priors on the rewards: 1) rewards are mostly constant and they change infrequently, 2) rewards can be represented by a linear combination of a small number of feature functions. We first show that the reward identification problem with the former prior can be recast as a sparsification problem subject to linear constraints. Moreover, we give a polynomial-time algorithm that solves this sparsification problem exactly. Then, we show that identifying rewards representable with the minimum number of features can be recast as a rank minimization problem subject to linear constraints, for which convex relaxations of rank can be invoked. In both cases, these observations lead to efficient optimization-based reward identification algorithms. Several examples are given to demonstrate the accuracy of the recovered rewards as well as their generalizability.**

I. INTRODUCTION

Reward identification, or Inverse Reinforcement Learning (IRL), is the problem of learning rewards from data. The premise behind IRL is that the reward function serves as the most succinct representation of an agent's behavior [1]. Learning a reward function from demonstrations allows agents to generalize beyond observed behaviors [1], infer underlying human intentions [2], and capture pairwise preferences [3], [4]. However, like many inverse problems, IRL is inherently ill-posed as there may be infinitely many reward functions consistent with the same observed behavior [5], [6], [7]. For instance, suppose an agent moves from location A to location B. One possible hypothesis is that the agent likes B, but another equally valid hypothesis is that the agent dislikes A. Both reward hypotheses are consistent with the observed behavior, highlighting the fundamental ambiguity in IRL [8].

To address this ambiguity, recent research has shifted toward learning the entire *set* of reward functions that can explain observed behaviors [9], [10], [11], [12]. However, there is generally no consensus on how to select a specific reward function from within this set. The choice is often guided by the requirements of the downstream task or heuristic considerations. For example, Linear Programming IRL [1] and Max Margin IRL [13] select the reward function that makes the demonstrated policy as optimal as possible relative to the next-best alternative, effectively maximizing the opportunity cost—a principle widely studied in economics, where rational agents seek to maximize the value of their chosen actions relative to foregone alternatives [14]. Adversarial IRL [15] aims to find a reward function that generalizes well across environments, often selecting a state-only reward that maximizes transferability. Maximum Entropy IRL [16] selects the reward function that maximizes the likelihood of the observed demonstrations, assuming an entropy-regularized policy model. More recently, [17] introduced a framework for quantitatively selecting the best reward—potentially outside the solution set—based on a given target application.

In this work, we study reward identification in finite-horizon settings with time-varying reward functions. This is an important generalization for real-world applications where rewards evolve over time due to changing preferences, environmental conditions, or task requirements. While most prior IRL methods assume static rewards, the dynamic setting makes IRL even more ill-posed, as reward ambiguity can now arise at every time step [6], [18]. Existing approaches for dynamic rewards either (1) impose restrictive parametric assumptions (e.g., Gaussian random walks [19] or generalized linear models [20]), limiting their expressiveness to predefined reward dynamics, or (2) assume privileged knowledge on the number of underlying reward regimes [21]. Relatedly, learning time-varying objective functions is also considered in the area of inverse optimal control [22], [23].

Building on prior work, we propose a principled framework that systematically incorporates structure-aware priors—such as minimal reward switches and shared feature bases—to resolve ambiguity in time-varying reward identification, enabling flexible yet interpretable reward identification without strong parametric assumptions. Our contributions include (1) a polynomial-time algorithm for recovering minimally switching rewards, (2) a convex relaxation for feature-based reward decomposition, and (3) robustness guarantees under finite-sample policy estimates. Empirical results validate our approach in several gridworld environments, showing improved interpretability and transferability over existing methods.

II. PRELIMINARIES

A. Notation

$\mathbb{R}$ and $\mathbb{N}$ are the sets of real and natural numbers respectively. The identity matrix in $\mathbb{R}^{n\times n}$ is denoted by $\mathbf{I}_n$. The zero matrix in $\mathbb{R}^{m\times n}$ is denoted by $\mathbf{0}_{m\times n}$ (m are n are

This work was supported in part by ONR CLEVR-AI MURI (#N00014-21-1-2431). The authors are with the University of Michigan, Ann Arbor. Emails: {mlshehab,tercan,necmiye}@umich.edu

*These two authors contributed equally.

dropped sometimes when they are clear from context). $\mathbf{1}_m$ is the constant vector of ones in $\mathbb{R}^m$. $\mathbb{I}(x \in X)$ is the indicator function. Given a matrix A, $\mathrm{rank}(A)$ and $\mathrm{colspan}(A)$ denote its rank and column span, respectively. When B is another matrix of compatible dimension, $\begin{bmatrix} A & B \end{bmatrix}$ denotes the horizontal concatenation of A and B, and $A \otimes B$ denotes their Kronecker product. Given a vector space V with a basis $B = \{v_1, \cdots, v_m\}$, $[w]_B$ is the vector representation of w in V. For a set S, $\Delta(S)$ denotes the set of probability distributions over it, and $|S|$ denotes its cardinality.

B. Markov Decision Processes

A Markov Decision Process (MDP) is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mu_0, r, \gamma, T)$, where $\mathcal{S} = \{s^{(1)}, \dots, s^{(n)}\}$ is a finite set of states with cardinality $|\mathcal{S}| = n$; $\mathcal{A} = \{a^{(1)}, \dots, a^{(m)}\}$ is a finite set of actions with cardinality $|\mathcal{A}| = m$; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is a Markov transition kernel; $\mu_0 \in \Delta(S)$ is an initial distribution over the set of states; $r = (r_t)_{t=0}^{T-1}$ is a time-varying reward function where each $r_t : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function at time step t; $\gamma \in [0, 1]$ is a discount factor; and $T \in \mathbb{N}$ is the non-negative time horizon. An MDP without a reward function, denoted $\mathcal{M} \setminus r$, is called an *MDP model*. A policy $\pi_t : \mathcal{S} \to \Delta(\mathcal{A})$ is a function that describes an agent's behavior at time step t by specifying an action distribution at each state. We denote by $\pi = (\pi_t)_{t=0}^{T-1}$ the *time-varying* stochastic policy throughout the entire horizon. A trajectory τ (of length T) is an alternating sequence of states and actions (ending with a state), i.e., $\tau = (s_0, a_0, s_1, a_1, \dots, s_{T-1}, a_{T-1}, s_T)$ with $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$. Under a policy π, a trajectory τ occurs with probability

$$\mathbb{P}^{\pi}_{\mu_0}(\tau) = \mu_0(s_0) \prod_{t=0}^{T-1} \pi_t(a_t|s_t) \prod_{t=0}^{T-1} \mathcal{T}(s_{t+1}|s_t, a_t), \quad (1)$$

which depends on the distribution of initial states, the policy, and the Markov transition kernel. We consider the Maximum Entropy Reinforcement Learning (MaxEntRL) objective given by:

$$J_{\text{MaxEnt}}(\pi; r) = \mathbb{E}^{\pi}_{\mu_0}[\sum_{t=0}^{T-1} \gamma^t \Big(r(s_t, a_t) + \mathcal{H}(\pi_t(.|s_t)) \Big)], \quad (2)$$

where $\mathcal{H}(\pi_t(.|s_t)) = -\sum_{a \in \mathcal{A}} \pi_t(a|s_t) \log(\pi_t(a|s_t))$ is the entropy of the policy π_t. The expectation is with respect to the probability measure $\mathbb{P}^{\pi}_{\mu_0}$. We define the optimal policy π^*_r, corresponding to a reward function r, as the maximizer of (2), i.e.:

$$\pi^*_r = \arg\max_{\pi} J_{\text{MaxEnt}}(\pi; r), \quad (3)$$

which is known to be unique [24] up-to accessible states. The maximum entropy policy is [25], [26], [27]:

$$\pi^*_t(a|s) = \frac{e^{Q^*_t(s,a)}}{\sum_{a' \in \mathcal{A}} e^{Q^*_t(s,a')}} \quad (4)$$

where Q^*_t is the optimal soft Q-function at time step t, given by the following backward-in-time computation:

$$\begin{aligned} Q^*_{T-1}(s, a) &= r_{T-1}(s, a), \\ Q^*_t(s, a) &= r_t(s, a) + \\ &\gamma \mathbb{E}_{s' \sim P(.|s,a)} \underbrace{[\log(\sum_{a' \in \mathcal{A}} \exp(Q^*_t(s', a')))]}_{\triangleq V^*_{t+1}(s')}, \end{aligned} \quad (5)$$

with $s \in \mathcal{S}$, $a \in \mathcal{A}$, for $t < T - 1$. V^*_t is the optimal soft value function at time step t, which is also known as reward-to-go.

C. Inverse Reinforcement Learning

Inverse reinforcement learning is the problem of inferring a reward function given an agent's actions [1]. Concretely, given an MDP model $\mathcal{M} \setminus r$ and an expert's time-varying policy π^{E}, the goal is to find a reward function r such that π^{E} is the optimal policy for r, in other words r *induces* π^{E}. However, this problem is ill-posed because multiple distinct reward functions can yield the same optimal policy, making reward inference inherently ambiguous [5], [6], [7], [8]. In the case of the Max Entropy RL objective, the set of reward functions that induce a given policy π^{E} can be derived in closed form [6], [18]. We present the following result.

Lemma 1 ([6]): For any time-varying policy $\bar{\pi}_t(a|s) : \{0, \dots, T-1\} \times \mathcal{A} \times \mathcal{S} \to (0, 1]$, and for any function $\nu : \{0, \dots, T\} \times \mathcal{S} \to \mathbb{R}$, the reward function given by

$$r_t(s, a) = \log \bar{\pi}_t(a|s) - \gamma \mathbb{E}_{s'}[\nu_{t+1}(s')] + \nu_t(s), \quad (6)$$

with $\nu_T = 0$, is the only reward function for which $\bar{\pi}$ is the optimal solution of (3) with optimal soft value function $V^*_t = \nu_t$, for all t.

Similar to [18], we vectorize Equation (6) to define the set of rewards consistent with a given policy. To this end, we define the matrix $\Phi_T \in \mathbb{R}^{Tmn \times Tn}$ and the vector $\Xi^{\mathrm{E}} \in \mathbb{R}^{Tmn}$ as:

$$\Phi_T = \begin{bmatrix} -\mathbf{E} & \gamma\mathbf{P} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & -\mathbf{E} & \gamma\mathbf{P} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & -\mathbf{E} & \gamma\mathbf{P} \\ \mathbf{0} & \cdots & \cdots & \mathbf{0} & -\mathbf{E} \end{bmatrix}, \Xi^{\mathrm{E}} = \begin{bmatrix} \pi_0^{\log} \\ \pi_1^{\log} \\ \vdots \\ \pi_{T-1}^{\log} \end{bmatrix}, \quad (7)$$

with $\mathbf{E} = \mathbf{1}_m \otimes \mathbf{I}_n$, $\mathbf{P} = \begin{bmatrix} P^{\intercal}_{a^{(1)}} & \cdots & P^{\intercal}_{a^{(m)}} \end{bmatrix}^{\intercal} \in \mathbb{R}^{mn \times n}$, where $P_{a^{(k)}} \in \mathbb{R}^{n \times n}$ is such that its ij-th entry is given by $\mathcal{T}(s^{(j)}|s^{(i)}, a^{(k)})$, $k \in \{1, \dots, m\}$ and $\pi_t^{\log}$ is the vectorized policy given by:

$$\pi_t^{\log} = \begin{bmatrix} \log(\pi^E_t(a_1|s_1)) \\ \log(\pi^E_t(a_1|s_2)) \\ \vdots \\ \log(\pi^E_t(a_m|s_n)) \end{bmatrix} \in \mathbb{R}^{mn}, \quad t = 0, 1, \dots, T-1.$$

The subscript T in Φ_T emphasizes the number of block rows. Letting $r = [r_0^{\top}, \cdots, r_{T-1}^{\top}]^{\top}$ and $\nu = [\nu_0^{\top}, \cdots, \nu_{T-1}^{\top}]^{\top}$, the set of rewards and value functions inducing π^{E} can be compactly written as:

$$\mathcal{R}^{\mathrm{E}} = \left\{ \begin{bmatrix} r \\ \nu \end{bmatrix} \mid \begin{bmatrix} \mathbf{I}_{Tmn} \; \Phi_T \end{bmatrix} \begin{bmatrix} r \\ \nu \end{bmatrix} = \Xi^{\mathrm{E}} \right\}. \tag{8}$$

III. PROBLEM STATEMENTS

By looking at the definition of $\mathcal{R}^{\mathrm{E}}$, it should be clear from a simple dimension argument that there exists an infinite number of rewards inducing the same policy π^{E}. Indeed, the key insight from Lemma 1 is that for any expert policy π^{E}, one can generate a valid inducing reward r by selecting an appropriate time-dependent value function ν and computing r via Equation (6). Hence, recovering any reward function that induces π^{E} is not particularly meaningful. Instead, we are interested in recovering reward functions consistent with prior knowledge we might have about the structure of the reward function. One such prior is to find the time-varying reward function inducing π^{E} while having the minimum number of switches. This is particularly important in many real-world settings, where less erratic reward functions enhance interpretability and better reflect underlying task structures (Occam's razor). This gives rise to the first problem considered in this paper, formally given as:

Problem 1: Given an MDP model $\mathcal{M} \setminus r$ and a time-varying policy π^{E}, find the reward function r inducing π^{E} with the least number of switches, i.e. $r_t = r_{t+1}$ for as many t's as possible.

Another approach is to find a reward function that induces π^{E}, expressed in terms of a structured basis, commonly referred to in the IRL literature as *featurization* [13], [28], [29]. Specifically, the reward function at time t is given by:

$$r_t(s,a) = \sum_{k=1}^{K} \alpha_{k,t} u_k(s,a) = U\boldsymbol{\alpha_t}, \tag{9}$$

where $U \in \mathbb{R}^{mn \times K}$ is a feature matrix, with each column corresponding to a feature function u_k, and $\boldsymbol{\alpha_t} = [\alpha_{1,t}, \ldots, \alpha_{K,t}]^\intercal$ represents the corresponding feature weights. However, unlike standard IRL settings where the feature matrix U is typically predefined, in our case, U **is unknown**. This leads to the second problem we address:

Problem 2: Given an MDP model $\mathcal{M} \setminus r$ and an expert policy π^{E}, determine a candidate feature matrix U and weight vectors $\{\boldsymbol{\alpha_t}\}$ such that the reward function $r_t = U\boldsymbol{\alpha_t}$ induces π^{E}.

Both problems 1 and 2 can be extended to the setting when only a finite-sample estimate $\hat{\pi}^{\mathrm{E}}$ of the expert policy is available, rather than the true policy π^{E}. In Section V, we address this practical scenario by introducing robust variants of $\mathcal{R}^{\mathrm{E}}$ that accounts for estimation errors and provides probabilistic guarantees.

IV. METHODOLOGY

Throughout this section, our solutions will be different instantiations of the following optimization problem:

$$\begin{aligned} \min_{r,\nu} \quad & \ell(r) \\ \text{s.t.} \quad & \begin{bmatrix} r \\ \nu \end{bmatrix} \in \mathcal{R}^{\mathrm{E}}. \end{aligned} \tag{P}$$

for some loss function $\ell : \{r_0, \cdots, r_{T-1}\} \to \mathbb{R}$. Problem (P) serves as our unifying optimization framework, where domain-specific knowledge is systematically incorporated through tailored loss functions ℓ, while the constraint ensures consistency with the expert policy.

A. Minimally Switching Rewards

Problem 1 can be reformulated naturally as a *sparsification problem*, where the objective is to maximize the number of zero entries in an appropriately defined vector-valued sequence. In particular, we define the differences Δr_t between the rewards at every two consecutive time steps:

$$\Delta r_t = r_{t+1} - r_t, \; t = 0, \cdots, T-2, \tag{10}$$

and consider the sequence $\{\Delta r_t\}_{t=0}^{T-2}$. It should be clear that any non-zero element Δr_t corresponds to a switch in the reward function r. Hence, to minimize the number of switches, a natural optimization problem is the following:

$$\begin{aligned} \min_{r,\nu} \quad & \|\{\Delta r_t\}_{t=0}^{T-2}\|_0 \\ \text{s.t.} \quad & \begin{bmatrix} r \\ \nu \end{bmatrix} \in \mathcal{R}^{\mathrm{E}}, \\ & \Delta r_t = r_{t+1} - r_t, \; t = 0, \cdots, T-2, \end{aligned} \tag{P1}$$

where $\|\{\Delta r_t\}_{t=0}^{T-2}\|_0 \triangleq |\{t \mid \|\Delta r_t\| \neq 0\}|$. While maximizing sparsity is a non-convex and hard to solve problem in general [30], [31], there exist efficient convex relaxations based on variants of ℓ_1-norm [32]. Moreover, as we show next, thanks to the additional structure in $\mathcal{R}^{\mathrm{E}}$, problem (P1) admits an *exact* polynomial-time solution.

In what follows, we devise a greedy algorithm to solve problem (P1) and prove its correctness. To do so, we define the parametric truncated counterpart of Equation (8) with time-invariant rewards as:

$$\mathcal{R}^{\mathrm{inv}}_{i:j}(\nu_o) = \left\{ \begin{bmatrix} r \\ \nu \end{bmatrix} \mid \left[\bar{\mathbf{E}} \middle| \Phi_{j-i} \middle| \begin{matrix} \mathbf{0} \\ \gamma \mathbf{P} \end{matrix} \right] \begin{bmatrix} r \\ \nu \\ \nu_o \end{bmatrix} = \Xi^{\mathrm{E}}_{i:j} \right\} \tag{11}$$

where $\bar{\mathbf{E}} = \mathbf{1}_m \otimes \mathbf{I}_{mn}$ and $\Xi^{\mathrm{E}}_{i:j} = [\pi_i^{\log\intercal}, \ldots, \pi_{j-1}^{\log\intercal}]^\intercal \in \mathbb{R}^{(j-i)mn}$.

Our strategy is to find intervals over which a time-invariant reward can explain the given policy while ensuring consistency with the overall policy. In particular, we do this by working backward in time and iteratively extending an interval until a time-invariant reward is not feasible over this interval and starting a new interval from that point. Algorithm 1 implements this idea by following a bisection approach to find the time step where the time-invariant reward becomes infeasible.

Algorithm 1 Greedy Interval Partitioning

```
Input: Horizon: $T$, Transition Matrix: $\mathbf{P}$, Policy $\pi$
Output: Sequence of minimum number of switch times $Z$
    and corresponding reward functions $R$
1: $Z \leftarrow ()$, $R \leftarrow ()$, $V_t \leftarrow 0\ \forall t : 0 \leq t \leq T$
2: $l \leftarrow -1$, $u \leftarrow T$, $j \leftarrow T-1$, $\tau \leftarrow T$
3: while $j \geq 0$ do
4:     if $\mathcal{R}^{\text{inv}}_{j:\tau}(V_\tau) \neq \emptyset$ then
5:         $u \leftarrow j$,
6:         Pick $\bar{r}, \bar{\nu}$ from $\mathcal{R}^{\text{inv}}_{j:\tau}(V_\tau)$
7:     else
8:         $l \leftarrow j$
9:         if $u = l + 1$ then
10:            Prepend $u$ to $Z$ and $\bar{r}$ to $R$
11:            $V_t \leftarrow \bar{\nu}_t\ \ \forall t \in [u, \tau - 1]$
12:            $\tau \leftarrow u$, $l \leftarrow -1$
13:        end if
14:    end if
15:    $j \leftarrow \lfloor (l+u)/2 \rfloor$
16: end while
17: Prepend $\bar{r}$ to $R$ and $V_t \leftarrow \bar{\nu}_t\ \ \forall t \in [0, \tau-1]$
18: return $Z$, $R$
```

Theorem 1: Algorithm 1 returns an optimal solution to Problem (P1). Moreover, it has polynomial-time complexity.

Before proving the theorem, we explain some of the notation in the algorithm. By convention, the set $\mathcal{R}^{\text{inv}}_{-1:j}(\nu)$ is considered to be empty for all j and ν. Moreover, V_t denotes reward-to-go at time t, i.e., it is equal to ν_t. The variables l and u are auxiliary bounds used in the bisection procedure, representing the current lower and upper bounds for the switch time being searched. Finally, τ is the horizon of the current interval for which the switch time is being searched. The following two lemmas will be useful in proving this theorem.

Lemma 2: Let the switch times returned by Algorithm 1 be $Z = [t_k, t_{k-1}, \dots, t_1]$ with $t_i > t_{i+1}$ and define $t_0 = T$. There is not any feasible time-invariant reward function for interval $t \in [t_{i+1}-1, t_i-1]$ for any $i \geq 0$, i.e., $\mathcal{R}^{\text{inv}}_{t_{i+1}-1:t_i}(\nu_o)$ is empty for all $\nu_o \in \mathbb{R}^n$.

Proof: First, we note that per Lines 9 and 10 of the algorithm, u is added as the new switch time only after $\mathcal{R}^{\text{inv}}_{u-1:\tau}(V_\tau)$ is found empty. Then, for all i, $\mathcal{R}^{\text{inv}}_{t_{i+1}-1:t_i}(V_{t_i})$ is empty. Next, we will show that this is equivalent to $\mathcal{R}^{\text{inv}}_{t_{i+1}-1:t_i}(\nu_o)$ being empty for all $\nu_o \in \mathbb{R}^n$.

It is trivial that if $\mathcal{R}^{\text{inv}}_{t_{i+1}-1:t_i}(\nu_o)$ is empty for all $\nu_o \in \mathbb{R}^n$, $\mathcal{R}^{\text{inv}}_{t_{i+1}-1:t_i}(V_{t_i})$ is empty as well. For the other direction, assume there exists $\bar{\nu}_o$ such that $\mathcal{R}^{\text{inv}}_{t_{i+1}-1:t_i}(\bar{\nu}_o)$ is not empty and pick $(r, \nu) \in \mathcal{R}^{\text{inv}}_{t_{i+1}-1:t_i}(\bar{\nu}_o)$. Define r' as $r'_{T-1}(s,a) = r_{T-1}(s,a) + \gamma \mathbb{E}_{s' \sim P(.|s,a)}[\bar{\nu}_o(s') - V_{t_i}(s')]$ and $r'_t(s,a) = r_t(s,a)$ for all $t < T-1$ and for all a and s. Then, it can be seen by inspection that (r', ν) is in $\mathcal{R}^{\text{inv}}_{t_{i+1}-1:t_i}(V_{t_i})$. Hence, $\mathcal{R}^{\text{inv}}_{t_{i+1}-1:t_i}(V_{t_i})$ is not empty. ■

Lemma 3: Let the switch times returned by Algorithm 1 be $[t_k, t_{k-1}, \dots, t_1]$. For any feasible switch time sequence $[t'_{k'}, t'_{k'-1}, \dots, t'_1]$:

$$t_i \leq t'_i \qquad \forall i : 1 \leq i \leq \min(k, k')$$

Proof: We will show this by induction on i. At each step, we will use contradiction to show that t'_i cannot be less than t_i.

Base case ($i = 1$): Assume $t'_1 < t_1$; hence, $t'_1 \leq t_1 - 1$. Then, by Lemma 2, $[t'_1, T-1]$ cannot have a feasible time-invariant reward. Thus, t'_1 must be greater than or equal to t_1.

Induction Step: Assume that $t'_i \geq t_i$ and $t'_{i+1} < t_{i+1}$. Then, $[t_{i+1}-1, t_i-1] \subset [t'_{i+1}-1, t'_i-1]$. Hence, the time-invariant reward that is feasible for $[t'_{i+1}-1, t'_i-1]$ is feasible for $[t_{i+1}-1, t_i-1]$ as well. This contradicts Lemma 2. ■

Now we are ready to prove Theorem 1.

Proof: (of Theorem 1) Assume that Algorithm 1 returns a sequence of reward functions $R = [\bar{r}_{k+1}, \bar{r}_k, \dots, \bar{r}_1]$ and switch times $Z = [t_k, t_{k-1}, \dots, t_1]$. We start our proof by showing that the returned rewards form a feasible solution to Problem (P1).

Let ν be a reward-to-go function such that for all $1 \leq i \leq k+1$, $[\bar{r}_i, \nu_{t_i}, \nu_{t_i+1}, \dots, \nu_{t_{i-1}-1}]$ is in $\mathcal{R}^{\text{inv}}_{t_i:t_{i-1}}(\nu_{t_{i-1}})$, where t_0 and t_{k+1} are taken as T and 0, respectively. It can be seen by inspection that V populated on Line 11 of Algorithm 1 is such a ν. Now, define r_t to be $\bar{r}_i$ for all $t_i \leq t < t_{i-1}$ for all i. Then, $[r_0, \dots, r_{T-1}, \nu_0, \dots, \nu_{T-1}]$ is in $\mathcal{R}^{\text{E}}$. Hence, Algorithm 1 yields a feasible solution.

Now, by contradiction, assume that Algorithm 1 returns a feasible but suboptimal solution. Then, there exists a feasible solution with switch times $[t'_{k'}, t'_{k'-1}, \dots t'_1]$ with $k' < k$. If $[0, t'_{k'}]$ is a valid interval that can be solved with a time-invariant reward, $\mathcal{R}^{\text{inv}}_{0:t'_{k'}}(\nu_o)$ is not empty for some ν_o. However, by Lemma 2, $\mathcal{R}^{\text{inv}}_{t_{k'+1}-1:t_{k'}}(\nu_o)$ is empty for all ν_o. Since $[t_{k'+1}-1 : t_{k'}] \subseteq [0, t'_{k'}]$ by Lemma 3, this is a contradiction. Therefore, Algorithm 1 finds an optimal solution.

Finally, the runtime of the algorithm is primarily dominated by the operation in Line 6, which involves solving a system of equalities with $(m + \tau - j)n$ variables and $mn(\tau - j)$ constraints, which can be done in polynomial-time. This operation is performed $O(k \log T)$ times. Therefore, the overall algorithm runs in polynomial-time. ■

We also remark that when the reward function is known to be featurized—that is, all r_t can be expressed as weighted sums of common feature functions—the problem can be further simplified by restricting the search to the space of feature weights, as described in [18]. However, this requires the knowledge of the feature functions. In the next section, we show how this requirement can be avoided.

B. Feature-Based Rewards

If the reward function at time step t is expressed as $r_t = U\boldsymbol{\alpha_t}$, optimizing over both U and $\boldsymbol{\alpha_t}$ leads to a bilinear optimization problem. Further, since the number of features is unknown, the dimension of U is an additional decision variable. Bilinear programs are generally NP-hard due to their inherent non-convexity, and even checking local optimality can be computationally intractable [33], [34]. Our key insight to avoid solving a bilinear program is that featurization imposes a low-rank structure on the reward function, which remains consistent across the entire horizon. This means that while the reward at each time step may vary, it lies in a subspace spanned by a fixed set of basis functions. Consequently, instead of independently optimizing U and $\boldsymbol{\alpha_t}$, we can directly model the reward function as a low-rank matrix, where each column corresponds to the reward at a given time step. By enforcing a low-rank structure on this matrix, we transform the problem into one of recovering a structured representation of rewards rather than solving a bilinear optimization. Consequently, our objective is:

$$\min_{r,\nu} \quad \operatorname{rank}(\begin{bmatrix} r_0 \cdots r_{T-1} \end{bmatrix}) \quad \text{s.t.} \quad \begin{bmatrix} r \\ \nu \end{bmatrix} \in \mathcal{R}^{\mathrm{E}}. \tag{P2}$$

The feature matrix and weights can then be recovered from the optimal solution of (P2) as follows:

$$\begin{aligned} U &= \operatorname{colspan}(\begin{bmatrix} r_0 \cdots r_{T-1} \end{bmatrix}), \\ \boldsymbol{\alpha_t} &= [r_t]_U, \quad t = 0, \cdots, T-1. \end{aligned} \tag{12}$$

While Problem (P2) is still a difficult non-convex problem [35], several heuristics have been developed to handle it, e.g., see [36], [37]. Notably, it has been established in [38] that the nuclear norm, under some regularity assumptions, serves as the tightest convex approximation for the rank function, generalizing ℓ_1-norm based relaxation of the ℓ_o-quasinorm to the rank function. Thus, instead of Problem (P2), we solve:

$$\min_{r,\nu} \quad \| \begin{bmatrix} r_0 \cdots r_{T-1} \end{bmatrix} \|_* \quad \text{s.t.} \quad \begin{bmatrix} r \\ \nu \end{bmatrix} \in \mathcal{R}^{\mathrm{E}}. \tag{P2-approx}$$

where for a given matrix $A \in \mathbb{R}^{m \times n}$, its nuclear norm $\|A\|_* = \sum_{i=1}^{\min(m,n)} \sigma_i(A)$ with σ_i denoting the singular values of A. To get a better approximation of the rank function, nuclear norm relaxation can be further refined by considering an iterative reweighted variant [39]. While we tried this variant in our experiments, the results remained identical to those obtained with the nuclear norm formulation, which was sufficiently accurate for our problem.

V. Given Demonstrations

The set $\mathcal{R}^{\mathrm{E}}$ depends on the expert policy through Ξ^{E}. However, in practice, the true expert policy π^{E} is typically unknown and a finite-sample estimate $\hat{\pi}^{\mathrm{E}}$ must be used instead. We use the following lemma to motivate our approach in this case.

Lemma 4: Fix a timestep t, state $s \in \mathcal{S}$, and number of samples $n(t,s) \in \mathbb{N}$. Assume there exists a function $\alpha_t(s,\cdot) : \mathcal{A} \to \mathbb{R}$ such that $\alpha_t(s,a) \leq \pi_t^{\mathrm{E}}(a|s)$ for all $a \in \mathcal{A}$. Let $\{a_i\}_{i=1}^{n(t,s)}$ be a collection of actions independently drawn from the policy $\pi_t^{\mathrm{E}}(\cdot|s)$. Define the empirical estimate $\hat{\pi}_t^{\mathrm{E}}(a|s)$ as:

$$\hat{\pi}_t^{\mathrm{E}}(a|s) = \frac{1}{n(t,s)} \sum_{i=1}^{n(t,s)} \mathbb{I}[a_i = a]. \tag{13}$$

Given a confidence level $\delta \in (0,1)$, the following inequality holds for all actions $a \in \mathcal{A}$:

$$\mathbb{P}\left(\left| \log \hat{\pi}_t^{\mathrm{E}}(a|s) - \log \pi_t^{\mathrm{E}}(a|s) \right| \leq \frac{\epsilon(t,s)}{\alpha_t(s,a) - \epsilon(t,s)} \right) \geq \delta. \tag{14}$$

where:

$$\epsilon(t,s) \triangleq \sqrt{\frac{1}{2n(t,s)} \log\left(\frac{2}{1-\delta}\right)}. \tag{15}$$

Proof: Consider a fixed action a and let $\{X_1, X_2, \ldots, X_{n(t,s)}\}$ be random variables defined such that $X_i = \mathbb{I}[a_i = a]$ for all i. Note that $\hat{\pi}_t^{\mathrm{E}}(a|s) = \frac{1}{n(t,s)} \sum_{i=1}^{n(t,s)} X_i$ and $\mathbb{E}[\hat{\pi}_t^{\mathrm{E}}(a|s)] = \pi_t^{\mathrm{E}}(a|s)$. Then by Hoeffding's inequality, for any $\epsilon > 0$ we have:

$$\mathbb{P}\left(|\hat{\pi}_t^{\mathrm{E}}(a|s) - \pi_t^{\mathrm{E}}(a|s)| \leq \epsilon \right) \geq 1 - 2e^{-2n(t,s)\epsilon^2}.$$

Picking $\epsilon = \epsilon(t,s)$ as defined in Equation (15) yields:

$$\mathbb{P}\left(|\hat{\pi}_t^{\mathrm{E}}(a|s) - \pi_t^{\mathrm{E}}(a|s)| \leq \epsilon(t,s) \right) \geq \delta. \tag{16}$$

To go from Equation (16) to Equation (14), we need a lower bound for both $\hat{\pi}_t^{\mathrm{E}}(a|s)$ and $\pi_t^{\mathrm{E}}(a|s)$. To this end, define the events $\mathcal{E}_1 \triangleq \{|\hat{\pi}_t^{\mathrm{E}}(a|s) - \pi_t^{\mathrm{E}}(a|s)| \leq \epsilon(t,s)\}$ and $\mathcal{E}_2 \triangleq \{\min(\hat{\pi}_t^{\mathrm{E}}(a|s), \pi_t^{\mathrm{E}}(a|s))) \geq \alpha_t(s,a) - \epsilon(t,s)\}$.

Since $\alpha_t(s,a) \leq \pi_t^{\mathrm{E}}(a|s)$, we have $\mathcal{E}_1 \subseteq \mathcal{E}_2$. Thus

$$\mathbb{P}(\mathcal{E}_1, \mathcal{E}_2) = \mathbb{P}(\mathcal{E}_1) \geq \delta. \tag{17}$$

Define $\mathcal{E}_3 \triangleq \{|\log \hat{\pi}_t^{\mathrm{E}}(a|s) - \log \pi_t^{\mathrm{E}}(a|s)| \leq \frac{\epsilon(t,s)}{\alpha(t,s) - \epsilon(t,s)}\}$, which is the event of interest in Inequality (14). By Mean Value Theorem we have that for some $\xi \geq 0$ between $\hat{\pi}_t^{\mathrm{E}}(a|s)$ and $\pi_t^{\mathrm{E}}(a|s)$ the following holds:

$$\log \hat{\pi}_t^{\mathrm{E}}(a|s) - \log \pi_t^{\mathrm{E}}(a|s) = \frac{1}{\xi}(\hat{\pi}_t^{\mathrm{E}}(a|s) - \pi_t^{\mathrm{E}}(a|s))$$

This implies that:

$$|\log \hat{\pi}_t^{\mathrm{E}}(a|s) - \log \pi_t^{\mathrm{E}}(a|s)| \leq \frac{|\hat{\pi}_t^{\mathrm{E}}(a|s) - \pi_t^{\mathrm{E}}(a|s)|}{\min(\hat{\pi}_t^{\mathrm{E}}(a|s), \pi_t^{\mathrm{E}}(a|s)))},$$

from which we see that $\mathcal{E}_1 \cap \mathcal{E}_2 \subseteq \mathcal{E}_3$. Therefore, by Equation (17), we get $\mathbb{P}(\mathcal{E}_3) \geq \delta$, which is the desired result. ■

In practice, the data is given in the form of a set of trajectories $\mathfrak{D} = \{\tau_i\}_{i=1}^N$ with $\tau^i = (s_0^i, a_0^i, s_1^i, \ldots, a_{T-1}^i, s_T^i)$. For a given state s and time t, we define the number of samples as $n(t,s) \triangleq \sum_{i=1}^N \mathbb{I}[s_t^i = s]$ and set of actions sampled from $\pi_t(\cdot|s)$ as $\{a_t^i \mid s_t^i = s\}$. Note that $n(t,s)$ is independent of the sampled actions as actions sampled at time t only impacts future states. Then, we construct a sample consistent estimate of the true policy π^{E} by computing the relative frequency of actions at each state for each t from 0 to $T-1$ as shown in Equation (13).

Note that, while using maximum entropy policies guarantees that π_t is always positive, a lower bound function $\alpha_t(s,a)$ may not be known. In practice, we replace $\alpha_t(s,a) - \epsilon(t,s)$ in Equation (14) with $\hat{\pi}_t^{\mathrm{E}}(a|s) - \epsilon(t,s)$. When $n(t,s)$ is large enough, this gives a good lower bound to both $\hat{\pi}_t^{\mathrm{E}}(a|s)$ and $\pi_t^{\mathrm{E}}(a|s)$.

While increasing the number of samples reduces estimation error as can be seen in Equation (15), even small inaccuracies can disrupt important structural properties, such as sparsity or low-rank characteristics, inherent in the true reward function. To address this issue, we introduce a policy-noise-robust variant of $\mathcal{R}^{\mathrm{E}}$, ensuring that solutions remain feasible for at least one possible realization of Ξ^{E}. Specifically, given a set of trajectories $\mathfrak{D}$ and a confidence level δ, we estimate the policy using Equation (13) and obtain the estimated policy vector $\hat{\Xi}^{\mathrm{E}}$ as in Equation (7). We also construct an error bound vector $\mathbf{b}$ from the upper bound in Equation (14) as follows. Define the error vector $\boldsymbol{\epsilon} \triangleq [\boldsymbol{\epsilon}_0^\top, \boldsymbol{\epsilon}_1^\top, \dots, \boldsymbol{\epsilon}_{T-1}^\top]^\top \in \mathbb{R}^{Tmn}$, where for each $t \in \{0, \dots, T-1\}$, we have $\boldsymbol{\epsilon}_t \triangleq \mathbf{1}_m \otimes [\epsilon(t, s_1), \dots, \epsilon(t, s_n)]^\top$. Then, $\mathbf{b} \in \mathbb{R}_{\geq 0}^{Tmn}$ is given by $\mathbf{b} = \boldsymbol{\epsilon} \oslash \left(\exp\left(\hat{\boldsymbol{\Xi}}^{\mathrm{E}}\right) - \boldsymbol{\epsilon}\right)$, where $\exp(\cdot)$ and $\oslash$ denote elementwise exponentiation and division, respectively.

Finally, we define the robust reward set:

$$\hat{\mathcal{R}}^{\mathrm{E}} = \left\{ \begin{bmatrix} r \\ \nu \end{bmatrix} \mid \hat{\Xi}^{\mathrm{E}} - \mathbf{b} \leq \begin{bmatrix} \mathbf{I}_{Tmn} & \Phi_T \end{bmatrix} \begin{bmatrix} r \\ \nu \end{bmatrix} \leq \hat{\Xi}^{\mathrm{E}} + \mathbf{b} \right\}. \tag{18}$$

In our numerical experiments when we only have access to demonstrations, we replace $\mathcal{R}^{\mathrm{E}}$ with $\hat{\mathcal{R}}^{\mathrm{E}}$ in the corresponding optimization problems.

VI. EXPERIMENTS

The goal of our experiments is to answer two questions [1]:

Q1. How well can our frameworks recover the ground-truth time-varying rewards?

Q2. Can our recovered rewards transfer to novel environments?

To answer Q1, we evaluate Problems (P1) and (P2-approx) in the 5x5 gridworld shown in Figure 1a. Each cell of the gridworld represents a state of the MDP. The actions available for the agent in each state are: $\{up, down, left, right, stay\}$. Upon taking an action, the agent transitions to the desired cell with a probability $1-p_w$, and transitions to a neighboring cell in one of the cardinal directions with a probability p_w, representing the wind probability. The MDP has two important landmarks: a home state (called s_{home}) at the top left, and a water state (called s_{water}) in the bottom middle. For example, an agent with high reward at the home state tries to reach the home as fast as possible. An agent with a uniform reward everywhere tries to explore the environment equally. By varying the rewards over time, we can capture and model a multitude of complex behaviors. For example, the agent might want to explore the environment at first. During exploration it gets "thirsty", and thus the reward switches to reach the water state as fast as possible. Eventually, the agent wants to go back to the home state. A horizon of 50 timesteps is used in all our experiments. Our frameworks improve over other baselines in qualitatively recovering the ground-truth weights and feature functions.

To answer Q2, we evaluate our frameworks in a transfer learning setting, where the reward function is learned in

[1]The code for these experiments is available at https://github.com/mlshehab/sparse_reward_identification.

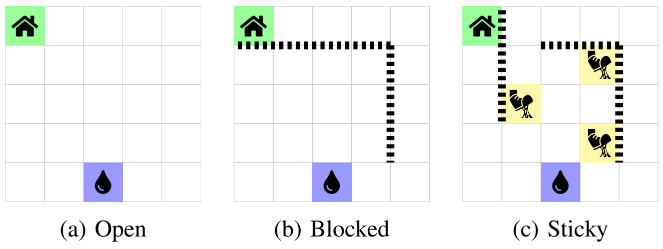
(a) Open (b) Blocked (c) Sticky

Fig. 1: 5x5 Gridworlds used in the experiments.

the gridworld of Figure 1a, but optimized in a different gridworld with different dynamics, shown in Figures 1b and 1c. The difference in the dynamics is characterized by adding blockings, shown as a dashed line, and adding "sticky" states, shown in yellow, where all actions result in staying in that state with a probability 0.8. We show that rewards learned with our algorithms still produce optimal or near-optimal behaviors, while baseline methods produce either lower quality policies or rewards that generalize poorly.

A. Experiment 1: Minimally Switching Rewards

In this experiment, we evaluate the performance of Algorithm 1. We construct tasks by dividing the time horizon into $k+1$ intervals, where $k=5$ switch points are chosen uniformly at random. These switch points correspond to time steps where $\Delta r_t \neq 0$. We begin with a time-invariant reward function, with values sampled uniformly between 0 and 1, for the first interval. For each interval, we generate a time-invariant reward function by perturbing the previous interval's reward using a uniformly sampled perturbation from $[0, \beta]^{mn}$. We vary β from 0.1 to 0.4 for each subsequent interval to induce reward changes of increasing magnitudes.

To evaluate the accuracy of the inferred switch times, we consider the resulting intervals as clusters. The predicted intervals are evaluated against the true interval partitioning using the Adjusted Rand Index (ARI), as defined in Equation 5 of [40]. The ARI is a widely used measure for comparing two clusterings: it equals 1 when they agree perfectly and has an expected value of 0 under random labeling (with possible negative values if agreement is worse than chance).

To study the impact of using a finite-sample estimate of the policy π^{E}, we compute the ARI scores of the switch times identified by Algorithm 1 when run on estimates of π^{E} obtained from varying numbers of trajectories. The experiment is repeated with 10 different reward functions. We report the mean and standard deviation of the ARI scores for each setting in Table I. As seen in Table I, increasing the number of trajectories leads to higher ARI scores and reduced variance. Our algorithm eventually recovers the true switching times. It is important to observe that the number of switches our algorithm identifies is always less than or equal to the true number of switches, hence our algorithm is able to explain the data with a simpler reward model when there is more uncertainty in the low-data regime.

B. Experiment 2: Feature-Based Rewards

For this experiment, we generated two ground-truth feature functions $u_1, u_2 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ representing the home state

Number of Trajectories	ARI	# of Switches
True Policy	1.000 ± 0.000	5.0 ± 0.0
8,000,000	1.000 ± 0.000	5.0 ± 0.0
4,000,000	0.968 ± 0.040	4.8 ± 0.4
2,000,000	0.881 ± 0.118	4.1 ± 0.3
1,000,000	0.726 ± 0.100	3.5 ± 0.5
800,000	0.674 ± 0.179	3.0 ± 0.45
400,000	0.566 ± 0.196	2.0 ± 0.63
200,000	0.393 ± 0.098	1.1 ± 0.3

TABLE I: Mean and standard deviation of ARI and the number of switches found across different dataset sizes over 10 random reward functions. The row labeled "True Policy" corresponds to results obtained using the exact policy π^{E} instead of trajectory data. Confidence level δ is set to 0.9999 when defining the robust reward set.

and water state positions. The reward function is given by:

$$r_t^{\text{true}}(s,a) = \alpha_{1,t} u_1(s,a) + \alpha_{2,t} u_2(s,a)$$

where $u_1(s,.) = 1$ if $s = s_{\text{home}}$, and 0 otherwise. Similarly, $u_2(s,.) = 1$ if $s = s_{\text{water}}$, and 0 otherwise. We generate the time-varying weights $\boldsymbol{\alpha_t}$ following a Gaussian random walk as in [19]. After finding the expert policy π^{E}, we solve (P2-approx) to find both the feature functions and the weights. The recovered time-varying weights are shown in Figure 2, which also includes results with policies esimated from demonstrations. The recovered feature functions[2] are shown in Figures 3 and 4 . As a benchmark, we implemented the dynamic IRL method from [19]. We note that the number of demonstrations fed to the method from [19] is much fewer than what our method used due to scalability issues with the former (yet we used 5 times the number of trajectories reported in [19]). Since the reward decomposition in (12) is not unique, there exists infinitely many valid choices of basis vectors U, leading to different recovered parameters for each basis choice. Thus, to qualitatively compare the ground-truth and recovered weights, and generate meaningful visualizations, we apply a two-step post-processing approach. First, we identify a basis U that satisfies the condition in (12). We then perform a projection step to align this basis with the ground-truth feature vectors, yielding a transformed basis U'. Next, we express the recovered weights relative to B' and apply a standardization step to eliminate trivial invariances due to shifting and scaling. This ensures that the recovered weights are comparable to the ground-truth while preserving their relative structure. Figures 3 and 4 compare our recovered feature mapping with that of [19]. By enforcing a low-rank decomposition in the objective function, our method successfully recovers the true feature functions, whereas dynamic IRL [19] produces a reasonable but noisier approximation.

For our transferability experiments, we implemented an additional baseline: the finite-horizon MaxEnt IRL of [16], which recovers a time-invariant reward. To assess transferability, we first solve for the reward function using different methods/baselines with the optimal ground-truth policy in the

[2]Since the feature vectors are $|\mathcal{S}| \times |\mathcal{A}|$ dimensional vectors, we only show the first $|\mathcal{S}|$ components, which correspond to the first action. The plots are the same for the remaining actions.

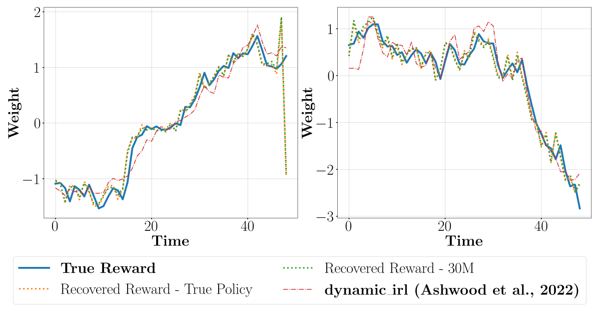

Fig. 2: Recovered weights for Experiment 2.

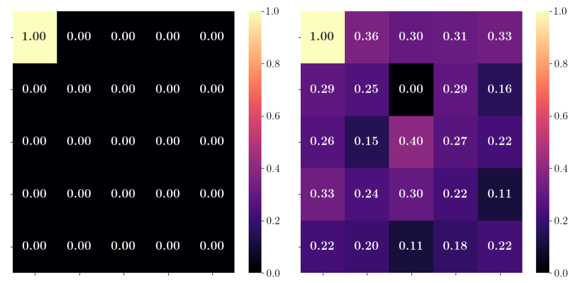

Fig. 3: Recovered feature vector for the home state: (left) Ours vs. (right) [19].

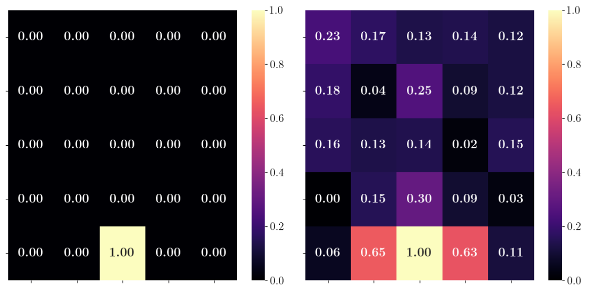

Fig. 4: Recovered feature vector for the water state: (left) Ours vs. (right) [19].

Open Gridworld as input. We then compute the optimal policies of these recovered rewards in the novel environments, namely the Blocked Gridworld and the Sticky Gridworld. We report the negative log-likelihood of a sample trajectory set, generated from the optimal policy for the ground-truth reward, for each of the computed policies. We report these log-likelihoods in Table II. Our algorithm achieves near-optimal performance in both novel environments, attaining the best transferability performance among all methods. It is worth mentioning that the Gaussian random walk structure of the weights is embedded in the learning algorithm of [19], which we do not assume in our approach. Also, finite-horizon MaxEnt IRL baseline is given the true feature function. Finally, both [19] and [16] produce state-only reward functions, which are usually more suited for transferability tasks [15].

VII. CONCLUSION

In this work, we addressed the challenge of reward identification in finite-horizon, time-varying settings by introducing a unifying framework that incorporates sparsity and rank priors. Our approach efficiently recovers minimally switching

Policy	Blocked Gridworld	Sticky Gridworld
π_1^*	-1.2851	-1.3021
$\hat{\pi}_1^*$	-1.2850	-1.3019
π_{true}^*	$\mathbf{-1.2495}$	$\mathbf{-1.2223}$
π_r^* (Ours)	$\mathbf{-1.2516}$	$\mathbf{-1.2438}$
π_{ash}^* [19]	-1.3546	-1.2778
π_{s}^* [16]	-1.3236	-1.3279

TABLE II: Performance of different policies in the transferability experiments. π_1^* is the optimal policy of r^{true} in the Open GridWorld, and $\hat{\pi}_1^*$ is its sample estimate. In the novel environments, π_{true}^* is the optimal policy of r^{true}, π_r^* is the optimal policy of the learned reward from (P2-approx), π_{ash}^* is the optimal policy using [19], and π_{s}^* is the optimal policy using [16].

rewards through a greedy interval partitioning algorithm and leverages low-rank matrix approximations to identify structured feature-based rewards. Empirical results on several gridworld environments demonstrate robustness to policy estimation noise and superior transferability compared to existing methods.

References

[1] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *International Conference on Machine Learning*, vol. 1, 2000, p. 2.

[2] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 66–73.

[3] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Proceedings of Robotics: Science and Systems (RSS)*, July 2017.

[4] E. Bıyık, D. P. Losey, M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning reward functions from diverse sources of human feedback: Optimally integrating demonstrations and preferences," *The International Journal of Robotics Research*, vol. 41, no. 1, pp. 45–67, 2022.

[5] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *International Conference on Machine Learning*, 1999, pp. 278–287.

[6] H. Cao, S. Cohen, and L. Szpruch, "Identifiability in inverse reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 362–12 373, 2021.

[7] K. Kim, S. Garg, K. Shiragur, and S. Ermon, "Reward identification in inverse reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5496–5505.

[8] J. M. V. Skalse, M. Farrugia-Roberts, S. Russell, A. Abate, and A. Gleave, "Invariance in policy optimisation and partial identifiability in reward learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 32 033–32 058.

[9] A. M. Metelli, G. Ramponi, A. Concetti, and M. Restelli, "Provably efficient learning of transferable rewards," in *International Conference on Machine Learning*. PMLR, 2021, pp. 7665–7676.

[10] A. M. Metelli, F. Lazzati, and M. Restelli, "Towards theoretical understanding of inverse reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 24 555–24 591.

[11] D. Lindner, A. Krause, and G. Ramponi, "Active exploration for inverse reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 5843–5853, 2022.

[12] A. M. Metelli, "Recent advancements in inverse reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 20, 2024, pp. 22 680–22 680.

[13] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *International Conference on Machine Learning*, 2004, p. 1.

[14] J. M. Buchanan, *Cost and choice: An inquiry in economic theory*. University of Chicago Press, 1978.

[15] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," in *International Conference on Learning Representations*, 2018.

[16] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *AAAI Conference on Artificial Intelligence*, vol. 8, 2008, pp. 1433–1438.

[17] F. Lazzati and A. M. Metelli, "On the partial identifiability in reward learning: Choosing the best reward," *arXiv preprint arXiv:2501.06376*, 2025.

[18] M. L. Shehab, A. Aspeel, N. Arechiga, A. Best, and N. Ozay, "Learning true objectives: Linear algebraic characterizations of identifiability in inverse reinforcement learning," in *6th Annual Learning for Dynamics & Control Conference*. PMLR, 2024, pp. 1266–1277.

[19] Z. Ashwood, A. Jha, and J. W. Pillow, "Dynamic inverse reinforcement learning for characterizing animal behavior," *Advances in Neural Information Processing Systems*, vol. 35, pp. 29 663–29 676, 2022.

[20] Q. P. Nguyen, B. K. H. Low, and P. Jaillet, "Inverse reinforcement learning with locally consistent reward functions," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[21] A. Likmeta, A. M. Metelli, G. Ramponi, A. Tirinzoni, M. Giuliani, and M. Restelli, "Dealing with multiple experts and non-stationarity in inverse reinforcement learning: an application to real-life problems," *Machine Learning*, vol. 110, pp. 2541–2576, 2021.

[22] R. Rickenbach, E. Arcari, and M. Zeilinger, "Time dependent inverse optimal control using trigonometric basis functions," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 1193–1204.

[23] K. Westermann, J. F.-S. Lin, and D. Kulić, "Inverse optimal control with time-varying objectives: application to human jumping movement analysis," *Scientific reports*, vol. 10, no. 1, p. 11174, 2020.

[24] M. Geist, B. Scherrer, and O. Pietquin, "A theory of regularized markov decision processes," in *International Conference on Machine Learning*, 2019, pp. 2160–2169.

[25] B. D. Ziebart, *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

[26] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1352–1361.

[27] A. Gleave and S. Toyer, "A primer on maximum causal entropy inverse reinforcement learning," *arXiv preprint arXiv:2203.11409*, 2022.

[28] F. Memarian, Z. Xu, B. Wu, M. Wen, and U. Topcu, "Active task-inference-guided deep inverse reinforcement learning," in *IEEE Conference on Decision and Control*. IEEE, 2020, pp. 1932–1938.

[29] M. Bloem and N. Bambos, "Infinite time horizon maximum causal entropy inverse reinforcement learning," in *IEEE Conference on Decision and Control*. IEEE, 2014, pp. 4911–4916.

[30] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, vol. 209, no. 1-2, pp. 237–260, 1998.

[31] G. Davis, S. Mallat, and M. Avellaneda, "Adaptive greedy approximations," *Constructive Approximation*, vol. 13, pp. 57–98, 1997.

[32] N. Ozay, M. Sznaier, C. M. Lagoa, and O. I. Camps, "A sparsification approach to set membership identification of switched affine systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 3, pp. 634–648, 2011.

[33] H. Konno, "A cutting plane algorithm for solving bilinear programs," *Mathematical Programming*, vol. 11, no. 1, pp. 14–27, 1976.

[34] F. A. Al-Khayyal and J. E. Falk, "Jointly constrained biconvex programming," *Mathematics of Operations Research*, vol. 8, no. 2, pp. 273–286, 1983.

[35] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[36] E. Beran and K. Grigoriadis, "A combined alternating projections and semidefinite programming algorithm for low-order control design," *IFAC Proceedings Volumes*, vol. 29, no. 1, pp. 1068–1073, 1996.

[37] R. E. Skelton, T. Iwasaki, and K. Grigoriadis, *A unified algebraic approach to linear control design*. Taylor & Francis, 2013.

[38] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.

[39] K. Mohan and M. Fazel, "Reweighted nuclear norm minimization with application to system identification," in *American Control Conference*. IEEE, 2010, pp. 2953–2959.

[40] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, pp. 193–218, 1985.