An Efficient, Generalized Bellman Update For Cooperative Inverse Reinforcement Learning

Dhruv Malik [*1] **Malayandi Palaniappan** [*1] **Jaime F. Fisac** [1] **Dylan Hadfield-Menell** [1] **Stuart Russell** [1] **Anca D. Dragan** [1]

Abstract

Our goal is for AI systems to correctly identify and act according to their human user's objectives. Cooperative Inverse Reinforcement Learning (CIRL) formalizes this *value alignment* problem as a two-player game between a human and robot, in which only the human knows the parameters of the reward function: the robot needs to learn them as the interaction unfolds. Previous work showed that CIRL can be solved as a POMDP, but with an action space size exponential in the size of the reward parameter space. In this work, we exploit a specific property of CIRL—the human is a full information agent—to derive an optimality-preserving modification to the standard Bellman update; this reduces the complexity of the problem by an exponential factor and allows us to relax CIRL's assumption of human rationality. We apply this update to a variety of POMDP solvers and find that it enables us to scale CIRL to non-trivial problems, with larger reward parameter spaces, and larger action spaces for both robot and human. In solutions to these larger problems, the human exhibits pedagogic behavior, while the robot interprets it as such and attains higher value for the human.

1. Introduction

As AI agents improve in their ability to optimize for a given objective, it becomes increasingly important that these agents pursue the *right* objective. The *value alignment* problem (Hadfield-Menell et al., 2016; Bostrom, 2014) is that of ensuring that robots optimize for what people want—that robot objectives are aligned with their end-users' objectives. (We henceforth use robot to refer generically to an AI agent.)

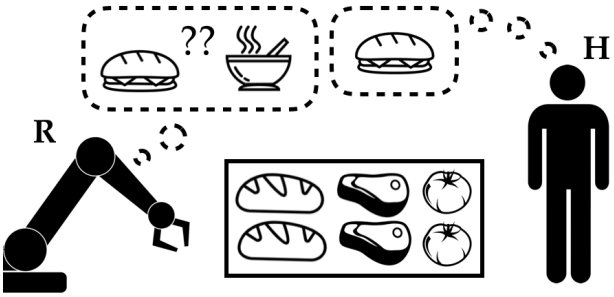

Figure 1. A CIRL game. The human **H** and the robot **R** need to work together to prepare a meal. **R** starts off unaware of which meal **H** wants, but both **H** and **R** get rewarded only if they prepare **H**'s desired meal. Solving such a CIRL game has thus far been intractable. In Section 3, we derive a modified Bellman update for computing optimal solutions to CIRL games that achieves an exponential reduction in running time and relaxes CIRL's assumption of human rationality.

A highly-capable autonomous agent working towards the wrong goal can cause undesired effects, the magnitude of which will tend to increase with the capabilities of the agent. Unfortunately, we humans have a hard time specifying what it is that we actually want. For example, customers may give mistaken instructions to an AI system and system designers may select simple, but potentially incorrect, reward functions to optimize (Amodei & Clark, 2016). Optimizing for the wrong objective can lead to unintended and negative consequences (Amodei et al., 2016).

Rather than optimize a pre-specified reward function, a robot may instead attempt to infer what people *internally* want but cannot perfectly explicate. The robot can use a person's actions to learn about the reward function over time. The most common approach for this is Inverse Reinforcement Learning (IRL) (Ng & Russell, 2000). IRL makes two implicit assumptions: 1) that the robot is a passive observer, watching the human, and 2) that the human acts as an expert in isolation, ignoring that the robot needs to learn.

Cooperative Inverse Reinforcement Learning (CIRL) (Hadfield-Menell et al., 2016) relaxes these two assumptions. It proposes a formulation in which the human $\mathbf{H}$ and the robot $\mathbf{R}$ are on the same team and collaborate to achieve the same goal. CIRL is a two-player game between $\mathbf{H}$ and

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

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

$\mathbf{R}$, in which *both take actions*, and *both get rewarded according to the same reward function*. The key to CIRL is that only $\mathbf{H}$ knows the parameters of this reward function.

Take for instance the domain from Figure 1. $\mathbf{H}$ and $\mathbf{R}$ work to prepare a meal using three ingredient types: bread, meat, and tomatoes. $\mathbf{H}$ wants to prepare either a sandwich (2 bread, 1 meat, 0 tomatoes), or tomato soup (1 bread, 1 meat, 2 tomatoes). $\mathbf{R}$ does not know *a priori* which meal $\mathbf{H}$ wants, and, to emulate the difficulty that people have in specifying what they want, we assume $\mathbf{H}$ cannot explicate this information directly to $\mathbf{R}$. At every time step, $\mathbf{R}$ and $\mathbf{H}$ each prepare a single unit of any ingredient, or no ingredient at all. They both receive reward of 1 if they succeed in preparing the right recipe, and 0 otherwise.

In this domain, CIRL captures that the human has an incentive for the robot to infer the correct recipe; and that the robot can take actions in response to the human's, as opposed to passively waiting until it knows which recipe is right. Crucially, the robot shares the reward function and has an incentive to maximize the human's internal reward. This creates an incentive to mitigate and avoid unintended consequences from misspecified rewards.

Solving a CIRL game, however, amounts to solving a Dec-POMDP. Previous work has shown that a CIRL game can be reduced to a POMDP. However, the action space in this POMDP is exponential in the size of the reward parameter space. Since POMDP algorithms scale poorly with the size of the action space, non-trivial CIRL games remain difficult to solve with this approach. Additionally, solutions to CIRL are only optimal under the assumption that the human is optimal. This is a strong assumption: it is a well-established fact in cognitive science that humans are often sub-optimal in decision making (Tversky & Kahneman, 1975; Simon, 1957). Our contributions in this paper are three-fold:

1. A Modified Bellman Update: We exploit the fact that the human is a full information agent in CIRL to derive an optimality-preserving modification of the standard Bellman update. This reduces the complexity of the problem by an exponential factor. We show how to apply this modification to existing POMDP solvers (both exact and approximate).

We further show that our modified Bellman update allows us to relax CIRL's assumption of human rationality. We instead only require that the human's policy be parameterized by her Q-values. This allows us to solve more realistic CIRL games where the human is modelled as sub-optimal.

2. Empirical Comparison: We show empirically that our method helps scale POMDP solvers to CIRL games with larger reward parameter and action spaces. We find a speed-up of several orders of magnitude for exact methods, and substantial improvements in value for approximate methods.

3. Implications: With the ability to solve more complex CIRL problems, we analyze the solutions that emerge. In contrast to IRL, we see solutions that exhibit implicit communication. The human takes explicitly suboptimal actions that are better signals for the right reward, and the robot attains higher value for the human because it can take advantage of these signals to learn faster. The coordination that emerges is a consequence of the human and robot being on the same team and reasoning about helping each other.

2. Background

2.1. POMDPs

POMDPs provide a rich model for planning under uncertainty (Sondik, 1971; Kaelbling et al., 1998). Formally, a POMDP is a tuple $\langle X, A, Z, T, O, r, \gamma \rangle$ where X is the set of states; A is the set of the agent's actions; Z is the set of observations; $T(x_t, a_t, x_{t+1})$ is the transition distribution; $O(x_{t+1}, a_t, z_{t+1})$ is the observation distribution; r is the reward function; and γ is the discount factor.

Consider a simplified instance of the cooking task from Figure 1 where $\mathbf{H}$ picks her actions according to only her desired recipe and the quantity of each ingredient prepared so far, i.e., she does not consider $\mathbf{R}$'s past or future behavior when picking her actions. The simplified cooking task is now a POMDP: $\mathbf{R}$ is the agent and $\mathbf{H}$ is a part of the environment. The state specifies $\mathbf{H}$'s desired recipe and the quantity of each ingredient already prepared. Thus, $\mathbf{H}$ picks her actions as a function of only the state.

In a POMDP, the agent cannot observe the state; instead, it maintains a belief $b \in \Delta X$, where $b(x)$ is the probability that the agent is in state x. At each time step, the agent receives an observation that helps inform its decisions. The agent in our cooking task, $\mathbf{R}$, does not know $\mathbf{H}$'s desired recipe—a component of the state. $\mathbf{R}$ observes $\mathbf{H}$'s actions and tries to infer the desired recipe from $\mathbf{H}$'s behavior.

The behavior of the agent is specified by a conditional plan $\sigma = (a, v)$; a denotes the agent's action and v is a mapping from observations to future conditional plans for the agent to follow. An example conditional plan for $\mathbf{R}$ is: prepare meat now and if $\mathbf{H}$ responds by preparing bread, prepare a second slice of bread; if $\mathbf{H}$ prepares tomatoes, prepare another batch of tomatoes; or if $\mathbf{H}$ prepares meat, do not prepare any ingredient.

The α-vector of a conditional plan contains the value of following the plan at any given state:

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

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

Value iteration (Sondik, 1971) can be used to compute the optimal conditional plan. This algorithm starts at the horizon

and works backwards. It generates new conditional plans at each time-step and evaluates them according to Eqn. 1. It constructs all potentially optimal plans of length T and selects the one with maximal value at the initial belief.

2.2. Cooperative Inverse Reinforcement Learning

Now, consider an instance of the cooking task where $\mathbf{H}$ is a second agent in the game and no longer behaves independently of $\mathbf{R}$. There is now a strong interdependence between $\mathbf{H}$'s and $\mathbf{R}$'s behavior: $\mathbf{H}$'s actions both depend on and influence $\mathbf{R}$'s belief. This problem is now no longer a POMDP; it is a CIRL game.

A CIRL game is an asymmetric-information two-player game between a human $\mathbf{H}$ and a robot $\mathbf{R}$ (Hadfield-Menell et al., 2016). $\mathbf{H}$ knows the true reward function and $\mathbf{R}$ does not initially. Formally, a CIRL game is a tuple: $M = \langle X, \{\mathcal{A}^H, \mathcal{A}^R\}, T, \{\Theta, r\}, \gamma \rangle$. 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; $T(x_t, a_t^H, a_t^R, x_{t+1})$ is the transition distribution; Θ is the set of reward parameters; r is the parameterized reward function shared by both agents; γ is the discount factor. A solution to a CIRL game is a pair of policies—one for $\mathbf{H}$ and $\mathbf{R}$ each—that maximizes the expected reward obtained by $\mathbf{H}$ and $\mathbf{R}$.

In our cooking task, Θ is the set of possible recipes. $\mathbf{R}$ does not know $\mathbf{H}$'s desired recipe, $\theta \in \Theta$. The reward function r is parameterized by Θ: both agents receive a reward of 1 if they succeed in preparing $\mathbf{H}$'s desired recipe.

Reducing a CIRL game to a POMDP A CIRL game is a Dec-POMDP (Bernstein et al., 2002) but it can be reduced to a POMDP where the optimal policy corresponds to optimal CIRL policy pairs (Hadfield-Menell et al., 2016). The states in this POMDP are tuples of world-state and reward parameter: $S = X \times \Theta$; the actions are tuples (δ^H, a^R) specifying a decision rule $\delta^H : \Theta \to \mathcal{A}^H$ for $\mathbf{H}$, which maps reward parameters to human actions, and an action a^R for $\mathbf{R}$; the observations are $\mathbf{H}$'s action at the last time step.

An example action in the reduced POMDP of the cooking task 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 $\mathbf{R}$ prepares bread (regardless of its belief).

This reduction enables us to solve a CIRL game using POMDP algorithms. However, the size of the action space in this POMDP is $|\mathcal{A}^H|^{|\Theta|}|\mathcal{A}^R|$, as shown in Figure 2. (There are $|\mathcal{A}^H|^{|\Theta|}$ possible decision rules for $\mathbf{H}$ and $|\mathcal{A}^R|$ actions for $\mathbf{R}$.) In other words, the action space in this POMDP grows exponentially with the size of the reward parameter space. Exact POMDP algorithms are exponential in the size of the action space, so this approach can only be applied to very small CIRL problems.

Additionally, the policy for $\mathbf{R}$ that is output by the reduced POMDP is optimal only if $\mathbf{H}$ is perfectly rational, i.e., if $\mathbf{H}$ is guaranteed to always pick the optimal action. This is an unrealistic assumption: humans are not idealized rational agents (Tversky & Kahneman, 1975; Simon, 1957).

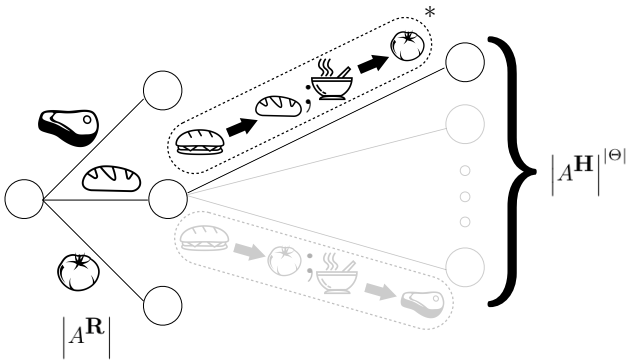

Figure 2. A node in the search tree from the POMDP reduction of our example CIRL game. Actions are tuples that contain an action for $\mathbf{R}$ and a decision rule for $\mathbf{H}$ – a mapping from her desired recipe to an action. This leads to a branching factor of $|\mathcal{A}^H|^{|\Theta|}|\mathcal{A}^R|$ and makes application of POMDP methods inefficient. In Section 3.2, we derive a modified Bellman update that prunes away all of H's decision rules but the optimal response. (In the diagram, the gray branches are pruned away by our modified Bellman update.)

3. A Modified Bellman Update for CIRL

If $\mathbf{H}$ were following a fixed policy based on the state $s = (x, \theta)$, we could encode $\mathbf{H}$ as a part of the environment. However, in the interactive setting of a CIRL game, $\mathbf{H}$ may plan for changes in $\mathbf{R}$'s belief. If we encode $\mathbf{H}$ in the environment, the dynamics change in response to $\mathbf{R}$'s intended plan and the problem is no longer a POMDP. Our main contribution is to derive a modified Bellman update for POMDP algorithms to solve this problem.

Our key idea is as follows. During planning, we know $\mathbf{R}$'s intended future response to each of $\mathbf{H}$'s actions. $\mathbf{H}$ has full state information, so the α-vectors in value iteration allow us to directly compute $\mathbf{H}$'s Q-values. We can therefore also compute her optimal action based on $\mathbf{R}$'s intended future response. This means we do not have to reason over the set of decision rules for $\mathbf{H}$: we can solve a CIRL game by instead solving a POMDP with time-varying dynamics and, importantly, where the action space has size $|\mathcal{A}^R|$. This is exponentially smaller than the action space of size $|\mathcal{A}^H|^{|\Theta|}|\mathcal{A}^R|$ in the reduced POMDP. This amounts to a modified Bellman update.

3.1. The Transition Dynamics

If $\mathbf{H}$ follows a policy that depends only on the state $s = (x, \theta)$, the dynamics of the game can be computed as:

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

However, in the CIRL formulation, $\mathbf{H}$ does not behave according to a fixed policy. $\mathbf{H}$, who is assumed to be rational, behaves according to her Q-values and picks the action that maximizes her expected value. Due to the interdependence between $\mathbf{H}$'s and $\mathbf{R}$'s behavior, these Q-values depend on $\mathbf{R}$'s conditional plan. The dynamics then are:

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

These dynamics change over time since they depend on the robot's future behavior. However, $\mathbf{R}$'s behavior depends on these dynamics, so, we cannot pre-compute them as part of a POMDP reduction. However, we do have access to $\mathbf{R}$'s future conditional plan *during* planning. This means we can compute $\mathbf{H}$'s Q-values, and, consequently, the transition probabilities, with a modification to the Bellman update.

3.2. Adapting POMDP Value Iteration

POMDP value iteration rolls back the values of the game from the horizon, storing them as α-vectors. If $\mathbf{R}$ follows a plan $\sigma = (a^R, v)$, then we can compute $\mathbf{H}$'s Q-values as

$$
Q_H(s, a^H, \sigma) = \sum_{s'} T(s, a^H, a^R, s') \cdot \alpha_{v(a^H)}(s').
$$

$\mathbf{H}$'s optimal action maximizes this expression. To leverage this, we adapt the Bellman update in Eqn. 1 to replace the dynamics of the game with $P(s', a^H \mid s, \sigma)$ from Eqn. 2. The modified Bellman update is then:

$$
\alpha_\sigma(s) = R(s) + \gamma \cdot \max_{a^H} \sum_{s' \in S} T(s, a^H, a^R, s') \cdot \alpha_{v(a^H)}(s'). \tag{3}
$$

We can then use value iteration with this modified Bellman update to compute the $\mathbf{R}$'s optimal policy. The following theorem establishes that the modified Bellman update, Eq. 3, is optimality-preserving.

Theorem 1. *For any CIRL game, the policy computed by value iteration with the modified Bellman update is optimal.*

All theorem proofs are presented in Appendix A in the supplementary material.

a. We let $T(s, a^H, a^R, s') = T(x, a^H, a^R, x') \cdot \mathbb{1}(\theta = \theta')$.

This modification to the Bellman update allows us to solve a CIRL game without having to include the set of $\mathbf{H}$'s decision rules in the action space. As depicted in Figure 2, the modified Bellman update computes $\mathbf{H}$'s optimal action given the current state and the robot's plan; all of $\mathbf{H}$'s other actions are pruned away in the search tree. The size of the action space is then $|\mathcal{A}^R|$ instead of $|\mathcal{A}^H|^{|\Theta|}|\mathcal{A}^R|$. POMDP algorithms are exponential in the size of the action space; this modification therefore allows us to solve CIRL games much more efficiently. The following theorem establishes the complexity gains made by algorithm.

Theorem 2. *The modification to the Bellman update presented above reduces the time and space complexity of a single step of value iteration by a factor of* $\mathcal{O}\left(|\mathcal{A}^H|^{|\Theta|}\right)$.

3.3. Relaxing CIRL's Assumption of Rationality

To achieve value alignment, we can now efficiently solve a CIRL game to find an optimal policy for $\mathbf{R}$. However, this policy is optimal only if $\mathbf{H}$ is perfectly rational: a strong assumption. This is rarely true in reality; we thus want to find an optimal policy for $\mathbf{R}$ even when $\mathbf{H}$ is sub-optimal.

In addition to improving efficiency, our modified Bellman update allows us to do exactly that and relax CIRL's assumption of rationality. The dynamics of our modified Bellman update, presented above as Eq. 2, do not require that $\mathbf{H}$ is perfectly rational. These dynamics will remain well-defined so long as we know the distribution over $\mathbf{H}$'s actions, π_H, and can compute the probability that she picks any action from her current state. To avoid compromising the interactive nature of CIRL, we require that π_H must be a function of $\mathbf{H}$'s Q-values, which account for the robot's future behavior. The dynamics of the game are then given by:

$$
P(s', a^H \mid s, \sigma) = T(s, a^H, a^R, s') \cdot \pi_H(a^H \mid Q_H(s, a^H, \sigma)).
$$

The modified Bellman update is then:

$$
\begin{aligned}
\alpha_\sigma(s) = R(s) + \gamma \cdot \sum_{a^H} \pi_H(a^H \mid Q_H(s, a^H, \sigma)) \cdot \\
\sum_{s' \in S} T(s, a^H, a^R, s') \cdot \alpha_{v(a^H)}(s').
\end{aligned} \tag{4}
$$

With this modified Bellman update, we may now use value iteration to solve CIRL games without assuming that the human is perfectly rational. We instead only require that the human selects her actions with respect to her Q-values. This restriction is rather broad and includes a variety of models of human decision making from cognitive science. A popular example of such a model is Boltzmann-rationality, where the human picks her actions according to a Boltzmann distribution over her Q-values, i.e.,

$$
\pi_H(a^H \mid Q_H(s, a^H, \sigma)) \propto \exp(\beta \cdot Q_H(s, a^H, \sigma))
$$

where β is a parameter which controls how rational the human is. (A higher β corresponds to a more rational human.)

Algorithm 1 Adapted Value Iteration for CIRL Games

$\Gamma_t \leftarrow$ Set of trivial plans
for $t \in \{T-1, T-2, \ldots, 1, 0\}$ **do**
  $\Gamma_{t+1} \leftarrow \Gamma_t$
  $\Gamma_t \leftarrow$ Set of all plans beginning at time t
  for $\sigma \in \Gamma_t$ **do**
    for $s = (x, \theta) \in S$ **do**
      $Q_H(s, a^H, \sigma) = \sum_{s'} T(s, a^H, a^R, s') \cdot \alpha_{v(a^H)}(s')$
      $\alpha_\sigma(s) = R(s) + \gamma \cdot \sum_{a^H} \pi_H(a^H \mid Q_H(s, a^H, \sigma)) \cdot Q_H(s, a^H, \sigma)$
    end for
  end for
  $\Gamma_t \leftarrow$ Prune(Γ_t)
end for
$a^R_* = argmax_{\sigma \in \Gamma_0} \alpha_\sigma \cdot b_0$
Return a^R_*

The time and space complexity of value iteration with this Bellman update is identical to that with the modified Bellman update presented in Section 3.2, and analyzed in Theorem 2. The pseudocode for our adapted algorithm is presented as Algorithm 1 below.

4. Adapting Approximate Algorithms

Approximate algorithms for POMDPs often rely on variants of the Bellman update. This lets us use our modified Bellman update to improve approximate algorithms for CIRL.

4.1. PBVI

Background Point Based Value Iteration (PBVI) is an approximate algorithm used to solve POMDPs (Pineau et al., 2003). The algorithm maintains a representative set of points in belief space and an α-vector at each of these belief points. It performs approximate value backups at each of these belief points using this set of α-vectors. Let Γ_{t+1} denote the set of α-vectors for plans that begin at time $t+1$. The value at time t at a belief b is approximated as:

$$V(b) = \max_{a \in A} \left[\sum_{s \in S} R(s) b(s) + \gamma \sum_{o \in O} \max_{\alpha \in \Gamma_{t+1}} \sum_{s \in S} \left(\sum_{s' \in S} P(s', o \mid s, a) \alpha(s) \right) b(s) \right].$$

The algorithm trades off computation time and solution quality by expanding the set of belief points over time: it randomly simulates forward trajectories in the POMDP to produce new, reachable beliefs.

Our Adaptation If $\mathbf{R}$ takes action a^R and follows a conditional plan σ, then $\mathbf{H}$'s Q-values are $Q_{\mathbf{H}}(x, a^H, a^R, \alpha) = \sum_{s'} T(s, a^H, a^R, s') \cdot \alpha_\sigma(s')$. Notice that we can compute these Q-values at each step of PBVI. This lets us use the modified Bellman update and to adapt PBVI to solve CIRL games specifically. We replace the transition-observation distribution in the PBVI backup rule with

$$P(s', a^H \mid s, a^R, \alpha) = T(s, a^H, a^R, s') \cdot \pi_{\mathbf{H}}(Q_{\mathbf{H}}(x, a^H, a^R, \alpha)).$$

The modified backup rule for PBVI is thus given by

$$V(b) = \max_{a^R \in \mathcal{A}^R} \left[\sum_{s \in S} R(s) b(s) + \gamma \sum_{a^H} \max_{\alpha \in \Gamma_{t+1}} \sum_{s \in S} \left(\sum_{s' \in S} P(s', a^H \mid s, a^R, \alpha) \alpha(s) \right) b(s) \right].$$

We now show that the approximate value function in PBVI converges to the true value function. Let ϵ_B denote the density of the set of belief points B in PBVI. Formally, $\epsilon_B = \max_{b \in \Delta} \min_{b' \in B} ||b - b'||_1$ is the maximum distance from any reachable, legal belief to the set B.

Theorem 3. *For any belief set B and horizon n, the error of our adapted PBVI algorithm $\eta = ||V_n - V_n^*||_\infty$ is bounded as*

$$\eta \leq \frac{(R_{\max} - R_{\min})\epsilon_B}{(1-\gamma)^2}.$$

4.2. POMCP

Background POMCP is a Monte Carlo tree-search (MCTS) based approximate algorithm for solving large POMDPs (Silver & Veness, 2010). The algorithm constructs a search tree of action-observation histories and uses Monte Carlo simulations to estimate the value of each node in the tree. During search, actions within the tree are selected by UCB1. This maintains a balance between exploiting actions known to have good return and exploring actions not yet taken (Kocsis & Szepesvári, 2006). At leaf nodes, a rollout policy accrues reward which is then backed up through the tree. The algorithm estimates the belief at each node by keeping track of the hidden state from previous rollouts.

POMCP scales well with the size of the state space, but not with the size of the action space, which determines the branching factor in the search tree. POMCP is thus ill-suited to solving the reduced POMDP of CIRL games since the size of the action space is $|\mathcal{A}^H|^{|\Theta|}|\mathcal{A}^R|$.

Our Adaptation Using the idea behind our modified Bellman update, we adapt POMCP to solve CIRL games more efficiently. We approximate $\mathbf{H}$'s policy while running the algorithm (much like we exactly compute $\mathbf{H}$'s policy in exact value iteration). We maintain a live estimate of the sampled Q-values for $\mathbf{H}$ at each node. With enough exploration of the search tree (for instance, if actions are selected using UCB1), the estimated Q-values converge to the true values (in the limit). This guarantees that $\mathbf{H}$'s policy converges to her true policy. The following result establishes convergence of our algorithm.

Theorem 4. *With suitable exploration, the value function constructed by our adapted POMCP algorithm converges in probability to the optimal value function,* $V(h) \to V^*(h)$. *As the number of visits* $N(h)$ *approaches infinity, the bias of the value function* $\mathbb{E}[V(h) - V^*(h)]$ *is* $O(log(N(h))/N(h))$.

The pseudocode for our adapted PBVI and our adapted POMCP algorithm is presented as Algorithm 1 and 2 respectively in Appendix B in the supplementary material.

5. Related Work

POMDP Algorithms We chose to explicate our modified Bellman update in the context of PBVI and POMCP because they are the seminal point-based and MCTS algorithms respectively, for solving POMDPs. For example, SARSOP (Kurniawati et al., 2008) and DESPOT (Ye et al., 2017), two state-of-the-art algorithm for POMDPs, are variants of PBVI and POMCP respectively. The principles we outlined in Sections 3 and 4 can be easily adapted to a large variety of point-based and MCTS algorithms, including any which may be developed in the future, to derive even more efficient algorithms for solving CIRL games.

MOMDP Algorithms The POMDP representation of CIRL is also a mixed-observability Markov decision process (MOMDP) since the state space can be factored into a fully- and a partially-observable component. This structure allows for more efficient solution methods; Ong et al. (2010) leverage the factored nature of the state space to create a lower dimensional representation of belief space. This core idea is orthogonal to ours, which exploits CIRL's information asymmetry instead. The two can be leveraged together.

Dec-POMDP Algorithms Dec-POMDP algorithms can be used to solve CIRL directly, without using the POMDP reduction. These solution methods are generally intractable, but recent work has made progress on this front. Such Dec-POMDP algorithms which attempt to prune away unreasonable strategies resemble our approach. Amato et al. (2009) use reachability analysis to identify reachable states, then consider all policies which are useful at such states. Hansen (2004) model other agents possible strategies as part of a players belief, and prune away weakly dominated strategies at each step. While such approaches use heuristics to prune away some suboptimal strategies, we leverage the information structure of CIRL to compute the optimal strategy for $\mathbf{H}$ and prune away *all* other strategies. This guarantees an exponential reduction in complexity while preserving optimality; this is not true for the other methods.

Value Alignment Recent work has explored relaxing the rationality requirement of CIRL (Fisac et al., 2017). Our work improves on their relaxation in several ways: (1) Their Bellman update assumes that the human acts Boltzmann-rationally. Our modification can model a large variety of human behaviors (including this). (2) Their discretized belief value iteration algorithm has neither the guarantee of optimality of our adapted VI algorithm nor the scalability of our adapted POMCP/PBVI algorithms.

Table 1. Time taken (s) to find the optimal robot policy using exact VI and our adaptation of it for various numbers of possible recipes. NA denotes that the algorithm failed to solve the problem.

# 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

6. Experiments

We now verify that our modified Bellman update allows POMDP algorithms to solve CIRL games more efficiently than the standard update. We ran three experiments: one for exact value iteration (VI), PBVI, and POMCP each. The results of the PBVI experiment are presented in Appendix C.1 in the supplementary material due to space constraints. To verify the results of our experiments, we ran further experiments on a second domain. The details and results of these experiments are presented in Appendix C.2.

6.1. Experimental Design

Domain Our experimental domain is based on our running example from Section 1. Assume there are m recipes and n ingredients. The state space is an n-tuple representing the quantity of each ingredient prepared thus far. At each time step, each agent can prepare any of the n ingredients or none at all. Each of the m recipes corresponds to a different θ (i.e. reward parameter) value. Both agents receive a reward of 1 if $\mathbf{H}$'s desired recipe is prepared correctly and a reward of 0 otherwise. The robot $\mathbf{R}$ begins the game entirely uncertain about $\mathbf{H}$'s desired recipe i.e. $\mathbf{R}$ has a uniform belief over Θ.

We want to stress that this experimental domain is not trivial: one of the domains we managed to solve in our experiments had $\sim 10^{10}$ states.

Manipulated Variables Our primary variable is the type of Bellman update used: modified vs. standard. We also varied the number of recipes, i.e., size of the reward parameter space, and the number of ingredients, i.e., size of $\mathbf{H}$'s and $\mathbf{R}$'s action space.

Dependent Measures In our first experiment (exact VI), we measured the time taken by the algorithms to solve the problem. In our second experiment (PBVI), we measured the value attained by the algorithms in a fixed time. In our third experiment (POMCP), we measured the value attained by the algorithms in a fixed number of samples.

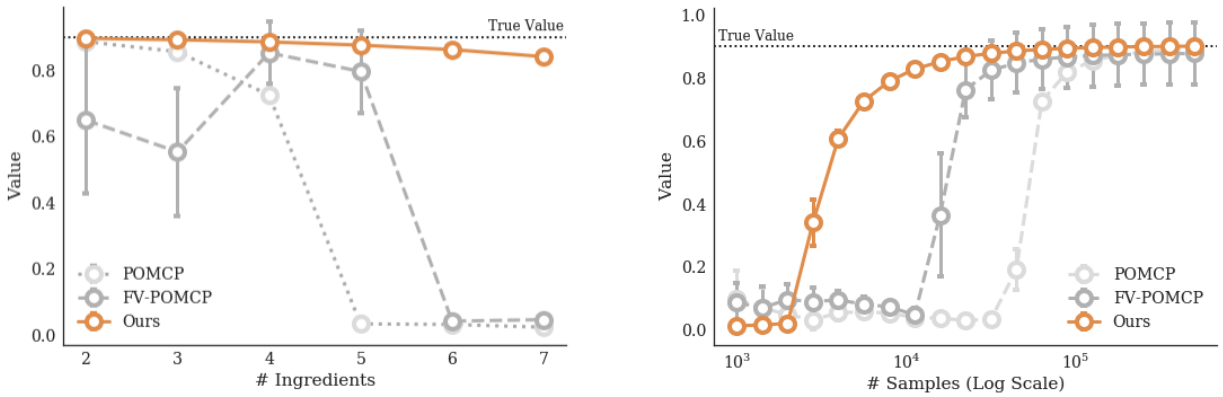

Figure 3. (Left) The value attained by POMCP, FV-POMCP, and our adapted algorithm in 30,000 samples with various numbers of ingredients. (Right) Value attained by POMCP, FV-POMCP and our approximate algorithm with 2 recipes and 6 ingredients.

Hypothesis *POMDP algorithms are more efficient at solving CIRL games with the modified Bellman update than with the standard Bellman update.*

6.2. Analysis

Exact VI In our first experiment, we compared the time taken by exact VI and by our adaptation of it with the modified Bellman update. We first fixed the number of ingredients at two and varied the number of recipes in the domain. Table 1 compares the results. For the simpler problems, where the number of recipes was 2 or 3, our adapted algorithm solved the problem up to $\sim$3500$\times$ faster than exact VI. On more complex problems where the number of recipes is greater than 3, exact VI failed to solve the problem after depleting our system's 16GB memory; in contrast, our adapted algorithm solved each of these more complex problems in less than 0.5 seconds. We next fixed the number of recipes and compared the performance of both these algorithms for various numbers of ingredients. Both the exact methods, but especially the one using the standard Bellman update, scaled much worse with the number of ingredients than with the number of recipes. With even three ingredients, exact VI timed out and failed to solve the problem within two hours; our algorithm however solved the problem in five seconds.

POMCP We compared the value attained in 30,000 samples by POMCP and by our adaptation with the modified Bellman update. We additionally compared these algorithms with FV-POMCP, a state-of-the-art MCTS method for solving MPOMDPs, a type of Dec-POMDP in which all agents can observe each others' behavior (as in CIRL).

We first fixed the number of recipes at 2 and varied the number of ingredients. Our adapted algorithm outperformed the other two algorithms across the board, especially for large numbers of ingredients. The results of this comparison are presented in Figure 3 (left). POMCP did poorly on games with more than 4 ingredients. Although FV-POMCP scaled better to more complex games than POMCP, its values had high variance. For the largest games, with 6 and 7 ingredients, our adapted algorithm was the only one capable of solving the problem in 30,000 iterations. We also compared the value attained by each algorithm across 500,000 samples on the 6 ingredient game. The results of this comparison are depicted in Figure 3 (right). Our algorithm converged to the true value faster than either of the other algorithms.

We next fixed the number of ingredients at 4 and varied the number of recipes. We found that the results of this experiment broadly matched the results of our previous experiment where we varied the number of ingredients. For example, with 4 recipes, our method achieves an average value of 0.631 ± 0.221 in 30,000 iterations while POMCP gets 0.429 ± 0.183 and FV-POMCP gets 0.511 ± 0.124.

These results together demonstrate that POMDP algorithms with our modified Bellman update scales much better to more complex CIRL games than with the standard Bellman update. This offers strong evidence for our hypothesis.

7. Discussion

The previous section showed that we can now solve larger, non-trivial CIRL games. While we are still far from addressing value alignment in the high dimensional and continuous real world, our work allows us to analyze CIRL solutions to non-trivial problems and understand their implications for value alignment.

7.1. CIRL vs IRL

In the absence of CIRL solutions, a standard approach to learning the human's internal reward is Inverse Reinforcement Learning (IRL) (Ng & Russell, 2000). We thus compare what advantages CIRL has compared to IRL. On a collaborative task, IRL is equivalent to assuming that $\mathbf{H}$ chooses her actions in isolation, and $\mathbf{R}$ uses observations of

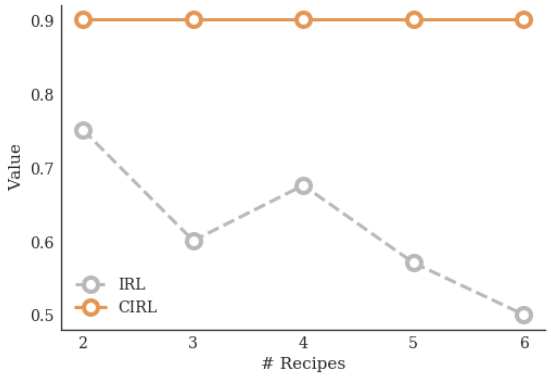

Figure 4. Value attained by CIRL and standard IRL on the cooking domain with various numbers of possible recipes. Unlike IRL, CIRL produces solutions where $\mathbf{H}$ picks her actions pedagogically and $\mathbf{R}$ reasons about $\mathbf{H}$ accordingly.

her behavior to infer her preferences. Specifically, $\mathbf{H}$ solves a single-agent, fully-observable, variant of the problem, and $\mathbf{R}$ responds by solving the POMDP described in Section 2.

We fix the number of ingredients at 3 and vary the number of recipes. Figure 4 shows the results. In each experiment the optimal CIRL solution prepares the correct recipe while the IRL solution fails to do so up to 50% of the time. To understand the nature of this difference in performance, we analyze the CIRL and IRL solutions. Consider a case of our running example with two recipes. The state is a tuple $(\#meat, \#bread, \#tomatoes)$ and $\Theta = \{sandwich = (1, 2, 0), soup = (1, 1, 2)\}$. For both approaches, $\mathbf{R}$ initially prepares meat. In the baseline IRL solution, $\mathbf{H}$ can initially make any ingredient if she wants soup and can make meat or bread if she wants a sandwich. In each case, she chooses uniformly at random between allowed ingredients. This conveys some information about her desired recipe, but is not enough to uniquely identify it. Since the same ingredient is optimal for multiple recipes, $\mathbf{R}$ is still confused after one turn. This means $\mathbf{R}$ will sometimes fail to complete the desired recipe, reducing average utility.

The CIRL solution, in contrast, relies on the implicit communication between the human and the robot. Here, if $\mathbf{H}$ wants soup, she prepares tomatoes, as opposed to any ingredients that are common with the sandwich. Even more interestingly, she waits (i.e. does nothing) if she wants a sandwich. This is pure signaling behavior—*waiting is suboptimal in isolation*, but picking an ingredient is more likely to confuse the robot. In turn $\mathbf{R}$ knows that $\mathbf{H}$ would have picked tomatoes if she wanted soup, and responds appropriately.

In other words, $\mathbf{H}$ teaches the robot about her preferences with her action selection. This works because $\mathbf{H}$ knows that $\mathbf{R}$ will interpret her behavior pragmatically, i.e., $\mathbf{R}$ expects to be taught by $\mathbf{H}$. This is reflected in the experiment: the optimal CIRL solution prepares the correct recipe each time.

The value alignment problem is necessarily cooperative: without the robot, the human is unable to complete her desired task, and without explicit signaling from the human, the robot learns inefficiently, is less valuable and is more likely to make a mistake. Pedagogic behavior from $\mathbf{H}$ naturally falls out of the CIRL solution. In response, $\mathbf{R}$ interprets $\mathbf{H}$'s actions pragmatically. These instructive and communicative behaviors allow for faster learning and create an opportunity to generate higher value for the human.

7.2. CIRL with Suboptimal Humans

To further investigate the performance of CIRL in realistic settings (e.g., where $\mathbf{H}$ may not be rational), we ran another experiment. We varied whether $\mathbf{H}$ behaved according to CIRL or IRL, $\mathbf{R}$'s model of $\mathbf{H}$ in training (rational or Boltzmann-rational), and the actual model of $\mathbf{H}$ (same as previous). We measured the proportion of times they prepared the correct recipe in each setting, fixing the number of ingredients at 3 and recipes at 4. Fig. 2 in Appendix D in the supp. material shows the results. (We also conducted a more comprehensive experiment with 20 human behaviors instead of 2. Results are presented in Appendix E in the supp. material.)

Averaged across different models of $\mathbf{H}$ used to train $\mathbf{R}$, when $\mathbf{H}$ behaved according to CIRL, $\mathbf{H}$ and $\mathbf{R}$ succeeded in preparing the correct recipe $> 90\%$ of the time. This was also true when $\mathbf{H}$ behaved Boltzmann-rationally. This suggests that the pedagogic behavior that arises from CIRL makes it more robust to any sub-optimality from $\mathbf{H}$. In contrast, when $\mathbf{H}$ behaved as in IRL (i.e., not pedagogically), they only prepared the correct recipe $\sim 70\%$ of the time when $\mathbf{H}$ was rational, and $\sim 40\%$ of the time when $\mathbf{H}$ was not. So, the importance of pedagogic behavior from $\mathbf{H}$ to achieve value alignment is clear.

7.3. Do People Behave Pedagogically?

A question arises at this point as to whether *real* people will adopt the pedagogic behavior predicted by CIRL solutions. To start testing this, we did run a (very preliminary) pilot study to start investigating whether CIRL improves interactions with real people. The details of this experiment are presented in Appendix F in the supplementary material. We found some encouraging evidence that suggests people do indeed behave pedagogically when collaborating with a robot; and that a CIRL-trained robot is can be better at exploiting this pedagogic behavior to achieve fluid human-robot collaboration than an IRL-trained robot.

In future work, we plan to conduct a more extensive human subjects study to validate these preliminary findings.We additionally plan to explore techniques to make the robot better elicit pedagogic behavior from the human in their interaction and to make CIRL robust to variations in human behavior.

Acknowledgements

This work was supported in part by grants from the NSF NRI, and the Open Philantrophy Project.

References

Amato, C., Dibangoye, J. S., and Zilberstein, S. Incremental Policy Generation for Finite-Horizon DEC-POMDPs. In Gerevini, A., Howe, A. E., Cesta, A., and Refanidis, I. (eds.), *ICAPS*. AAAI, 2009. ISBN 978-1-57735-406-2. URL http://dblp.uni-trier.de/db/conf/aips/icaps2009.html#AmatoDZ09.

Amodei, D. and Clark, J. Faulty Reward Functions in the Wild. https://blog.openai.com/faulty-reward-functions/, 2016.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete Problems in AI Safety. *CoRR*, abs/1606.06565, 2016. URL http://arxiv.org/abs/1606.06565.

Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

Bostrom, N. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, Oxford, UK, 1st edition, 2014. ISBN 0199678111, 9780199678112.

Fisac, J. F., Gates, M. A., Hamrick, J. B., Liu, C., Hadfield-Menell, D., Palaniappan, M., Malik, D., Sastry, S. S., Griffiths, T. L., and Dragan, A. D. Pragmatic-pedagogic value alignment. *International Symposium on Robotics Research*, 2017.

Hadfield-Menell, D., Russell, S. J., Abbeel, P., and Dragan, A. Cooperative Inverse Reinforcement Learning. In *Advances in neural information processing systems*, pp. 3909–3917, 2016.

Hansen, E. A. Dynamic Programming for Partially Observable Stochastic Games. In *In Proceedings Of The Nineteenth National Conference On Artificial Intelligence*, pp. 709–715, 2004.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and Acting in Partially Observable Stochastic Domains. *Artif. Intell.*, 101(1-2):99–134, May 1998. ISSN 0004-3702. doi: 10.1016/S0004-3702(98)00023-X. URL http://dx.doi.org/10.1016/S0004-3702(98)00023-X.

Kocsis, L. and Szepesvári, C. Bandit Based Monte-Carlo Planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pp. 282–293, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-45375-X, 978-3-540-45375-8. doi: 10.1007/11871842_29. URL http://dx.doi.org/10.1007/11871842_29.

Kurniawati, H., Hsu, D., and Lee, W. S. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *In Proc. Robotics: Science and Systems*, 2008.

Ng, A. Y. and Russell, S. Algorithms for Inverse Reinforcement Learning. In *in Proc. 17th International Conf. on Machine Learning*. Citeseer, 2000.

Ong, S. C., Png, S. W., Hsu, D., and Lee, W. S. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068, 2010.

Pineau, J., Gordon, G., and Thrun, S. Point-Based Value Iteration: An Anytime Algorithm for POMDPs. In *IJCAI*, volume 3, pp. 1025–1032, 2003.

Silver, D. and Veness, J. Monte Carlo Planning in Large POMDPs. In *Advances in neural information processing systems*, pp. 2164–2172, 2010.

Simon, H. A. Models of man; social and rational. 1957.

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

Tversky, A. and Kahneman, D. Judgment under uncertainty: Heuristics and biases. In *Utility, probability, and human decision making*, pp. 141–162. Springer, 1975.

Ye, N., Somani, A., Hsu, D., and Lee, W. DESPOT: Online POMDP Planning with Regularization. 58:231–266, 2017.