Efficient Cooperative Inverse Reinforcement Learning

Malayandi Palaniappan [*1] **Dhruv Malik** [*1] **Dylan Hadfield-Menell** [1] **Anca Dragan** [1] **Stuart Russell** [1]

1. Introduction

For an autonomous system to provide value to humans without endangering them, it must be able to accurately and reliably identify humans' intended objective, and act accordingly to maximize humans' gain. This *value alignment* problem is difficult given that humans are notoriously prone to misstating their objectives.

One formalization of the value alignment problem is the Cooperative Inverse Reinforcement Learning (CIRL) framework, which formulates the problem as a two-player, asymmetric information game between a human and a robot (Hadfield-Menell et al., 2016).

Previous work showed that the problem can be reduced into a particular type of partially observable Markov decision process (POMDP), known as a Coordinator-POMDP, but observed that the formulation was still not tractable. The large size of the action space in the Coordinator-POMDP makes it ill suited for POMDP algorithms, whose complexity largely depends on the size of the action space. We verify this experimentally – exact POMDP value iteration fails to solve all but the simplest CIRL problems.

We derive a more efficient exact algorithm for solving CIRL games by introducing a modification to the POMDP value iteration update rule that significantly reduces the size of the action space in the problem. This update naturally extends itself to existing approximate POMDP planning algorithms. Finally, we introduce a benchmark CIRL game and test our algorithms experimentally, verifying that they outperform their exact and approximate counterparts.

2. Background

2.1. Running Example

Consider a cooking task, which we refer to as *ChefWorld*, where a robot seeks to prepare a meal for a human. The robot has m units each of n ingredients, which may be used to prepare the meal. At any time step, the robot may choose to prepare a single unit of any ingredient or to not prepare any ingredient at all. The robot receives a reward of 1 if it succeeds in preparing the meal and a reward of 0 otherwise. We introduce definitions in the context of this example.

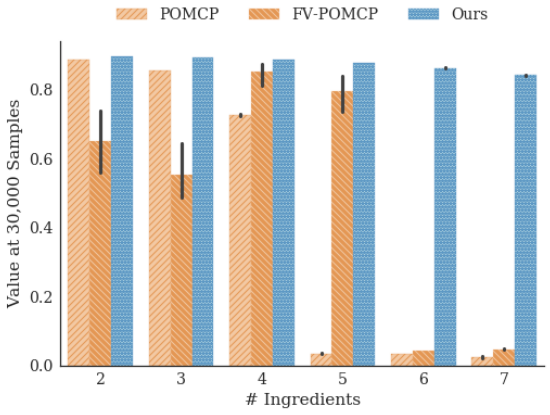

Figure 1. The value attained by POMCP, FV-POMCP, and our approximate algorithm in 30,000 samples on the *ChefWorld* domain with various numbers of ingredients.

2.2. POMDPs

The POMDP framework provides a rich model for planning under uncertainty (Sondik, 1971; Kaelbling et al., 1998). Formally, a POMDP is defined as a tuple $\langle S, A, Z, T, O, r, \gamma \rangle$ where S is the set of states; A is the set of actions available to the agent; Z is the set of observations; T is the transition distribution which specifies the probability of transitioning to some state given the current state and action to be taken; O is the observation distribution which specifies the probability of receiving some observation given the current state and last action taken; r is the reward function; and γ is the discount factor.

In a POMDP, the agent cannot observe the state; instead, it receives an observation at every time step, which helps inform its decision making. The behavior of the agent is given by a conditional plan $\sigma = (a, v)$ where a denotes the agent's action, and v is a mapping from observations to future conditional plans for the agent to follow.

[*]Equal contribution [1]Department of Electrical Engineering and Computer Sciences, UC Berkeley, CA, United States. Correspondence to: Malayandi Palaniappan <malayandi@berkeley.edu>, Dhruv Malik <dhruvmalik@berkeley.edu>.

Formally, a policy in a POMDP is a mapping from action-observation histories to conditional plans. However, (Sondik, 1971) showed that the belief $b \in \Delta S$, where $b(s)$ is the probability that the agent is in state s given her history of actions and observations, is a sufficient statistic in a POMDP i.e. policies in a POMDP need only depend on this belief instead of entire action-observation histories.

We can frame our cooking task as a POMDP. Consider a robot who wants to prepare a meal for a human, but does not know whether the human prefers a sandwich or soup; instead, it maintains a belief, which describes how much it thinks that the human prefers either of the two meals. The robot has access to the following ingredients: bread, tomato and meat. At each time step, the robot may choose to prepare any of the three ingredients. The human then responds by preparing an ingredient herself according to some fixed policy; the robot observes which ingredient the human prepares. An example of a conditional plan for the robot would be as follows: prepare meat now and if the human responds by preparing bread, prepare a second slice of bread to make the sandwich; if the human prepares tomatoes, prepare another batch of tomatoes to make soup; or if the human prepares meat, do not prepare any ingredient.

The α-vector of a conditional plan is defined as the vector containing the value of following the plan at any given state, which is given by

$$\alpha_\sigma(s) = R(s) + \gamma \sum_{s' \in S} \sum_{z \in Z} P(s', z \mid s, a) \alpha_{v(z)}(s') \quad (1)$$

The value of a plan at a belief b is defined as the expected value of the plan across the states i.e. $V_\sigma(b) = b \cdot \alpha_\sigma = \sum_{s \in S} b(s)\alpha_\sigma(s)$. The goal of an agent in a POMDP is to find the plan with maximal value from her current belief.

The value iteration algorithm can be used to exactly compute the optimal conditional plan for an agent in a POMDP with finite horizon T (Sondik, 1971). The algorithm functions by rolling back values from the horizon, generating new conditional plans at each time-step and evaluating them in terms of future plans using Eqn. (1). The algorithm evaluates all plans of length T and selects the one with maximal value at the agent's initial belief.

2.3. Cooperative Inverse Reinforcement Learning

Cooperative Inverse Reinforcement Learning (CIRL) formalizes the value-alignment problem as a two-player, asymmetric information game between a human and a robot, both of whose goals are to maximize their expected reward (Hadfield-Menell et al., 2016). Crucially, the robot shares the human's reward function, which is parameterized by a reward parameter θ known only to the human. This structure incentivizes the robot to learn the human's true preference and to maximize reward *for* the human, hence addressing the value alignment problem.

Formally, a CIRL game is a multi-agent game between a human $\mathbf{H}$ and a robot $\mathbf{R}$ denoted by the tuple $M = \langle X, \{\mathcal{A}^H, \mathcal{A}^R\}, T, \{\Theta, r\}, P_0, \gamma \rangle$ where X is the set of observable world-states; $\mathcal{A}^H$ and $\mathcal{A}^R$ are the actions available to $\mathbf{H}$ and $\mathbf{R}$ respectively; T is the transition distribution specifying the probability of transitioning to some state given the current state and actions of both agents; Θ is the set of possible reward parameters; r is the reward function shared by both agents; P_0 is the initial distribution over states and reward parameters; and γ is the discount factor.

By adapting *ChefWorld* to include a human $\mathbf{H}$ and a robot $\mathbf{R}$ who collaborate to make a meal, we can frame the problem as a CIRL game. Say $\mathbf{H}$ wants to prepare either a sandwich or soup with the help of $\mathbf{R}$ who does not know which meal the human has chosen. At each time step, both agents prepare an ingredient and $\mathbf{R}$ observes which ingredient $\mathbf{H}$ prepared. Both agents receive a reward of 1 if they successfully prepare the correct meal, and 0 otherwise.

This is rather distinct from the POMDP formulation of *ChefWorld* where the human behaved with a fixed policy that did not take into account the robot's future behavior. In this case, there is a strong interdependence between $\mathbf{H}$'s actions and the robot's future behavior. In order to maximize her reward, $\mathbf{H}$ will choose to prepare that ingredient which conveys the most information about her preferred meal to the robot. Her actions influence $\mathbf{R}$'s belief over her preferred meal, which in turn influence both $\mathbf{R}$'s and her future actions.

As a multi-agent game, a CIRL game technically belongs to the class of Dec-POMDPs (Bernstein et al., 2002). However, the problem of finding an optimal pair of strategies in a CIRL game can be reduced to solving a Coordinator-POMDP. The states in the coordinator-POMDP are tuples containing the world-state and reward parameter i.e. $S = X \times \Theta$; the actions are tuples (δ^H, a^R) specifying for $\mathbf{H}$ a decision rule $\delta^H : \Theta \to \mathcal{A}^H$, which maps reward parameters to human actions, and for $\mathbf{R}$ an action a^R; the observations are $\mathbf{H}$'s action at the last time step; the remaining parameters of the Coordinator-POMDP are inherited from the CIRL game (Hadfield-Menell et al., 2016).

In our cooking task, an example of an action in the Coordinator-POMDP is a tuple, where the first entry specifies that $\mathbf{H}$ prepares bread if she prefers a sandwich and prepares tomatoes if she prefers soup, and the second entry specifies that the robot prepares bread regardless.

3. A Modified Bellman Update for CIRL

Formulating the CIRL game as a Coordinator-POMDP allows us to find an optimal policy for $\mathbf{R}$ using standard

POMDP algorithms. However, this problem is still very difficult. The action space in the Coordinator POMDP has size $|\mathcal{A}^H|^{|\Theta|}|\mathcal{A}^R|$ and thus POMDP value iteration, which scales poorly with the size of the action space (Russell et al., 1995), can only feasibly solve the most straightforward CIRL problems.

If $\mathbf{H}$ were following a fixed policy based on the augmented state (x, θ), we could reduce the complexity of the problem significantly by accounting for $\mathbf{H}$'s behavior implicitly in the transition-observation distribution as opposed to explicitly doing so in the action space.

However, in the interactive setting of a CIRL game, $\mathbf{H}$ may plan for any changes in $\mathbf{R}$'s belief and so, if we included $\mathbf{H}$'s behavior in the transition-observation distribution, the distribution would change at each time step based on $\mathbf{R}$'s belief, or, more accurately, $\mathbf{R}'s$ intended response to $\mathbf{H}$'s action.

Our key insight is that, during planning, we have access to $\mathbf{R}$'s intended future response to each of $\mathbf{H}$'s action and hence have enough information to compute $\mathbf{H}$'s action. We can therefore compute the optimal robot policy in a CIRL game by solving a POMDP with time-varying dynamics, where the action space has size $|\mathcal{A}^R|$ as opposed to $|\mathcal{A}^H|^{|\Theta|}|\mathcal{A}^R|$. In our *ChefWorld* domain, where n denotes the number of ingredients and k denotes the number of recipes, our approach reduces the size of the action space from $(n+1)^{k+1}$ to simply $n+1$.

3.1. The Transition Dynamics

If $\mathbf{H}$ were to follow some policy that depends only on the state $s = (x, \theta)$, the joint transition-observation dynamics of the Coordinator-POMDP can be computed according to

$$\begin{aligned} &P(s', a^H \mid s, a^R) = P((x', \theta'), a^H \mid (x, \theta), a^R) \\ &= P((x', \theta') \mid (x, \theta), a^H, a^R) \cdot P(a^H \mid (x, \theta), a^R) \\ &= T(x, a^H, a^R, x') \cdot \mathbb{1}(\theta = \theta') \cdot P(a^H \mid x, a^R, \theta) \end{aligned}$$

However, in reality, $\mathbf{H}$ behaves according to her Q-values, picking the action that maximizes her expected value, and not according to some fixed policy. Due to the interdependence between $\mathbf{H}$'s and $\mathbf{R}$'s behavior, these Q-values necessarily depend on $\mathbf{R}$'s conditional plan. The joint transition-observation dynamics of the game are then:

$$\begin{aligned} &P(s', a^H \mid s, \sigma) = P((x', \theta'), a^H \mid (x, \theta), (a^R, v)) \\ &= T(x, a^H, a^R, x') \cdot \mathbb{1}(\theta' = \theta) \cdot P(a^H \mid x, a^R, v, \theta) \\ &= T(x, a^H, a^R, x') \cdot \mathbb{1}(\theta' = \theta) \cdot \\ &\qquad \mathbb{1}(a^H = \arg\max_{a^H} Q(x, a^H, a^R, v, \theta)) \end{aligned} \tag{2}$$

These dynamics are not static since they depend on the robot's future actions, and hence our problem is no longer a POMDP, which requires that the transition-observation dynamics be static. However, we can still solve the problem exactly by adapting POMDP value iteration.

3.2. Adapting POMDP Value Iteration

Due to the fact that POMDP value iteration functions by rolling back the values of the game from the horizon in the form of α-vectors, for a given augmented-state and robot-action pair, we can easily compute the human's Q-values as an expectation over the future alpha vectors: $Q(x, a^H, a^R, v, \theta) = \sum_{s'} T(x, a^H, a^R, x') \cdot \alpha_{v(a^H)}(s')$

We can thus adapt the backup rule used to compute the α-vectors for conditional plans, given in Eqn. (1), by replacing the joint transition-observation probability with $P(s', a^H \mid s, \sigma)$ as in Eqn. (2). With the sole exception of this small change, we can follow the POMDP value iteration algorithm identically to compute the optimal robot strategy in the CIRL game while reasoning over an action space with size $|\mathcal{A}^R|$ instead of $|\mathcal{A}^H|^{|\Theta|}|\mathcal{A}^R|$. The pseudocode for our algorithm is presented below.

Algorithm 1 Exact Value Iteration for CIRL games

```
Γ ← set of height 1 plans (i.e. actions)
for t in {1, 2, ..., T} do
    Γ' ← Γ
    Γ ← set of all plans consisting of an action and a
    map from observations to plans in Γ'
    for σ = (a^R, v) ∈ Γ do
        for s ∈ S do
            for a^H ∈ A^H do
                α_{v(a^H)}(s) = R(s) + γ·
                    Σ_{s'} Σ_{a^H} P(s', a^H | s, σ) α_{v(a^H)}(s')
                where P(s', a^H | s, σ) is as in Eqn. 2
            end for
        end for
    end for
    Γ ← Prune(Γ)
end for
Return Γ
```

If $|\Gamma_{t+1}|$ denotes the number of plans beginning at time $t+1$, one step of POMDP value iteration runs in $\mathcal{O}\big(|S|^2|Z||A||\Gamma_{t+1}|^{|Z|}\big)$ time, generating and evaluating $|A||\Gamma_{t+1}|^{|Z|}$ new plans which each begin at time t[1].

The action space in the Coordinator-POMDP has size $|\mathcal{A}^H|^{|\Theta|}|\mathcal{A}^R|$. In our adapted approach, the action space has size $|\mathcal{A}^R|$. Therefore, our algorithm reduces the number of plans generated at, and the time taken to run, each

[1]"Pruning" methods which reduce the size of the set of conditional plans at each time step often improve performance, but the exponential nature of the algorithm's growth remains unhindered (Monahan, 1982; Cassandra et al., 1997).

Table 1. Time taken (in seconds) to find the optimal robot policy using exact value iteration and our exact algorithm on the *ChefWorld* domain with various numbers of possible recipes. NA denotes that the algorithm depleted the memory in our system.

# RECIPES	EXACT VI	OURS
2	4.448 ± 0.057	0.071 ± 0.013
3	394.546 ± 6.396	0.111 ± 0.013
4	NA	0.158 ± 0.003
5	NA	0.219 ± 0.007
6	NA	0.307 ± 0.005

time step by a factor of $|\mathcal{A}^H|^{\Theta}$.

Given that this exponential explosion of the number of plans occurs at every time step, it is clear that our algorithm offers a *significant* improvement to value iteration applied on the Coordinator-POMDP, in terms of both time and space complexity.

4. Adapting POMCP

POMCP is a state-of-the-art approximate algorithm used to solve large POMDPs (Silver & Veness, 2010). The algorithm incrementally constructs a search tree of action-observation histories, using Monte-Carlo simulations to estimate the value of each history node in the tree. At each time step, an action is selected and an observation is then sampled, determining the current node in the search tree. We repeat this simulation process until we reach a leaf node, at which point we use a rollout policy to gather reward, and then back this reward up the tree.

We use our updated backup rule to modify POMCP such that it may solve our reduced problem with only $|\mathcal{A}^R|$ actions. The key idea is to approximately compute $\mathbf{H}$'s policy while running the algorithm. We do so by maintaining a live estimate of the sampled Q-values for $\mathbf{H}$ at each node. In the limit of infinite samples, these Q values converge, allowing us to find a policy for $\mathbf{H}$ that is the best response to $\mathbf{R}$'s policy.

While POMCP scales well with the size of the state space, it does not scale well with the size of the action and observation space, which determine the branching factor in the search tree. The branching factor for the search tree in POMCP when applied on the Coordinator-POMDP is $|\mathcal{A}^H|^{|\Theta|}|\mathcal{A}^R|$ and thus POMCP struggles to optimally solve even reasonably complex CIRL games.

This same problem plagues Factored-Value POMCP (FV-POMCP), a state-of-the-art algorithm for solving multi-agent problems, when used to solve CIRL games. FV-POMCP, like POMCP, maintains a search tree of joint histories (Amato et al., 2015) and so, a large action space still results in a large branching factor. Thus, FV-POMCP similarly struggles to solve non-trivial CIRL games.

By implicitly computing $\mathbf{H}$'s policy while running the algorithm, we reduce the branching factor in the search tree to $|\mathcal{A}^H||\mathcal{A}^R|$. Our algorithm thus scales much better to larger CIRL problems than either POMCP or FV-POMCP.

5. Experiments

We ran exact POMDP value iteration and our adapted method on the *ChefWorld* domain with two ingredients and various numbers of possible recipes, measuring the average time taken to compute the optimal policy across 20 runs. The results are given in Table 1. Our method significantly outperforms exact POMDP value iteration, which failed to compute the optimal policy in three of the five experiments after depleting the memory in our system on every run.

We ran POMCP, FV-POMCP and our approximate algorithm on the *ChefWorld* domain with two recipes, varying the number of ingredients in the game and keeping other parameters constant. We measured the average performance of each algorithm in 30000 samples across 20 runs. The results of our experiment, depicted in Figure 1, demonstrate that our algorithm outperforms the other two across the board, but especially for larger number of ingredients (i.e. number of actions available to both agents).

Additionally, we ran each algorithm on a single instantiation of the *ChefWorld* domain and tracked the value attained across 500,000 samples. Our results are depicted in Figure 2. While all algorithms eventually compute the optimal strategy, our algorithm does so significantly faster.

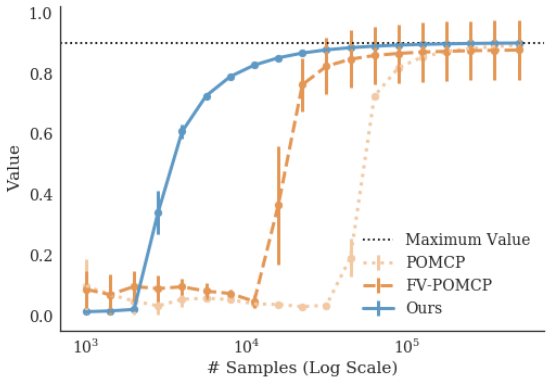

Figure 2. Value attained by POMCP, FV-POMCP and our approximate algorithm applied on the *ChefWorld* domain with 2 recipes and 6 ingredients.

References

Amato, Christopher, Oliehoek, Frans A, et al. Scalable planning and learning for multiagent pomdps. In *AAAI*, pp. 1995–2002, 2015.

Bernstein, Daniel S., Givan, Robert, Immerman, Neil, and Zilberstein, Shlomo. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

Cassandra, Anthony R., Littman, Michael L., and Zhang, Nevin Lianwen. Incremental pruning: A simple, fast, exact method for partially observable markov decision processes. In *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, Brown University, Providence, Rhode Island, USA, August 1-3, 1997*, pp. 54–61, 1997.

Hadfield-Menell, Dylan, Russell, Stuart J, Abbeel, Pieter, and Dragan, Anca. Cooperative inverse reinforcement learning. In *Advances in neural information processing systems*, pp. 3909–3917, 2016.

Kaelbling, Leslie Pack, Littman, Michael L, and Cassandra, Anthony R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1): 99–134, 1998.

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

Russell, Stuart, Norvig, Peter, and Intelligence, Artificial. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27, 1995.

Silver, David and Veness, Joel. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pp. 2164–2172, 2010.

Sondik, Edward J. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.