Deceptive Reinforcement Learning in Model-Free Domains

Alan Lewis, Tim Miller

The University of Melbourne
lewisa1@student.unimelb.edu.au, tmiller@unimelb.edu.au

Abstract

This paper investigates deceptive reinforcement learning for privacy preservation in model-free and continuous action space domains. In reinforcement learning, the reward function defines the agent's objective. In adversarial scenarios, an agent may need to both maximise rewards and keep its reward function private from observers. Recent research presented the *ambiguity model* (AM), which selects actions that are ambiguous over a set of possible reward functions, via pre-trained Q-functions. Despite promising results in model-based domains, our investigation shows that AM is ineffective in model-free domains due to misdirected state space exploration. It is also inefficient to train and inapplicable in continuous action space domains. We propose the *deceptive exploration ambiguity model* (DEAM), which learns using the deceptive policy during training, leading to targeted exploration of the state space. DEAM is also applicable in continuous action spaces. We evaluate DEAM in discrete and continuous action space path planning environments. DEAM achieves similar performance to an optimal model-based version of AM and outperforms a model-free version of AM in terms of path cost, deceptiveness and training efficiency. These results extend to the continuous domain.

1 Introduction

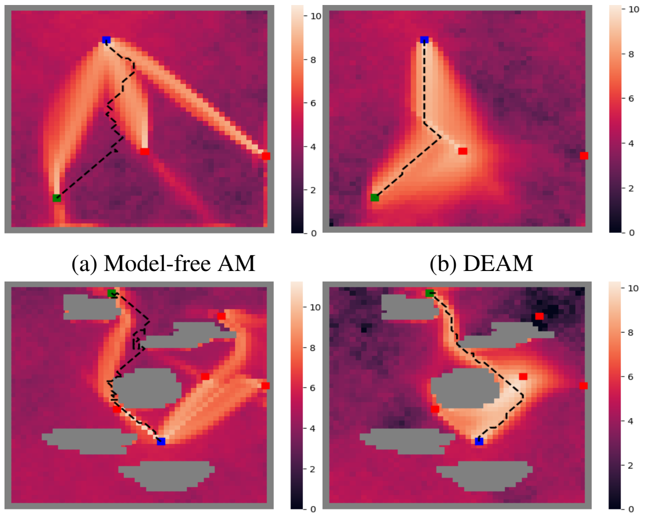
Model-free reinforcement learning (RL) has emerged as a powerful framework for solving a wide variety of tasks, from challenging games (Silver et al. 2017) to the protein folding problem (Jumper et al. 2021). RL agents learn by interacting with their environment, with the objective to maximise accumulated rewards over a trajectory. The *reward function* is a succinct and robust definition of the task, from which one can deduce an agent's policy (Sutton and Barto 2018; Ng and Russell 2000). There are cases where we want to preserve the privacy of a reward function (Ornik and Topcu 2018). Consider a military commander moving troops. If they fail to keep the destination private, an adversary can ambush them. This creates a dual objective—to reach the destination efficiently while not revealing the destination for as long as possible. As goal-directed actions leak information, optimising for a reward-function while keeping it private is challenging.

Deception is a key mechanism for privacy preservation in artificial intelligence (AI) (Liu et al. 2021; Savas, Verginis, and Topcu 2021; Ornik and Topcu 2018; Masters and Sardina 2017b). Deception is the distortion of a target's perception of reality by hiding truths and/or conveying falsities (Bell 2003; Whaley 1982). In the military example, the commander may hide the true destination by moving towards a fake destination. Although there are successful deceptive AI methods, most are model-based, requiring prior knowledge of the environment dynamics (Savas, Verginis, and Topcu 2021; Kulkarni et al. 2018; Masters and Sardina 2017b).

Recently, Liu et al. (2021) introduced the *ambiguity model* (AM) for deception in RL. AM selects actions that are ambiguous over a set of candidate reward functions—including the real reward function and at least one fake reward function. AM estimates the probability that a candidate is real from an observer's perspective using a pre-trained Q-function for each candidate. It then selects the action that maximises the entropy of this probability distribution. Liu et al. (2021) learn these Q-functions using value iteration.

We carefully study the AM, finding that it surprisingly fails in model-free domains—that is, using a model-free approach to learn Q-functions leads to poor performance. Our analysis shows why: by separately pre-training honest subagents to learn Q-functions, AM explores the states along the honest trajectories 'towards' each candidate reward function, instead of those visited by the deceptive policy. Figures 1a and 1c show AM's state space exploration in a path planning domain (lighter parts denote more frequently visited states). There is little overlap between the heavily explored states and the final deceptive path; showing that AM over-explores states that have little influence on its final policy, and under-explores states along good deceptive trajectories. This leads to inaccurate Q-values for these states and a poor policy. AM's ineffectiveness with model-free methods prevents its application to many problems, including model-free and simulation-based problems.

We propose the *deceptive exploration ambiguity model* (DEAM), an ambiguity-based model that collectively learns candidate Q-functions using the entropy-based policy to select actions during training. DEAM has three key improvements compared to AM. First, by training with a deceptive policy, DEAM explores states along deceptive trajectories, shown by the overlap between the frequently visited states and the final path in Figures 1b and 1d. This leads to effective deceptive behaviour in model-free domains. Sec-


(a) Model-free AM (b) DEAM

(c) Model-free AM (d) DEAM

Figure 1: State-visitation heatmaps vs deceptive path. *Start:* blue. *Real goal:* green. *Fake goals:* red. *Final path:* dashed line. Lighter parts denote more frequently visited states. State visit frequencies are log-scaled.

ond, DEAM extends to continuous action space domains by using actor-critic (AC) subagents to learn Q-functions and changes to action selection. Third, DEAM shares experiences between subagents, improving training efficiency as all subagents learn from every environment interaction.

We evaluate DEAM in discrete and continuous path planning domains, assessing deceptiveness, path costs and training efficiency. DEAM is more deceptive than an honest agent, with similar performance to an optimal, model-based (value iteration) version of AM. DEAM resolves the misdirected state space exploration problem, leading to a more cost-efficient policy, reaching the real goal in $1.1\times$ the optimal path cost compared to $1.6\times$ for the model-free version of AM. DEAM converges to a deceptive policy in approximately 70% of the allotted training steps, while the model-free AM does not converge in the allotted interactions.

Overall, we make three key contributions to deceptive RL:

1. Via a thorough analysis, we identify important weaknesses in AM that are not apparent in Liu et al. (2021)'s paper. These weaknesses make AM ineffective with model-free methods, limiting its application.
2. We resolve these weaknesses by introducing DEAM, the only model-free deceptive RL framework. This enables a significant expansion in the scope of deceptive RL, including environments with unknown dynamics.
3. By evaluating the agents against passive and active adversaries, we show scenarios where DEAM and AM's entropy-based model leads to counter-productive behaviour, pointing to important areas for future research.

2 Related Work

Masters and Sardina (2017b) define deception through *simulation* and *dissimulation* (Bell 2003; Whaley 1982). We focus solely on dissimulation. Dissimulation measures the entropy (Shannon 1948) over the candidate goal probabilities: dissimulation $= -\sum_{g_i \in G} P(g_i \mid \vec{o}) \cdot \log_2 P(g_i \mid \vec{o})$, where G is a set of candidate goals and $P(g_i \mid \vec{o})$ is the probability that g_i is real from an observer's perspective, given the observation sequence $\vec{o}$. This definition is measurable, enabling agents to manage the trade-off between deception and rewards. However, their deceptive agent requires deterministic environments with known dynamics. DEAM is applicable in model-free, stochastic and continuous environments.

Ornik and Topcu (2018) extend deception to stochastic environments with a deceptive Markov Decision Process (MDP), which incorporates the observer's beliefs into a regular MDP. Section 3 has a precise definition. They highlight several challenges to optimisation in a deceptive MDP, such as a lack of knowledge of the observer's beliefs and unifying beliefs with environment rewards. Savas, Verginis, and Topcu (2021) use an intention recognition system to model the observer's beliefs. Despite effectively balancing deception and rewards, their approach is model-based.

Liu et al. (2021)'s AM for deceptive RL is closest to ours. Although it only requires Q-functions, and therefore theoretically extends to model-free domains, they use value iteration to learn these Q-functions in their evaluation. We show that AM is ineffective when using model-free RL, due to misdirected state space exploration. Our model resolves this issue by following the deceptive policy during training. We also enable application in continuous action spaces.

Finally, intention and plan recognition research is relevant since it is an inverse problem to deception. Plan recognition is the task of inferring an agent's plan by observing its behaviour (Ramírez and Geffner 2009; Masters and Sardina 2019). A common approach is cost-based plan recognition, which measures the divergence of behaviour from optimal. Intuitively, a rational agent will take a cost-minimising path. Therefore, the goal that best explains the observed path is more likely to be the real goal. This can be used to generate a probability distribution over the candidate goal-set (Masters and Sardina 2017a). *Inverse reinforcement learning* (IRL) is the task of deducing a reward function given traces of the agent's behaviour (Ng and Russell 2000). However, IRL requires many behaviour traces whereas deceptive RL aims to deceive over a single trace of behaviour.

3 Preliminaries

Definition 1 (Markov decision process (MDP) (Puterman 2014))**.** *An MDP is defined by a tuple* $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma)$*, where* $\mathcal{S}$ *is the state space,* $\mathcal{A}$ *is the action space,* $\mathcal{T}(s_t, a_t, s_{t+1})$ *is a transition function defining the probability of moving from state* s_t *to state* s_{t+1} *given action* a_t*,* $r(s_t, a_t, s_{t+1})$ *is the reward received for executing action* a_t *in state* s_t *and transitioning to state* s_{t+1}*, and* $\gamma \in (0, 1)$ *is a discount factor.*

An MDP is solved by a *decision policy* $\pi : \mathcal{S} \to \mathcal{A}$, which selects actions given a state. The objective is to maximise the value function: $V_\pi(s) = \mathbb{E}[\sum_{t \in T} \gamma^t r(s_t, a_t, s_{t+1})]$.

RL approaches. We separate RL approaches into three groups of methods: (1) *Value-based*, such as *Q-learning*; (2) *Policy gradient (PG)*; and (3) *AC*. Value-based methods learn a Q-function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, which maps all state-action pairs onto an estimate for the true value function (Sutton and Barto 2018); $Q(s_t, a_t)$ represents the future expected reward from selecting action a_t in state s_t and following π from there. Q-learning agents iterate over the action space to maximise Q-values, which is infeasible in continuous action spaces (Sutton and Barto 2018). PG methods learn a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ using gradient descent (Sutton and Barto 2018), so are applicable in continuous action spaces. However, they suffer from high variance and instability (Sutton et al. 1999). AC methods combine PG methods with value-based methods (Konda and Tsitsiklis 2000). The actor is policy-based, enabling application in continuous action spaces. The critic learns the value function for stability.

Deceptive RL. Ornik and Topcu (2018) define a *deceptive MDP*. As agents need to maximise rewards and deceive observers, they include the observer's beliefs in the task:

Definition 2 (Deceptive MDP)**.** *A deceptive MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, r, \mathcal{B}, \mathcal{L}, \gamma)$, where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, r, and γ are the same as a regular MDP, $\mathcal{R}$ is a set of candidate reward functions, including the real reward function r and at least one fake reward function, $\mathcal{B}$ is the observer's belief-set and $\mathcal{L}(s_t, a_t, s_{t+1}, b_t)$ is a belief-induced reward function, defining the reward for executing action a_t in state s_t and transitioning to state s_{t+1} incorporating the impact of the observer's belief b_t about the agent's real reward function.*

Deceptive RL challenges. The deceptive RL objective is to maximise the belief-induced value function: $V_\pi(s) = \mathbb{E}[\sum_{t \in T} \gamma^t \mathcal{L}(s_t, a_t, s_{t+1}, b_t)]$. Compared to standard RL, there are three challenges. First, as $\mathcal{L}$ is abstract in Definition 2, we need to instantiate $\mathcal{L}$ such that it balances the trade-off between deception and reward-maximisation. Since there are two competing objectives, we can frame this as a *multi-objective RL* (MORL) problem; given an observer model and a mapping from observer beliefs to a deceptiveness score $\mathcal{D}(b_t) : \mathcal{B} \rightarrow \mathbb{R}$, we can define $\mathcal{L}$ as a vector of the two objectives: $\mathcal{L}(s_t, a_t, s_{t+1}, b_t) = \langle r(s_t, a_t, s_{t+1}), \mathcal{D}(b_t) \rangle$. This is useful as MORL offers established optimisation methods. Second, we need to model observer beliefs b_t. We can conceptualise b_t as a probability distribution where probabilities are the likelihood that each candidate reward function is real (Liu et al. 2021; Ornik and Topcu 2018; Savas, Verginis, and Topcu 2021). This transforms the challenge to an intention recognition problem. Third, we need to define a deceptiveness mapping $\mathcal{D}$. Recent approaches use simulation and dissimulation (Liu et al. 2021; Savas, Verginis, and Topcu 2021; Nichols et al. 2022).

4 Ambiguity Model

Liu et al. (2021) introduce AM for dissimulative deception in RL settings. Below we outline AM in the context of how it resolves the three deceptive RL challenges.

Modelling observer beliefs. AM models observer beliefs via cost-based plan recognition. Rather than using cost-differences to measure the divergence of behaviour from optimal, it uses *Q-differences*:

$$\Delta_{r_i}(\vec{o}) = \sum_{(s_t, a_t) \in \vec{o}} \left(Q_{r_i}(s_t, a_t) - \max_{a'_t \in A} Q_{r_i}(s_t, a'_t)\right) . \quad (1)$$

If behaviour is optimal for a reward function r_i, then $\Delta_{r_i}(\vec{o}) = 0$. Otherwise, $\Delta_{r_i}(\vec{o}) < 0$ and decreases with more sub-optimality. AM estimates probabilities for the candidate reward functions via a Boltzmann distribution:

$$P(r_i \mid \vec{o}) = \frac{\exp(\Delta_{r_i}(\vec{o}))}{\sum_{r_j \in \mathcal{R}} \exp(\Delta_{r_j}(\vec{o}))} \cdot P(r_i) ,$$

where $P(r_i)$ is an estimate of the prior probability that r_i is the real reward function. A less optimal sequence leads to a lower probability. This is a proxy for the observer beliefs.

Deceptiveness mapping. AM uses dissimulation (entropy) to map observer beliefs to deceptiveness. Dissimulation makes actions ambiguous and therefore more resilient against any observer; actions that move equally towards two reward functions makes those reward functions indistinguishable, regardless of one's recognition system. This leads to deceptive behaviour against both naïve plan recognition models and non-naïve human observers (Liu et al. 2021).

Defining $\mathcal{L}$. Instead of explicitly defining $\mathcal{L}$, AM satisfies the dual objective by maximising deceptiveness from a set of pruned actions that sufficiently satisfy the reward objective:

$$\pi(\vec{o}, s_t) = \arg\max_{a \in A^+} - \sum_{r_i \in \mathcal{R}} P(r_i \mid \vec{o} \cup (s_t, a)) \cdot \log_2 P(r_i \mid \vec{o} \cup (s_t, a)) ,$$

where $\vec{o} \cup (s_t, a)$ is the current observation sequence including (s_t, a) and A^+ is the set of actions remaining after pruning. AM prunes actions according to:

$$\text{prune}(\delta, s_t) = \{a \in \mathcal{A}(s_t) \mid Q_{\text{real}}(s_t, a) - R > \delta\} , \quad (2)$$

where $R = Q_{\text{real}}(s', a') - Q_{\text{real}}(s_0, a_0)$, Q_{real} is the Q-function for the real reward function, and (s', a') and (s_0, a_0) are the previous and initial observations respectively. Thus, R represents the *residual* expected reward received so far. As such, actions that do not increase expected future rewards beyond the threshold δ are pruned from consideration. For path planning, this ensures that AM reaches the real goal. From the remaining actions, AM minimises the information gain for the observing agent. This mirrors *multi-criteria RL*, an instance of MORL that maximises a single objective subject to the constraint that the other objectives meet a given threshold (Gábor, Kalmár, and Szepesvári 1998). Here, deceptiveness is the unconstrained objective and environment rewards is the constrained objective.

Limitations. First, AM learns Q-functions by separately pre-training a subagent for each candidate reward function. As discussed in Section 1, this is ineffective in a model-free setting due to misdirected exploration of the state space. That is, AM explores the states along the honest trajectories to each candidate reward function, rather than those visited by the final policy, leading to poor deceptive behaviour. This restricts the application of AM to domains where state space exploration is not important, such as those with known environment models. Second, AM is inapplicable in continuous action spaces. In particular, the Q-difference formulation requires a maximisation over the action space and the pruning procedure considers all actions in the action space.

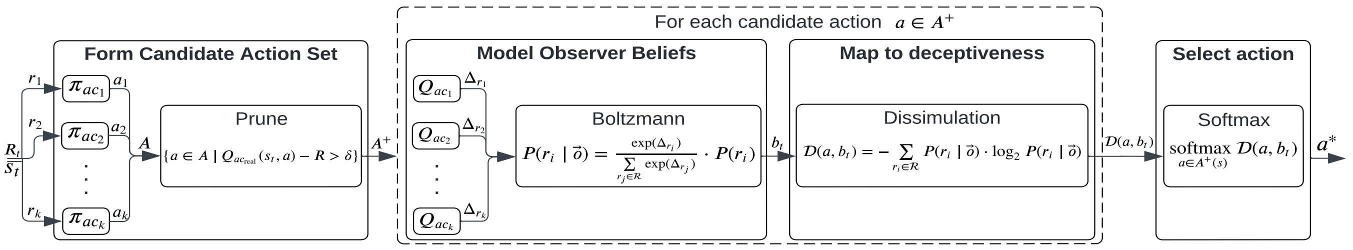

For each candidate action $a \in A^+$
Form Candidate Action Set
Model Observer Beliefs
Map to deceptiveness
Select action
R_t
S_t
r_1
r_2
r_k
π_{ac_1}
π_{ac_2}
π_{ac_k}
a_1
a_2
a_k
A
Prune
$\{a \in A \mid Q_{ac_{\text{real}}}(s_t, a) - R > \delta\}$
A^+
Q_{ac_1}
Q_{ac_2}
Q_{ac_k}
Δ_{r_1}
Δ_{r_2}
Δ_{r_k}
Boltzmann
$P(r_i \mid \vec{o}) = \frac{\exp(\Delta_{r_i})}{\sum_{r_j \in \mathcal{R}} \exp(\Delta_{r_j})} \cdot P(r_i)$
b_t
Dissimulation
$\mathcal{D}(a, b_t) = -\sum_{r_i \in \mathcal{R}} P(r_i \mid \vec{o}) \cdot \log_2 P(r_i \mid \vec{o})$
$\mathcal{D}(a, b_t)$
Softmax
$\operatorname{softmax}_{a \in A^+(s)} \mathcal{D}(a, b_t)$
a^*

Figure 2: DEAM: DEAM forms a candidate action set using AC subagents, pruning those that insufficiently satisfy rewards. It selects the action that soft-maximises the entropy of the candidate reward probabilities from the modelled observer perspective.

5 Deceptive Exploration Ambiguity Model

We introduce DEAM to address these limitations with three improvements: (1) It explores deceptive trajectories during training, improving performance in model-free settings; (2) It shares experiences between subagents, improving training efficiency; and (3) It extends to continuous action spaces.

Figure 2 shows DEAM's architecture and Algorithm 1 has the complete details. DEAM collectively trains AC subagents for each candidate reward function, using the entropy-based policy to select actions. Given a state s_t, each subagent uses its policy π_{ac_i} to submit a candidate action a_i, forming a set of candidate actions A (line 6). From these actions, DEAM prunes those that lead to insufficient rewards with the same pruning procedure as AM (Equation 2), leading to a pruned action set A^+ (line 7). For each of these actions, DEAM calculates the entropy of the reward function probabilities using an observer model (lines 11 and 12). It then selects the action a^* that soft-maximises entropy (line 14), using a temperature parameter τ which decays during training (line 18). This leads to an environment update and a reward for each candidate reward function. The experience $(s_t, a^*, s_{t+1}, r_i(s_t, a^*, s_{t+1}))$ is added to the replay buffer of each subagent (line 15) which improves policies and Q-functions at the end of each epoch (line 19). Below we outline DEAM's three improvements.

Deceptive exploration during training. Rather than separately pre-training an honest subagent for each candidate reward function, DEAM trains the subagents together, using the deceptive policy to select actions. As such, DEAM explores the deceptive part of the state space, shown by the overlap in the heavily visited states and the deceptive path in Figures 1b and 1d. Thus, it uses training time to learn a policy and Q-functions for the states visited by the final policy.

To accommodate collectively training subagents using the deceptive policy, DEAM uses a soft-maximum policy that converges to a hard-maximum policy as training progresses. During early training, Q-functions are inaccurate. When using a hard-maximum policy, this inaccuracy favours actions submitted by subagents with high magnitude Q-functions. This occurs because high magnitude Q-functions lead to relatively high Q-differences for sub-optimal actions (i.e. those submitted by the other subagents). This leads to a skewed probability distribution for these actions. Actions submitted by subagents with high Q-functions are approximately optimal for themselves, avoiding the disproportionately high Q-difference and skewed distribution. As

Algorithm 1: Deceptive Exploration Ambiguity Model

```
Require: $k$ candidate reward functions
Require: prior probability $p_i$ for each reward function
Require: soft-max temperature ($\tau$), $\tau$ decay rate ($\lambda_\tau$), and
    pruning threshold ($\delta$)
 1: $\mathcal{AC} \leftarrow \{ac_1, ..., ac_k\}$        ▷ Instantiate AC subagents and
 2: $\mathcal{D} \leftarrow \{D_1, ..., D_k\}$                ▷ Instantiate replay buffers
 3: for each training episode do
 4:     $\vec{o} \leftarrow \{\}$                 ▷ Initialise observation sequence
 5:     for each environment step ($s_t$) do
 6:         $A \leftarrow \{a_i \mid a_i \sim \pi_{ac_i}(s_t)$ for $ac_i$ in $\mathcal{AC}\}$
 7:         $A^+ \leftarrow$ prune($A$, $\delta$)
 8:         $E \leftarrow \{\}$
 9:         for $a \in A^+$ do
10:             $\vec{o_a} \leftarrow \vec{o} \cup (s_t, a)$
11:             $P \leftarrow \{p'_i | p'_i = \frac{\exp(\Delta_{r_i}(\vec{o_a}))\cdot p_i}{\sum_{r_j \in \mathcal{R}} \exp \Delta_{r_j}(\vec{o_a})}$ for $r_i \in \mathcal{R}\}$
12:             $E \leftarrow E \cup \{(a, \text{entropy}(P))\}$
13:         end for
14:         $a^* \leftarrow$ softmax($E, \tau$)
15:         $D_i \leftarrow D_i \cup \{(s_t, a^*, s_{t+1}, r_i\}$ for $D_i \in \mathcal{D}$
16:         $\vec{o} \leftarrow \vec{o} \cup (s_t, a^*)$
17:     end for
18:     $\tau \leftarrow \tau \cdot \lambda_\tau$                              ▷ Decay $\tau$
19:     Update subagents using replay-buffers in $\mathcal{D}$
20: end for
```

such, these actions tend to have higher entropy values and are selected disproportionately often. This leads to positive rewards for these subagents, compounding the problem. The soft-maximum policy selects actions with probability: $P(a_i) = \exp(e_i/\tau)/\sum_{e_j \in E} \exp(e_j/\tau)$, where e_i is the entropy value of action a_i. The temperature parameter τ starts high, leading to more randomness and therefore exploration. This allows all subagents to learn more accurate Q-values before exploiting the deceptive policy as τ decays.

Sharing experiences between subagents. As AM separately pre-trains subagents, they do not learn from the experiences sampled by the other subagents. DEAM's subagents learn from all environment interactions, even if the selected action is submitted by another subagent. That is, DEAM adds $(s_t, a^*, s_{t+1}, r_i(s_t, a^*, s_{t+1}))$ to the replay buffer of every subagent (line 15). This improves training efficiency as subagents learn from many experiences without the cost of sampling them. For subagents to learn from experiences sampled by other subagents, they must use *off-policy* RL (Sutton and Barto 2018). AM can also benefit from shared experiences. But, it does not solve the misdirected exploration that leads to poor performance in model-free domains.

Extension to continuous action spaces. We make three changes to enable application in continuous action space domains. First, we use AC subagents to learn Q-functions. The policy-based actor submits actions while the Q-learning critic calculates Q-differences, leading to probabilities for the candidate reward functions. Second, instead of using the Q-difference calculation, as per Equation 1, DEAM uses the action selected by the actor as the optimal action and the Q-function of the critic to evaluate that state-action pair. This leads to a modified Q-difference formula:

$$\Delta_{r_i}(\vec{o}) = \sum_{(s,a)\in\vec{o}} (Q_{r_i}(s,a) - Q_{r_i}(s,a^*)) \text{ where } a^* \sim \pi_i(s).$$

As the policy is stochastic, $\max_{a\in A} Q_{r_i}(s,a) \neq Q_{r_i}(s,a^*)$, even for a converged subagent. However, as $\max_{a\in A} Q_{r_i}(s,a)$ is intractable, $Q_{r_i}(s,a^*)$ is an appropriate approximation. Third, instead of selecting the optimal action from all actions in the action space, DEAM selects from a candidate action set formed using subagent policies. This supports continuous action spaces as the size of the candidate action set is equal to the number of candidate reward functions rather than the size of the action space. However, it can lead to suboptimal action selection. The subagents may not submit the optimal action to the candidate action set and therefore DEAM would not select this action. Despite the theoretical suboptimality, we show that DEAM performs as well as an optimal value iteration AM, that does not have this problem, in the discrete domain. Also, restricting the size of the candidate action set is required to extend the model to continuous action spaces.

6 Evaluation

We conduct two experiments. The first uses a passive observer to estimate a probability distribution over the candidate goals without participation in the environment. This follows Liu et al. (2021)'s experimental set-up. The second evaluates the agents using an active adversary that needs to beat the agent to the real goal. This follows Nichols et al. (2022)'s *pirate deception scenario*. Our experiments aim to assess whether DEAM addresses the limitations of AM.

Independent variables. We compare DEAM with: (1) an honest agent (moves directly to the real goal using Q_{real}); (2) an optimal, model-based AM (denoted: value iteration AM); and (3) a model-free, AC version of AM (denoted: AM). We do not know of other model-free RL agents to use as baselines. Value iteration AM is the upper bound possible for DEAM as it calculates dissimulation optimally. For the model-free AM, we add the changes that enable extension to continuous action spaces to allow evaluation in the continuous domain.

6.1 Passive Observer Experiment Design

Environments. We assess the agents in two path planning environments: one discrete and one continuous. The discrete environment supports lateral and diagonal movements. Agents receive a reward of 100 for reaching the goal, -1 for lateral actions and $-\sqrt{2}$ for diagonal actions. We use the same 40 environment settings (5 maps $\times$ 8 goal placements) as Liu et al. (2021). The maps have varied sizes and obstacles as follows: (1) 49×49 without obstacles, (2) 49×49 with few large obstacles, (3) 49×49 with many small obstacles, (4) 100×100 with large 'island' obstacles, and (5) 100×100 maze. In the continuous environment (Henderson et al. 2017), agents select a velocity between $[-1, 1]$ and direction between $[0, 2\pi]$. Rewards are based on the distance to the goal. We use the first 4 maps from the discrete environment, giving a total of 32 settings. We omitted the maze map since it was too time consuming to train even an honest agent. Value iteration AM is inapplicable in continuous domains, so it is omitted from the second environment.

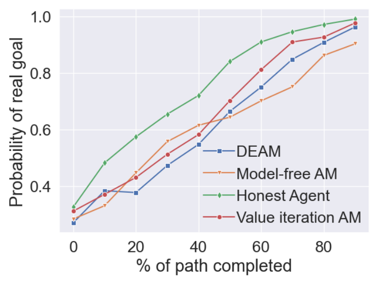
Agent implementation. We use Haarnoja et al. (2018)'s soft actor-critic (SAC) as the subagent model. Since SAC is an *off-policy* algorithm, it enables shared experiences between subagents. Hyperparameter choices are taken from the original SAC paper. We found that higher stochasticity was beneficial during the start of training and less so as training progressed. Therefore we chose $\tau = 1$ and $\lambda_\tau = 0.9$. We chose $\delta = 0$ since we found that aggressive pruning is required to reliably reach the real goal (see Section 6.5).

Measures. For deceptiveness, we measure the real goal probability at each time-step, using Masters and Sardina (2017a) intention recognition model. We also measure the number of steps taken after the last deceptive point (LDP). The LDP is the point after which the real goal is most likely from the observer's perspective (Masters and Sardina 2017b). After the LDP, a rational observer is not deceived. Thus, fewer steps after the LDP indicates better deception. For reward, we use path cost, proportional to the optimal, honest path cost. Finally, we use the number of environment steps to convergence for training efficiency.

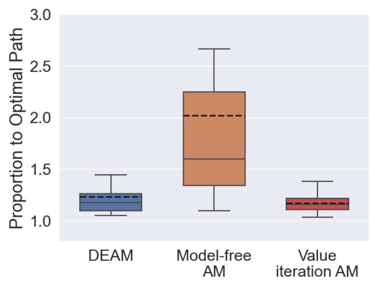
6.2 Passive Observer Experiment Results

Deceptiveness. Figure 3 shows the trade-off between the deceptiveness and environment rewards. The real goal probability increases for all agents as they progress along the path, since they move towards and reach the real goal. However, the ambiguity-based agents are consistently more deceptive than the honest agent. We calculated the average real goal probability over each trajectory for all environment settings and compared them across agents using an ANOVA test. We tested assumptions of normality and homoscedasticity using a Shapiro-Wilk and Levene's test respectively. The ANOVA test found a difference in the agent's real goal probability distributions ($p < 0.001$). We performed one-sided paired t-tests to test relative deceptiveness. Value iteration AM ($p < 0.001$), AM ($p < 0.001$) and DEAM ($p < 0.001$) all have a lower average probability than the honest agent. None of the ambiguity agents have significantly lower probabilities than each other, indicating that DEAM achieves deception as well as an optimal, model-based AM.

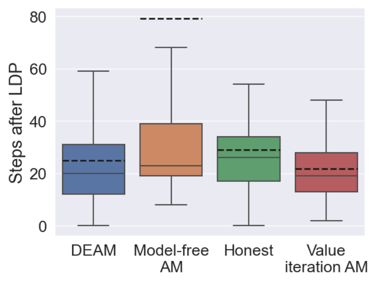
Path cost. The path costs (shown in Figures 3b and 3e) indicate that DEAM is more cost efficient than AM. AM's pre-trained subagents do not learn accurate Q-values and



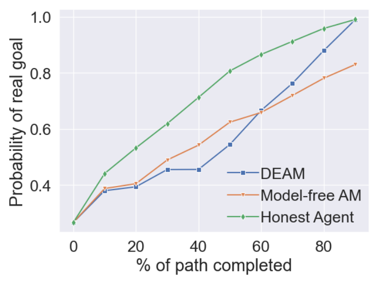
(a) Deceptiveness (discrete)



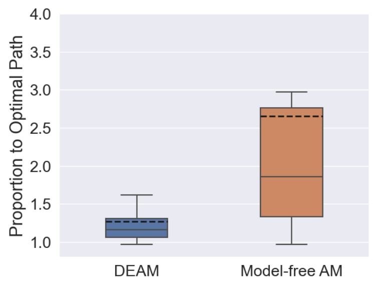
(b) Path costs (discrete)



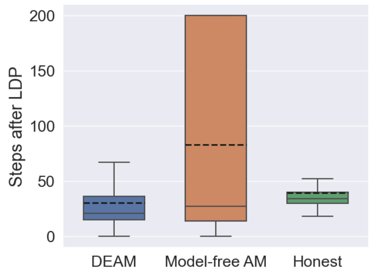
(c) Steps after LDP (discrete)



(d) Deceptiveness (continuous)

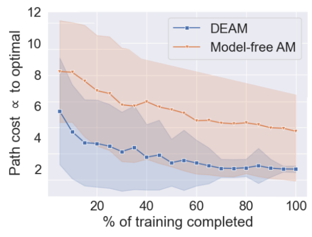


(e) Path costs (continuous)

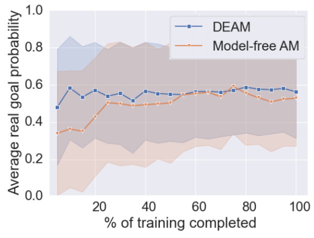


(f) Steps after LDP (continuous)

Figure 3: Performance: Deceptiveness is measured by the probability of the real goal at different percentages of path completion. Path cost is measured proportional to the optimal, honest path. In the box-plots, the solid lines are medians, dotted lines are means and shaded regions are interquartile ranges.



(a) Path cost



(b) Deceptiveness

Figure 4: Performance vs training interactions. Measures are captured over a trajectory at 20 intervals during training and averaged over all environment settings. We used $2.4m$ and $3m$ time-steps for 3 and 5 goal environments respectively. Shaded regions are a 95% confidence interval.

policies for states on the deceptive path in the allotted training time, leading to inefficient action selection. In Figure 1c, when AM moves away from the honest paths explored during training, it selects actions that are neither deceptive nor progressive towards the real goal. In other cases, AM moves into unfamiliar states and never reaches the real goal. Since we measure real goal probabilities at different percentages of path completion, it is important to consider it alongside path costs. Due to higher path costs, each path completion percentage corresponds to more steps for AM. Therefore, it travels further with a high real goal probability. This is seen in the LDP results. The average number of steps after the LDP is significantly higher for AM than the other agents. As such, observers have more time between deducing the agent's real goal and the agent arriving there.

Continuous environment. In the continuous environment, AM is the most deceptive towards the end of the trajectory. However, this is distorted by scenarios where it fails to reach the real goal, leading to comparatively low probabilities. These scenarios also lead to a high average path cost. Therefore, the apparent deceptiveness is rather a failure to consistently learn to efficiently reach the real goal. These results show DEAM's ability to learn more efficiently in complex continuous scenarios due to improved state space exploration. This suggests that DEAM successfully extends dissimulative deception to continuous action space domains.

Training efficiency. We trained DEAM and AM for $2.4m$ and $3m$ interactions for 3 and 5 goal environments respectively. At 20 intervals, we measured the average real goal probability and path cost in each setting. In Figure 4a DEAM's path costs stabilise in approximately 70% of the allotted interactions, showing convergence to a deceptive path. AM's path cost is approximately $4\times$ optimal after training. The constantly high variance indicates a failure converge to a consistent path in the allotted interactions. Figure 4b shows the average real goal probability. DEAM stabilises after completing 70% of training as it reaches a consistent path. Although AM appears more deceptive, this is distorted by scenarios where it fails to reach the real goal. DEAM balances the trade-off between deceptiveness and reward accumulation with significantly fewer environment interactions.

6.3 Active Adversary Experiment Design

We use Nichols et al. (2022)'s *pirate deception scenario* to assess the agents with an active adversary. The agent must deliver an asset to a goal location as a pirate tries to capture the agent at the goal and steal the asset. This requires deception to lure the pirate away from the real goal, and cost efficiency to prevent being overtaken by the pirate. Figure 5 shows an example of the pirate versus an honest agent and DEAM. The pirate's path is black and the agent's path is blue. The pirate determines the honest agent's real goal early, reducing the problem to a race to the real goal. In contrast, DEAM successfully deceives the pirate towards a fake goal, allowing DEAM to reach the real goal safely.

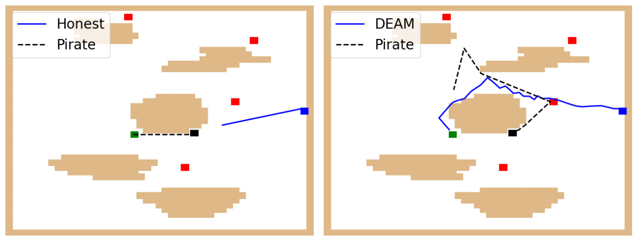


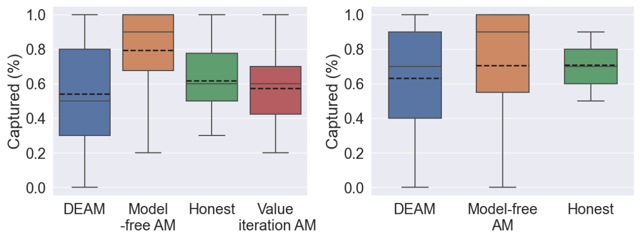
Figure 5: Pirate deception scenario. *Real goal:* Green cell. *Fake goals:* Red cells. *Pirate start:* Black cell. *Agent start:* Blue cell. *Pirate path:* Black line. *Agent path:* Blue line.

The pirate has two components: (1) An intention recognition system to estimate candidate goal probabilities; and (2) A model-based path planner that selects actions towards the most likely goal. The pirate and the agent can only take one action per time-step, with identical action spaces. We use the same environment settings, intention recognition system and agent implementation as Section 6.1. The pirate and agent are initially placed randomly. For each setting, we run 10 trials with different, random placements, totalling 720 trials. We use capture rate to evaluate performance.

6.4 Active Adversary Experiment Results

Figure 6 shows the results. In the discrete domain, DEAM performs best (mean capture rate $= 54.0\%$), followed by value iteration AM (mean capture rate $= 57.1\%$), the honest agent (mean capture rate $= 64.5\%$) and AM (mean capture rate $= 79.5\%$). These results transfer to the continuous domain. AM performs poorly due to learning inefficiencies; as it does not converge effectively, it selects actions that are not deceptive nor progressive towards the real goal. This allows the pirate to overtake it, even when initially deceived.

The performance improvement of DEAM and value iteration AM relative to the honest agent is less significant than expected. In some cases, like in Figure 5, DEAM is very effective. However, in other cases, it suffers from counter-productive deceptive actions. Deceptive actions are only effective if they significantly alter the pirate's path away from the real goal. Otherwise, they can harm performance by increasing the path cost. We identify two types of counter-productive actions. First, actions that reduce the pirate's real goal probability, but not enough to change the predicted real goal (see Figure 7). The pirate's actions matter rather than its real goal probability estimate. Deceptive actions near the end of a trajectory are often in this category as the pirate is



(a) Discrete (b) Continuous

Figure 6: Proportion of captures. For each environment setting, we used 10 random pirate placements, giving a total of 720 runs. Solid lines are medians; dashed lines are means.

already confident in the real goal. Second, actions that alter the pirate's predicted real goal to a fake goal, but the fake goal is on the path to the real goal (see Figure 8). Despite successfully deceiving the pirate into believing that a fake goal is real, the action is ineffective in sufficiently altering the pirate's path. For deceptive agents to avoid this type of counter-productive action, it needs awareness of the pirate's location, so that it can predict its path and alter it effectively. Neither DEAM nor AM have this awareness by default.

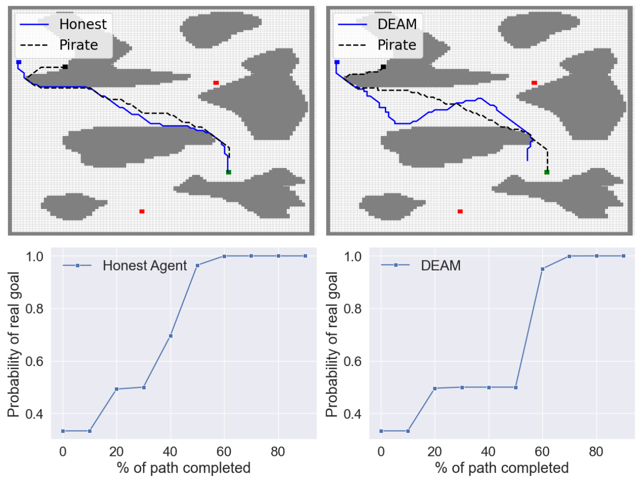


Figure 7: *Counter-productive deception 1: Top:* Agent and pirate paths. *Bottom*: Pirate probability estimate. DEAM deceives better than the honest agent, but DEAM is captured while the honest agent is not.

6.5 Hyperparameter Studies

DEAM introduces three hyperparameters: a pruning constant δ, a soft-maximum temperature τ and a soft-maximum temperature decay rate λ_τ. Here, we investigate the impact of these hyperparameters on performance. We use all 72 environment settings for each hyperparameter combination. Unless otherwise stated, hyperparameters are set to the values in Table 1 in the Appendix. Figure 9 shows the results.

Pruning constant δ. High magnitude δ values have lower real goal probabilities and higher path costs, suggesting that

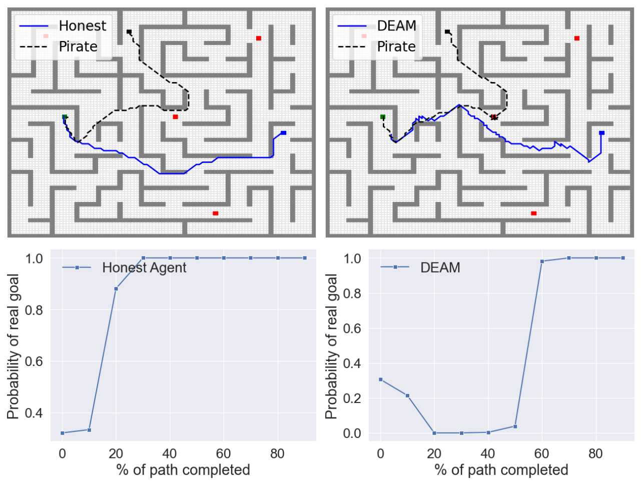


Figure 8: *Counter-productive deception 2:* The deceptive fake goal is along the pirate's path to the real goal. Therefore, DEAM is captured while the honest agent is not.

δ controls the trade-off between deceptiveness and rewards; when pruning is more aggressive ($\delta \rightarrow 0$) the agent moves more directly to the real goal. However, when δ is high, DEAM moves ambiguously between candidate goals without reaching the real goal. Therefore, it maintains a low real goal probability throughout the trajectory and has a high path cost. This is undesirable as DEAM only achieves deceptiveness while neglecting rewards. Aggressive pruning is needed to ensure that DEAM reliably reaches the real goal.

Soft-maximum hyperparameters τ and λ_τ. τ controls the stochasticity in the soft-maximum policy; when $\tau \rightarrow 0$, the soft-maximum policy converges to a hard-maximum policy. When τ starts low or decays quickly, DEAM's behaviour becomes unstable, resulting in high path costs. The added stochasticity in the soft-maximum policy during early training stages enables sufficient exploration of all candidate reward functions, allowing the subagents to learn more accurate Q-values before exploiting the deceptive policy as τ decays. Therefore, a high initial τ value that decays during training is beneficial for performance. The results also show that decaying τ slower than $\lambda_\tau = 0.75$ does not improve performance, indicating that it is not important to maintain high stochasticity in the policy throughout training. This is due to the subagents ability to quickly learn accurate Q-values.

7 Discussion and Broader Impact

We introduce DEAM, a dissimulative deceptive RL agent that explores deceptive trajectories during training for application in model-free domains, extends to continuous action spaces and improves training efficiency. Our results show that DEAM achieves similar performance to an optimal value iteration AM, and extends to a continuous action space environment. Compared to AM, DEAM explores the states that are visited by the deceptive policy, improving deceptiveness, path-costs, and training efficiency.

This study has several limitations. First, DEAM is tied to dissimulative deception, but there are other ways to deceive, such as simulation. Second, we assume that the observer is naïve, so is unaware that it is being deceived. Third, we only investigate fully observable environments for the agent and the observer. Finally, we evaluated only on path planning environments. In future work, we aim to address these limitations by including a more sophisticated observer model and to extend deceptive RL to new types of domains.

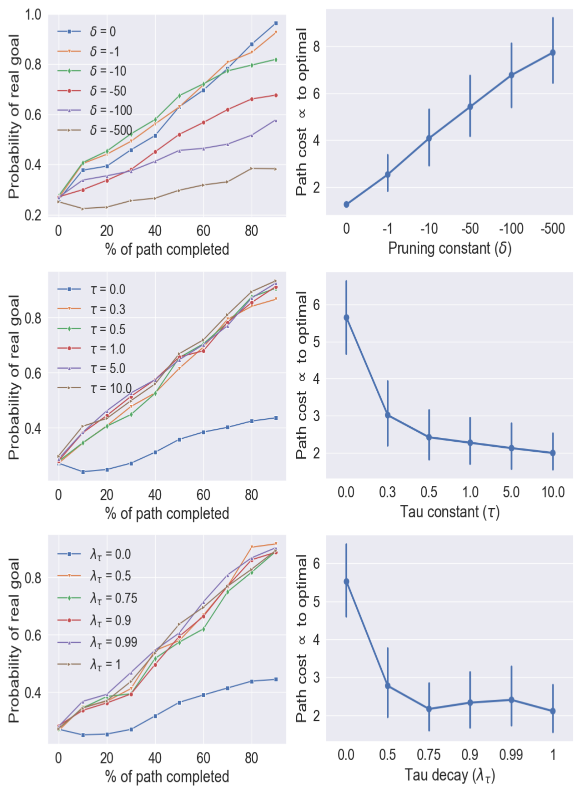


Figure 9: DEAM's sensitivity to hyperparameters: pruning constant δ (top), soft-maximum temperature τ (middle) and decay rate λ_τ (bottom). Left: average real goal probability over the trajectories. Right: average path costs with 95% CI.

Deceptive AI is a sensitive topic that requires ethical considerations (Boden et al. 2017; Danaher 2020). Boden et al. (2017) argue that robots "should not be designed in a deceptive way to exploit vulnerable users". A key distinction is the part of the state that is distorted. Deception about the agent's external world is problematic as it distorts the target's perception of reality that exists independent of the deceptive agent (Danaher 2020). However, deception about the agent's internal state depends on the motivation (Danaher 2020). DEAM deceives observers about its internal state (reward function) for privacy. This is an important part of responsible AI with positive applications, such as cyber security and enhancing computer-human interaction (Adar, Tan, and Teevan 2013; Dignum 2019; Sarkadi et al. 2019). Nevertheless, dissimulative deception may be used by malicious agents to deceive vulnerable users. We urge users of our research to consider the ethical implications prior to development.

References

Adar, E.; Tan, D. S.; and Teevan, J. 2013. Benevolent deception in human computer interaction. In *Proceedings of the SIGCHI conference on human factors in computing systems*, 1863–1872.

Bell, J. B. 2003. Toward a Theory of Deception. *International Journal of Intelligence and Counterintelligence*, 16(2): 244–279.

Boden, M.; Bryson, J.; Caldwell, D.; Dautenhahn, K.; Edwards, L.; Kember, S.; Newman, P.; Parry, V.; Pegman, G.; Rodden, T.; et al. 2017. Principles of robotics: regulating robots in the real world. *Connection Science*, 29(2): 124–129.

Danaher, J. 2020. Robot Betrayal: a guide to the ethics of robotic deception. *Ethics and Information Technology*, 22(2): 117–128.

Dignum, V. 2019. *Responsible artificial intelligence: how to develop and use AI in a responsible way*. Springer Nature.

Gábor, Z.; Kalmár, Z.; and Szepesvári, C. 1998. Multi-criteria reinforcement learning. In *International Conference on Machine Learning*, volume 98, 197–205. Citeseer.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 1861–1870. PMLR.

Henderson, P.; Chang, W.-D.; Shkurti, F.; Hansen, J.; Meger, D.; and Dudek, G. 2017. Benchmark Environments for Multitask Learning in Continuous Domains. *ICML Lifelong Learning: A Reinforcement Learning Approach Workshop*.

Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; et al. 2021. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873): 583–589.

Konda, V. R.; and Tsitsiklis, J. N. 2000. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, 1008–1014. Citeseer.

Kulkarni, A.; Klenk, M.; Rane, S.; and Soroush, H. 2018. Resource Bounded Secure Goal Obfuscation. In *AAAI Fall Symposium on Integrating Planning, Diagnosis and Causal Reasoning*.

Liu, Z.; Yang, Y.; Miller, T.; and Masters, P. 2021. Deceptive Reinforcement Learning for Privacy-Preserving Planning. *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*.

Masters, P.; and Sardina, S. 2017a. Cost-based goal recognition for path-planning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 750–758.

Masters, P.; and Sardina, S. 2017b. Deceptive Path-Planning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 4368–4375.

Masters, P.; and Sardina, S. 2019. Goal recognition for rational and irrational agents. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 440–448.

Ng, A. Y.; and Russell, S. 2000. Algorithms for Inverse Reinforcement Learning. In *Proceedings 17th International Conference on Machine Learning*. Citeseer.

Nichols, H.; Jimenez, M.; Goddard, Z.; Sparapany, M.; Boots, B.; and Mazumdar, A. 2022. Adversarial Sampling-Based Motion Planning. *IEEE Robotics and Automation Letters*.

Ornik, M.; and Topcu, U. 2018. Deception in optimal control. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 821–828. IEEE.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Ramírez, M.; and Geffner, H. 2009. Plan recognition as planning. In *25th International Joint Conference on Artificial Intelligence*.

Sarkadi, Ş.; Panisson, A. R.; Bordini, R. H.; McBurney, P.; Parsons, S.; and Chapman, M. 2019. Modelling deception using theory of mind in multi-agent systems. *AI Communications*, 32(4): 287–302.

Savas, Y.; Verginis, C. K.; and Topcu, U. 2021. Deceptive Decision-Making Under Uncertainty. *arXiv preprint arXiv:2109.06740*.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423.

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

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; Mansour, Y.; et al. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 99, 1057–1063. Citeseer.

Whaley, B. 1982. Toward a general theory of deception. *The Journal of Strategic Studies*, 5(1): 178–192.