Zero-Shot Assistance in Sequential Decision Problems

Sebastiaan De Peuter[1], **Samuel Kaski**[1,2]

[1] Department of Computer Science, Aalto University, Espoo, Finland
[2] Department of Computer Science, University of Manchester, Manchester, UK
sebastiaan.depeuter@aalto.fi, samuel.kaski@aalto.fi

Abstract

We consider the problem of creating assistants that can help agents solve new sequential decision problems, assuming the agent is not able to specify the reward function explicitly to the assistant. Instead of acting in place of the agent as in current automation-based approaches, we give the assistant an advisory role and keep the agent in the loop as the main decision maker. The difficulty is that we must account for potential biases of the agent which may cause it to seemingly irrationally reject advice. To do this we introduce a novel formalization of assistance that models these biases, allowing the assistant to infer and adapt to them. We then introduce a new method for planning the assistant's actions which can scale to large decision making problems. We show experimentally that our approach adapts to these agent biases, and results in higher cumulative reward for the agent than automation-based alternatives. Lastly, we show that an approach combining advice and automation outperforms advice alone at the cost of losing some safety guarantees.

Introduction

In this paper we consider the problem of assisting agents in tackling sequential decision problems which they have never encountered before. Human decision makers are routinely faced with this problem. Take for example engineering design (Rao 2019), where one looks to find or construct the best possible design within a space of designs that are feasible. Every design problem is new: each time an architect builds a house it is for different clients. Although the problem is novel to the agent, we can assume that it already knows how to solve it in principle, though not optimally. An architect does not need to re-learn architecture when designing a new house, but can apply their general knowledge and experience to this new design problem.

These design problems can be thought of as single-episode decision problems: they consist of a sequence of decisions, each changing or elaborating a design in some way. Once a satisfactory design has been found, the episode terminates. The decisions are driven by a goal, which can be encoded as a reward function, known only to the agent. This goal is usually tacit and complex, meaning that the agent is unable to provide an accurate explicit description of it.

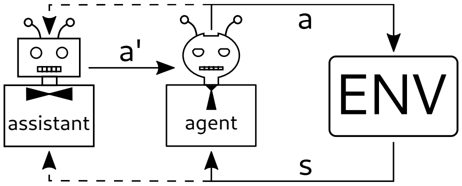


Figure 1: In *zero-shot assistance* an assistant helps an agent solve a problem without initially knowing the agent's reward function. We propose an assistant which helps the agent primarily by advising it, leaving the agent in direct control of its environment. In every time step the assistant gives new advice a', appropriate for the current state, based on its inference of the agent's reward function and potential biases. When acting the agent incorporates the advice into its own decision making. The assistant observes both the action a taken by the agent and the new state of the environment s, and uses this to infer the agent's reward function and biases.

We seek to create *assistants* which can assist agents in solving these types of decision problems. Although the assistant is technically an agent, to avoid confusion we will always refer to it as the *assistant* and will only use the term *agent* to refer to the agent being assisted. We think of the agent as an online decision maker who can be assisted in their decision making. The goal for the assistant is to increase the quality of the agent's decisions, measured by cumulative reward, relative to the agent's effort. There are two things the assistant does not know a-priori about the agent: the agent's reward function, and any biases that may cause the agent to deviate from optimal behaviour. Although the agent knows its reward function, it has never solved its problem before – ruling out inferring the reward function from prior observations – and is not able to provide an explicit description of the reward function. Similarly, biases are not something an agent is aware of, and thus are not something it can communicate to the assistant. To assist, the assistant must infer both during the episode. Therefore, we call this *zero-shot assistance*. Zero-shot assistance is closely related to zero-shot coordination (Hu et al. 2020), though in the latter the reward function is known to all participants.

We introduce *AI-Advised Decision-making* (AIAD). In AIAD an assistant helps an agent primarily by giving advice, while the agent remains responsible for taking actions in the environment. Figure 1 shows the interaction between the agent and the assistant. The advice is based on a reward function that is inferred from the agent's behaviour. For simplicity we will focus on advice of the type "have you considered doing a", where a is an action, though AIAD readily generalizes to other forms of advice. We see two fundamental advantages in having an assistant that advises. First, taking advice into account takes little effort while it can be really helpful. Bad advice can be swiftly rejected. Second, it keeps the agent firmly in control and able to reject advice that would have a negative impact. This is a minimum requirement in applications where safety is a concern.

Choosing advice to give to a biased agent requires the assistant to account for those biases. Biases can cause apparently irrational behaviour, including the rejection of useful advice. We consider these biases to include innate limitations or constraints within the agent's decision making process, incorrect problem understanding, or limited knowledge. Incorrect problem understanding, for example, has been shown to cause humans to reject advice which rationally is in their interest (Elmalech et al. 2015). More generally, research in psychology starting in the 1970s has shown that humans exhibit a number of cognitive biases, caused by various heuristics they employ in their decision making, which cause a deviation from optimal behaviour (Kahneman et al. 1982; Ho and Griffiths 2022). Accounting for specific manifestations of these biases will be especially important when assisting human agents. Prior work on assistance has been able to incorporate agent biases that were known *a priori* (Fern et al. 2014; Hadfield-Menell et al. 2016; Shah et al. 2020). However, if they are not – as is the case in zero-shot assistance – they must be inferred online. To address this we model the uncertainty over biases explicitly, allowing the agent to maintain beliefs over which biases an agent has, and to incorporate present and future beliefs into its planning.

Contributions

In this paper we formalize the assistant's problem of advising agents with unknown reward function and biases as a decision problem. We propose a planning algorithm, a variant of Monte Carlo Tree search (MCTS), for finding the assistant's policy. To evaluate the practicality of AI-advised decision-making we introduce two decision problems: planning a day trip and managing an inventory of products with stochastic demand. A popular baseline approach for reducing agent effort and improving decision making is to automate, by leaving the decision making entirely to an assistant. When no reward function is available, prior work has proposed to first elicit the reward function (Ng and Russell 2000; Wirth et al. 2017), and then automate. In simulation experiments we show that **(1)** AIAD significantly outperforms these automation-based baselines. We implement two versions of AIAD: a standard version which only makes recommendations and a hybrid form which has direct access to the decision problem and can therefore automate as well at the cost of agent control. We also show that **(2)** an assistant which infers and accounts for agent biases outperforms one that does not.

Related Work

Learning reward functions from others The main alternative to our proposed approach of advising agents is to take decisions in their place, i.e. automation. For this, the reward function must be known. Thus, before automating one needs to elicit or learn the reward function from the agent. Inverse Reinforcement Learning (IRL) proposes to learn a reward function directly from observing the agent act (Ng and Russell 2000; Abbeel and Ng 2004; Ramachandran and Amir 2007; Arora and Doshi 2021). In preference-based elicitation (Wirth et al. 2017; Christiano et al. 2017; Brown et al. 2019), the agent is asked which of two trajectories or individual decisions it prefers. The agent is assumed to prefer the trajectory with the highest reward. An alternative is to ask the agent for direct feedback on a single trajectory of decisions (Knox and Stone 2009; Warnell et al. 2018). A subset of this literature has looked specifically into the feasibility and utility of learning both agents' reward functions and biases (Evans and Goodman 2015; Evans, Stuhlmueller, and Goodman 2016). Chan, Critch, and Dragan (2021) found that biases can make agents' behaviour more informative of their reward function, and that incorrectly modeling biases can result in poor reward inference. Armstrong and Mindermann (2018), however, show that jointly identifying biases and reward from observations is not always possible. Shah et al. (2019) investigate under what assumptions biases can be learnt purely from data.

Where applicable, inference and automation happen in two distinct phases in these works; the elicitation process is not informed by the immediate needs of automation. In an assistance method like ours, both happen at the same time, allowing the assistant to reduce its uncertainty with regards to the agent's biases and reward where it matters for the decisions it needs to make (Shah et al. 2020). We note also that under reward uncertainty, an automating policy must necessarily be more risk-averse than an assistant that gives advice, as the automating policy cannot rely on the agent to prevent it from making bad decisions.

Human-AI collaboration Our work fits within a larger body of approaches which consider collaboration between an assistant and an agent to solve a common problem. Dimitrakakis et al. (2017) consider a setting in which an assistant acts autonomously but can be overridden by an agent at a cost. Çelikok, Oliehoek, and Kaski (2022) consider a similar problem in partially observable environments. Both, however, assume the assistant already knows the reward function. Others have considered collaboration when the assistant does not know the agent's reward function. Shared autonomy (Javdani, Srinivasa, and Bagnell 2015; Reddy, Dragan, and Levine 2018) considers a setting in which the agent gives commands to the assistant, which then acts in the environment. As the assistant does not necessarily follow the commands directly, but uses them to infer the agent's reward function which it then maximizes, we consider this an automating approach. In Cooperative Inverse Reinforcement Learning (Hadfield-Menell et al. 2016) an assistant and agent jointly solve a problem. The assistant uses IRL to learn the agent's reward function. Fern et al. (2014) has proposed assistants which assist agents (not necessarily through advice) in decision problems; Shah et al. (2020) proposed similar assistants for partially observable settings. These last three works are similar to ours but assume that, except for the reward function, everything that determines the agent's policy – including any potential biases – is known *a priori*. Unlike our method, these methods not support the online inference of biases needed for zero-shot assistance.

Problem Setup

We consider an agent solving a decision problem which can be modeled as an infinite-horizon MDP $E = \langle \mathcal{S}, \mathcal{A}, T, \mathcal{R}_\omega, \gamma, p_{0,s} \rangle$. Here $\mathcal{S}$ is a set of states and $\mathcal{A}$ is the set of actions available to the agent. At time step t the transition function $T(s_{t+1} \mid s_t, a_t)$ defines a distribution of potential next states s_{t+1} given that the agent has taken action a_t in current state s_t. $\mathcal{R}_\omega(s_t, a_t, s_{t+1})$ is the reward function. It defines the instantaneous reward for taking action a_t in state s_t and ending up in s_{t+1}. $\gamma \in (0, 1]$ is the discounting rate. The agent's objective is to maximize its expected discounted cumulative reward $\mathbb{E}\left[\sum_{t=0}^{\infty} r_t \gamma^t \mid T, p_{0,s}\right]$ where r_t is the reward it achieved at time step t. Finally, $p_{0,s}$ is the start state distribution: $s_0 \sim p_{0,s}$.

When assisting an agent we will assume that we know certain things about that agent's problem E. In line with prior work (Abbeel and Ng 2004) we assume that we know $\mathcal{S}$, $\mathcal{A}$, T, γ and $p_{0,s}$. Though we do not know $\mathcal{R}_\omega$, we do have access to its parametric function class $\mathcal{R} = \{\mathcal{R}_{\omega'}\}_{\omega' \in \Omega}$. Note that this assumption is not particularly restrictive; $\mathcal{R}$ could be the space of all reward functions.

AI-Advised Decision-Making (AIAD)

We now formalize AI-Advised Decision-making. As shown in Figure 1, the agent acts in E based on advice from the assistant. The goal of the assistant is to maximize the cumulative discounted reward obtained by the agent through this advice. We define this from the assistant's point of view as a decision problem with reward function $\mathcal{R}_\omega(s_t, a_t, s_{t+1})$ and as actions the advice it can give.

For the assistant to be able to plan, we will assume we have an *agent model* $\hat{\pi}(a \mid s, a'; \theta, \omega)$ available, a model of the agent's fixed policy upon receiving advice a'. This could be an expert-created model based on appropriate assumptions, or could have been learned based on observed agent behaviour on similar problems. It depends on two sets of unobserved parameters: $\omega \in \Omega$ and $\theta \in \Theta$. We defined Ω earlier as the parameter space of the reward function. Θ is the parameter space for the possible biases the agent has. We call θ the bias parameters.

Advice as a Decision Problem

We define the assistant's decision problem as a generalized hidden parameter MDP (GHP-MDP) $\mathcal{M}$ (Perez, Such, and Karaletsos 2020). A GHP-MDP is an MDP in which both the transition and reward function are parameterized but where the true values of those parameters are not observed. In our definition these parameters are ω and θ, the two unobserved parameters which determine an agent's reward function and biases. We define $\mathcal{M}$ such that the instance $\mathcal{M}_{\omega,\theta}$ defines the problem of assisting an agent with reward parameters ω and bias parameters θ.

Let $\mathcal{M} = \langle \mathcal{S}, \Omega, \Theta, \mathcal{A}', \mathcal{A}, \mathcal{T}, \hat{\pi}, \mathcal{R}, \gamma, p_{0,s}, p_{0,\omega}, p_{0,\theta} \rangle$. Here $\mathcal{S}, \mathcal{A}, \gamma$, and $p_{0,s}$ are the same as in the agent's problem E. $\mathcal{R}$ and $\hat{\pi}(a \mid s, a'; \theta, \omega)$ are as defined earlier. The assistant's actions $\mathcal{A}'$ constitute advice that can be given to the agent. Ω and Θ are the parameter spaces for the reward function and biases, and $p_{0,\omega}$ and $p_{0,\theta}$ are prior distributions over them. Lastly, $\mathcal{T}$ is a collection of transition functions $\mathcal{T}_{\omega,\theta}$ for all possible values of $\omega \in \Omega$ and $\theta \in \Theta$. It encodes the interaction between the assistant and the agent and between the agent and the environment from Figure 1.

When the assistant gives advice $a'_t \in \mathcal{A}'$ to the agent, the agent is free to choose which action $a_t \in \mathcal{A}$ to take in E. It is this action taken by the agent that leads to a new state, according to the transition function of E. Thus, the assistant only indirectly influences the change of state, by using advice to induce a different policy from the agent. The agent model $\hat{\pi}$ predicts which policy will be induced by advice. Thus, for given reward and bias parameters ω and θ, the transition function is

$$\mathcal{T}_{\omega,\theta}(s_{t+1} \mid s_t, a'_t) = \sum_{a_t \in \mathcal{A}} \hat{\pi}(a_t \mid s_t, a'_t; \theta, \omega) T(s_{t+1} \mid s_t, a_t)$$

Because the true values of θ and ω are not known, we can think of $\mathcal{M}$ as defining a space of MDPs. The challenge in planning over $\mathcal{M}$ is that planning must happen without knowing which MDP in this space the assistant is truly operating in, i.e. what kind of agent the assistant is advising. We can, however, maintain beliefs over θ and ω based on the transitions we observe. For every transition (s_t, a'_t, s_{t+1}) we observe we can calculate the likelihood of that transition under the various possible parameter values in $\Omega \times \Theta$ to update our posterior belief distributions over the parameters. In other words, by observing the agent's decisions in response to advice we can maintain beliefs about the agent's biases and reward function.

Root Sampling for GHP-MDPs

Finding an optimal policy over $\mathcal{M}$ involves not only planning on a belief distribution over MDPs, but also accounting for how that distribution will change as we act. With every action we take, we observe a new transition which will change our beliefs over Ω and Θ. However, not every action is equally informative: advice which gets wildly different reactions from different types of agents will be more useful for determining what type of agent we are assisting than advice to which all agents react in the same way. Planning must consider both the expected long-term reward of actions, and their informativeness towards the unknown parameters.

We propose a modification of *Bayes-adaptive Monte Carlo Planning* (BAMCP) (Guez, Silver, and Dayan 2012). BAMCP is based on MCTS (Browne et al. 2012) and enables planning over MDPs where the transition function is not known, and needs to be inferred from transition observations. The advantage of BAMCP is that it efficiently maintains all current and potential future beliefs over transition functions by using the tree as a particle filter (Guez, Silver, and Dayan 2012). This allows it to incorporate the future information value of actions into its value estimates. We extend this algorithm from operating on beliefs over transition functions to joint beliefs over transition and reward functions, i.e. beliefs over Θ and Ω. We call this new variant *Generalized Hidden Parameter Monte Carlo Planning* (GHPMCP).

We give a short overview of the algorithm here, and refer the reader to the appendices for a detailed explanation. Like any MCTS algorithm, in every planning iteration GHPMCP simulates an MDP down the tree following a UCT policy. The main difference is that the simulated MDP is resampled for every iteration. Before an iteration starts, parameters θ, ω are sampled from p_θ, p_ω. $\mathcal{M}_{\omega,\theta}$ is then simulated down the tree. The Q-function estimates along the path are updated using $\mathcal{M}_{\omega,\theta}$'s specific reward function R_ω. Here lies the difference to BAMCP, which – because it only considers uncertainty over the transition function – uses the same fixed reward function R in every iteration.

An Agent Model for Assistance

We now introduce a general-purpose agent model, applicable to any decision problem E as defined above. We have developed this agent model to be a good starting point for most use cases. It is based on established and grounded theories of human decision making. It is also consistent with how RL agent policies are often implemented. We use instances of it in our experiments here, but stress that our proposed method does not require this agent model specifically.

The proposed agent model is built on the following choice rule:

$$p(a|u) = \frac{p(a)\exp\left(\beta u(a)\right)}{\sum_{\hat{a}\in\mathcal{A}} p(\hat{a})\exp\left(\beta u(\hat{a})\right)} \qquad (1)$$

where $a \in \mathcal{A}$ is an action, u is a function that assigns a utility to every action, $p(a)$ is a prior distribution over actions, and β is a temperature parameter. β allows us to interpolate between fully rational choices ($\beta = \infty$) and fully random choices ($\beta = 0$). This choice rule has repeatedly proven to be a useful and practical model of human cognition (Lucas et al. 2008; Baker, Saxe, and Tenenbaum 2009; Viappiani and Boutilier 2010; Christiano et al. 2017; Brown et al. 2019). Theoretical work has proposed it as a result of a bounded rational view of human cognition with information processing costs (Ortega and Braun 2013; Genewein et al. 2015). Surprisingly, this rule is also a popular choice of model in RL – where it is known as a softmax policy – for representing agent policies (Sutton and Barto 2018).

The agent model $\hat{\pi}(a \mid s, a'; \theta, \omega)$ consists of a sequence of choices made according to this choice rule. It is based on the idea that an agent will only settle for the assistant's action if it cannot find a better one itself. As utility function we use the Q-function of the current state $u(a) = \hat{Q}(s, a; \theta, \omega)$. This Q-function could be derived directly from the agent's problem E, or from some derivative of it $\hat{E}$ in case we want to model an agent with an incorrect or limited view of the world. Some biases can therefore be modeled as part of $\hat{E}$. $\hat{Q}$ then depends on ω to allow us to model different reward functions, and on θ so that we can use the bias parameters as parameters of the problem model $\hat{E}$.

Under our model the agent starts by choosing the best action it can think of. This is a stochastic choice which we represent as a random variable A_1 defined over the actions. The distribution of A_1 results from a straightforward application of equation (1):

$$p_{A_1}(a) = \frac{p(a)\exp\left(\beta_1 \hat{Q}(a)\right)}{\sum_{\hat{a}\in\mathcal{A}} p(\hat{a})\exp\left(\beta_1 \hat{Q}(\hat{a})\right)} \qquad (2)$$

As the state and parameters are constant within this context, we have used the shorthand $\hat{Q}(a) := \hat{Q}(s, a; \theta, \omega)$ and $p(A_1 = a) := p(A_1 = a|s; \theta, \omega)$. We will drop the conditioning variables s, θ and ω for the rest of this section.

Next the agent chooses whether to switch to the assistant's recommended action a' or to stick to the action a it has chosen. The probability of switching from a to a' (denoted $a \to a'$) is

$$p\left(a \to a'\right) = \frac{\exp\left(\beta_2\left(\hat{Q}(a') - \hat{Q}(a)\right)\right)}{1 + \exp\left(\beta_2\left(\hat{Q}(a') - \hat{Q}(a)\right)\right)} \qquad (3)$$

This probability is the result of applying equation (1) to the binary choice of switching or not with a uniform prior. The utility for both choices is the gain in Q-value realised: $\hat{Q}(a') - \hat{Q}(a)$ in the case of switching and 0 otherwise. As this choice is easier than the choice in equation (2) we use a different temperature parameter β_2 here. We represent the agent's choice of action after considering a' by A_2, a random variable with distribution

$$p_{A_2}(a|a') = \begin{cases} [1 - p(a \to a')]p_{A_1}(a) & \text{if } a \neq a' \\ [1 - p(a' \to a')]p_{A_1}(a') & \\ \quad + \sum_{a''\in\mathcal{A}} p(a'' \to a')p_{A_1}(a'') & \text{if } a = a' \end{cases}$$

This distribution is then the agent model's policy $\hat{\pi}$.

In some problems, such as design problems, there is a special NOOP action which allows the agent to choose to do nothing. We model this as the agent recommending the NOOP action to itself, after arriving at A_2, analogously to the switch to the assistant's recommendation a'.

Experiments

We present here results from a number of simulation experiments on two single-episode decision problems: a day trip design problem and an inventory management problem[1] Both exhibit a very different structure. The day trip design problem has vast state and action spaces that represent a design that evolves over time. Actions do not have an inherent cost, but rather the value of the design produced at the end of the episode is important. Returning to the start state (the starting design) without restarting the episode is trivial. The inventory management problem on the other hand has a

[1] Our code is available from https://github.com/AaltoPML/Zero-Shot-Assistance-in-Sequential-Decision-Problems. The appendices can be found at https://arxiv.org/abs/2202.07364.

smaller state space but all actions contribute to the cumulative reward of the episode, and it is generally impossible to reset the problem back to its starting point within an episode.

We will compare agents assisted by AIAD to agents assisted by a number of baselines on these two decision problems. We consider two versions of AIAD: **AIAD** and **AIAD + automation**. The latter is an extension which gives the assistant actions that directly change the environment. These actions have transition function $\mathcal{T}_{\omega,\theta}(s_{t+1} \mid s_t, a'_t) = T(s_{t+1} \mid s_t, a'_t)$. We consider four baselines: **(1) unassisted agent** To create an unassisted agent we modify our agent model by removing the switch to a recommended action encoded in eq. (3) ($p(a \to a') = 0 \ \forall a \in \mathcal{A}$). **(2) IRL + automation**. This is an IRL-based approach following prior work in learning rewards from biased agents. It observes N time steps of the unassisted agent acting without assistance. It then infers both θ and ω from the observations, using the same agent model but without knowledge of the parameters, and completes the rest of the episode in place of the agent. For day trip design we first return back to the starting design before automating. The automation policy is an optimal policy for the agent's problem E with as reward function the expected reward under the inferred posterior over reward parameters. **(3) PL + automation**. This approach is based on preference learning (PL). At the start of the episode the agent is presented with N comparison queries. These queries are selected based on their expected information gain. We create an agent model that chooses between the two options according to the choice rule from eq. (1) with $u(s) = f_\omega(s)$. This model is both used to simulate the agent and to infer the reward function from the agent's responses. We then automate all decisions within the episode based on the inferred reward function. **(4) partial automation** This method is a more flexible version of IRL + automation. It automates by default, but in any time step can temporarily hand control back to the agent. The agent then acts – without advice – in that time step. This allows partial automation to rely on the agent when it is too uncertain about how to act. The agent's observed action is used to update the beliefs about θ and ω. This baseline is representative for approaches such as (Shah et al. 2020; Hadfield-Menell et al. 2016), albeit reimplemented here within our framework so that biases can be inferred.

Every run of our experiments lasts for one episode. Posterior beliefs in all implementations are maintained using a weighted particle filter. Within a run every method is applied once to assist a simulated agent in an instance of the problem considered. The cumulative reward obtained through different assistance methods is then compared using a paired Wilcoxon signed rank test at significance level $p < 0.05$.

Day Trip Design

The day trip design problem is an idealized but otherwise realistic instance of a design problem. Like most other design problems this has a large action space and a vast state space. The agent is given 100 points of interest (POI) and must choose a subset of them which it wants to visit within a day. Its goal is to choose a subset that it would maximally enjoy visiting. Every POI has a location, visit duration, admission cost, and belongs to a number of topics. There are 20 topics in total. The enjoyment of visiting a POI is a function of the overlap between the topics to which it belongs and the topics the agent is interested in. The total enjoyment of visiting the POIs must be traded off against the sum of their admission costs. The value of a trip s, $f_\omega(s)$, is therefore a combination of these two scores, parameterized by parameters ω which capture the agent's topic interests and tolerance for high admission prices. Choosing POIs involves accounting for the time needed to travel between the chosen points. Any time spend walking cannot be spend enjoying a visit to a POI. To help with this, the agent is automatically given an optimal itinerary which minimizes the travel time for its current selection of POIs (this involves solving a traveling salesperson problem (TSP)).

To formalize this as an MDP E we define the state space as the space of all subsets of the POIs. We introduce an action for every POI which allows the agent to add that POI to the current trip or to remove it, depending on whether it is already part of it or not. To enforce the constraint that all selected POIs must be visited within a day, in states corresponding to a trip that would take more than 12 hours we only allow actions that remove POIs. The agent seeks to maximize $f_\omega(s)$. Therefore, we define its reward function as the improvement in objective value from one time step to the next: $R_\omega(s_t, a_t, s_{t+1}) = f_\omega(s_{t+1}) - f_\omega(s_t)$. Additional details about this experiment can be found in the appendices.

We use the agent model we introduced in the previous section. The problem $\hat{E}$ on which the agent plans differs in two key aspect from E. The first is that the agent uses a visual heuristic to determine how the itinerary for its current choice of POIs will change as it considers additions and deletions. We model this using a visual heuristic for TSPs commonly used by humans (MacGregor, Ormerod, and Chronicle 2000). The second difference is that we introduce an anchoring bias. This bias, which is typical in human designers, causes the agent to resist large changes to its design. In our implementation, agents with this bias will refuse to consider adding any POI that is more than 500 meters away from their current itinerary. Concretely, this means that in a state s in $\hat{E}$, the available actions will only be those that add POIs which are within 500 meters of the itinerary, or that remove POIs. The Q-values used in the agent model are determined using depth-limited best-first search on $\hat{E}$.

Because this is a design problem, the cost of taking actions – i.e. changing the design – is non-existent. We are therefore mainly interested in minimizing the effort required from the agent; how many actions the assistant takes is not a significant factor. Thus, our quantity of interest is the objective value $f_\omega(s)$ achieved as a function of the number of agent interactions N. Due to our definition of the reward function, this is equal here to the undiscounted cumulative reward after N interactions. As interactions we count both actions taken and queries answered by the agent. For PL, queries consist of of a comparisons between two day trips (i.e. states). We have created a separate agent model that chooses between the two day trips according to the choice rule from eq. (1) with $u(s) = f_\omega(s)$. The PL and IRL baselines are evaluated at 0, 5, 10 15, 20, 25 and 30 interactions, while the other methods are evaluated on a continuous range of N from 1 to 30. We ran this experiment 75 times. For every run we sampled new agent model parameters θ, ω and a new set of POIs. Roughly half of the agents simulated in these runs had an anchoring bias.

Results We observe that agents assisted by AIAD achieved significantly higher objective value than those assisted by any of the baselines from 18 interactions onward (figure 2a). Because every episode starts from an empty trip, and AIAD needs to interact with the agent to change the design, there is a minimum number of agent interactions required to produce a complete design. This puts it at a disadvantage to the automation-based baselines which can change the design directly. AIAD + automation addresses this by allowing the assistant to change the design directly. This also allows the assistant to bypass the agent when the anchoring bias would prevent the agent from accepting certain recommendations. We see that AIAD + automation significantly outperforms all baseline-assisted agents from 8 interactions. Although these results are a clear improvement over AIAD, AIAD + automation lacks the safety guarantees and agent control of AIAD, as the agent cannot interrupt the assistant if it automates poorly.

Though AIAD achieves lower uncertainty with regards to the agent parameters and lower mean loss in its inference of the reward (see additional results in the appendices), compared to the automation-based baselines, we consider these differences too small to explain its performance edge. We see a larger part of the explanation in a specific advantage of assistance through advice. Bad advice – as a result of uncertain inferences of the parameters or mis-specification of the agent model – does not usually affect the design, as the agent should simply ignore it, whereas bad direct changes to the design will. This is true for AIAD + automation too, which can use its uncertainty estimates to decide whether to use advice or make a direct change. In our experiments the reward function was technically mis-specified, as the particles in the particle filter could not cover the whole reward parameter space. We see evidence of this for all methods in the low uncertainty and high inference error in the latter part of the episode.

We analyze in the appendices the relation between the loss in the inferred reward function and the rejection rate of advice, and show that the rejection rate decreases as the inference of the reward function in AIAD improves. Of course, this inherent safety of advice works only if the agent is likely to reject bad advice. This was the case in the experiments considered here, but in other instances specific biases may cause an agent to follow such bad advice. Luckily, the assistant can account for this if these biases are modeled in the agent model.

To verify that an assistant that infers and accounts for the anchoring biases is more helpful than one that does not we ablate our AIAD implementation by considering two alternatives which make assumptions about this bias, rather than inferring it. One assumes that no agents have an anchoring bias and does not model it, and the other assumes that all have it. Both were significantly outperformed by standard AIAD after 11 interactions. More details can be found in the appendices.

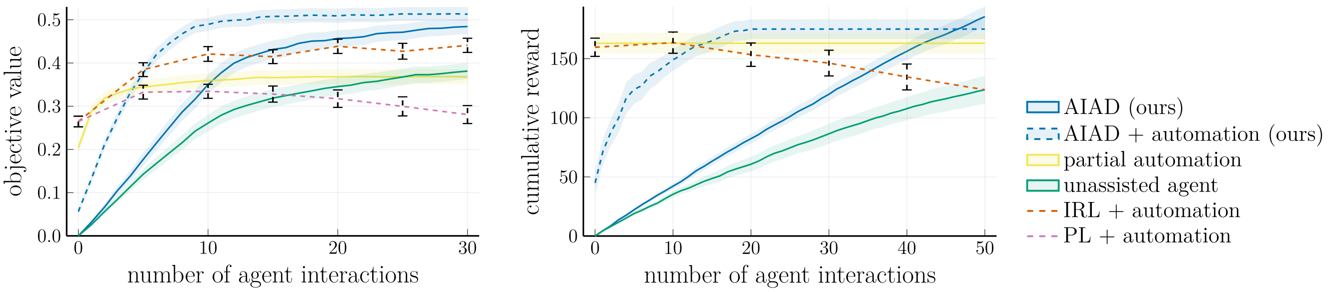
Inventory Management

In the inventory management problem the agent is tasked with managing an inventory of three different products. In every time step t every product i has some amount of demand $d_{i,t}$ sampled from a known demand distribution $D_{i,t}$. At the start of every time step, before observing $d_{i,t}$, the agent must decide how much of each product to produce, so that the the current inventory of product i, $I_{i,t}$, and the chosen production quantity $P_{i,t}$ are sufficient to meet the demand. There is however a limit on how much product can be produced in total within any given time step. Every piece of product that is sold yields a profit v_i, and any unsold product goes into the inventory. Storing a piece of product has a cost c per time step. Any demand that cannot be met from inventory and production on a given day is lost, and the agent incurs a future loss of business cost l. Imagine if the agent is running a bakery and sells out of croissants before the end of the day. Any customer who had wanted to buy croissants will have to go to a competitor, and thus may not come back. The future loss of business cost represents these lost sales.

This problem can be defined as an MDP E with as state the current inventory for all products $\{I_{i,t}\}_{i=1:3}$ and all future demand distributions $\{D_{i,t}\}_{i=1:3,t}$. The actions represent choices of how much of each product to produce. In our implementation case the agent is able to produce any multiple of 2 of any product, up to a sum of 12. The reward function is parameterized by c, l and $\{v_i\}_{i=1:3}$.

We use the agent model we introduced in the previous section. The problem $\hat{E}$ on which the agent plans differs from E in that we assume the agent has insufficient computational resources to work with the full demand distributions $D_{i,t}$, and instead plans using point estimates of the demand $\hat{d}_{i,t}$. We define $\hat{d}_{i,t} = \mu(D_{i,t}) + \theta\sigma(D_{i,t})$ where μ and σ are respectively the mean and standard deviation of $D_{i,t}$. $\theta \in \Theta$ is a single continuous bias parameter. When $\theta \neq 0$ it introduces a bias into the agent model, specifically an optimism bias ($\theta > 0$) under which the agent overestimates the expected demand, or a pessimism bias ($\theta < 0$) under which the agent underestimates the demand. These biases are typical for humans. Optimism could lead to excessive production, and therefore excessive cost of storing unsold inventory, while pessimism could cause the agent to incur a high future loss of business cost due to insufficient production. The Q-values used in this agent model are determined using depth-limited best-first search on $\hat{E}$.

Our main quantity of interest for this problem is the discounted cumulative reward, both calculate over the whole episode, and as a function of the number of agent actions. We ran this experiment 20 times, with episodes of 50 time steps. For the IRL baseline we change from agent to automation after 0, 10 20, 30, and 40 actions taken by the agent. For every run we sample new parameters θ, ω and new demand distributions. The optimism/pessimism bias parameter θ is sampled from a zero-centered Gaussian distribution. Addi-



(a)

(b)

Figure 2: **(a)** Mean objective value achieved by agents supported by the methods considered as a function of different numbers of interactions for the day trip design problem. This plot only shows agent interactions. Changes in objective value achieved by the assistant are added to the last agent action that preceded them. The Shading shows the standard error around the mean. **(b)** Cumulative discounted reward achieved by agents supported by the methods considered as a function of the number of agent interactions for the inventory management problem.

tional details and results can be found in the appendices.

Results Table 1 shows the mean cumulative discounted reward for episodes of the inventory management problem. We can see that if we are not trying to minimize agent effort and only aim to maximize cumulative discounted reward, AIAD significantly outperforms all other methods. In fact, AIAD comes very close to automation based on the true reward function (**oracle + automation**). Looking at cumulative discounted reward as a function of the number of agent interactions (figure 2b) the picture is different. For low levels of agent effort it is best to automate based on the prior, i.e. without interacting with the agent at all (this is represented by IRL + automation at 0 interactions). From 19 interactions onward AIAD + automation significantly outperforms all the other methods, until 50 interactions where standard AIAD significantly outperforms it.

To verify that an assistant that infers and accounts for an optimism/pessimism bias is more helpful than one that does not we compare AIAD to three ablations which assume a certain bias. The first assumes that all agents are optimistic, the second that all agents are pessimistic and the last that no agents are biased. We find that agents assisted by AIAD achieve significantly higher cumulative reward if AIAD infers this bias. More detail can be found in the appendices.

method	cumulative reward
AIAD	$\mathbf{185.5 \pm 8.5}$
AIAD + automation	175.0 ± 8.3
unassisted agent	123.7 ± 11.7
IRL + automation	165.6 ± 9.0
partial automation	163.1 ± 8.7
oracle + automation	187.6 ± 7.6

Table 1: Mean cumulative discounted reward (± standard error) achieved by agents supported by the methods considered on the inventory management problem over a complete episode. Bold indicates that the method is significantly better than the baselines. For IRL we show the best achieved result, which switched to automation after 10 interactions.

Conclusion

In this paper we have considered zero-shot assistance: the problem of assisting an agent in a decision problem when no prior knowledge of the agent's reward function or biases is available. To this end we have introduced *AI-Advised Decision-making* (AIAD), in which an assistant helps an agent primarily by giving advice. We also introduced a version of AIAD which allowed the agent to automate, at the cost of losing some of the safety guarantees of AIAD. We have introduced a decision-theoretic formalization of the assistant's problem of advising such an agent, and have proposed a planning algorithm for determining the assistant's policy. An important novelty in this formalization is that it accounts for individual agent biases, something which we showed experimentally improves the quality of the assistant's advice. Through our experiments we have also shown that assistance through advice, potentially combined with some automation, yields better results than assistance through automation alone.

Limitations Although our work does not require the explicit definition of a reward function, we do require the definition of a space of reward functions, which may still be difficult to provide. This difficulty is, however, inherent to any reward learning approach. Though our framework supports advice of any type, our experiments only covered action recommendations. Other types of advice could be designed to push an agent's reasoning in a general direction rather than toward a single action, or could include additional information (such as visualizations) designed to convince the agent of the quality of an action recommendation. We leave this to future work. Our experiments did not cover the effects of misspecification in the agent model itself, only in the reward function. For complex agent like humans, it is unlikely that we would be able to create a perfect agent model.

Ethical Statement

Assistance through advice is a promising solution for the value alignment problem (Everitt and Hutter 2018). As we have discussed in this paper, advice can reduce the negative effects of value misalignment in an assistant while ensuring agent (human) control. Further, advice forces the assistant to be understandable, as it must convince the agent to follow its advice. Equipping an assistants with a highly accurate agent model – whether in AIAD or other methods – does pose some safety risks. There is the potential for an assistant to use this model to find ways to weaken the agent's control, for example by exploiting its biases.

Acknowledgements

We would like to thank Pierre-Alexandre Murena and Mustafa Mert Çelikok for their valuable feedback and suggestions. This works was supported by the Technology Industries of Finland Centennial Foundation and the Jane and Aatos Erkko Foundation under project Interactive Artificial Intelligence for Driving R&D, the Academy of Finland (flagship programme: Finnish Center for Artificial Intelligence, FCAI; grants 328400, 345604 and 341763), and the UKRI Turing AI World-Leading Researcher Fellowship, EP/W002973/1. We further acknowledge the computational resources provided by the Aalto Science-IT project.

References

Abbeel, P.; and Ng, A. Y. 2004. Apprenticeship Learning via Inverse Reinforcement Learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*.

Armstrong, S.; and Mindermann, S. 2018. Occam's razor is insufficient to infer the preferences of irrational agents. *Advances in Neural Information Processing Systems*, 31.

Arora, S.; and Doshi, P. 2021. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297.

Baker, C. L.; Saxe, R.; and Tenenbaum, J. B. 2009. Action understanding as inverse planning. *Cognition*, 113(3): 329–349.

Brown, D.; Goo, W.; Nagarajan, P.; and Niekum, S. 2019. Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations. In *Proceedings of the 36th International Conference on Machine Learning*, 783–792. PMLR.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1): 1–43.

Çelikok, M. M.; Oliehoek, F. A.; and Kaski, S. 2022. Best-Response Bayesian Reinforcement Learning with Bayes-adaptive POMDPs for Centaurs. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 235–243.

Chan, L.; Critch, A.; and Dragan, A. 2021. Human irrationality: both bad and good for reward inference. *arXiv preprint arXiv:2111.06956*.

Christiano, P. F.; Leike, J.; Brown, T. B.; Martic, M.; Legg, S.; and Amodei, D. 2017. Deep Reinforcement Learning from Human Preferences. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 4302–4310.

Dimitrakakis, C.; Parkes, D. C.; Radanovic, G.; and Tylkin, P. 2017. Multi-View Decision Processes: The Helper-AI Problem. In *Advances in Neural Information Processing Systems*, volume 30.

Elmalech, A.; Sarne, D.; Rosenfeld, A.; and Erez, E. S. 2015. When Suboptimal Rules. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 1313–1319. AAAI.

Evans, O.; and Goodman, N. 2015. Learning the preferences of bounded agents. *NIPS Workshop on Bounded Optimality*.

Evans, O.; Stuhlmueller, A.; and Goodman, N. 2016. Learning the Preferences of Ignorant, Inconsistent Agents. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 323–329.

Everitt, T.; and Hutter, M. 2018. The Alignment Problem for Bayesian History-Based Reinforcement Learners. Technical report. https://www.tomeveritt.se/papers/alignment.pdf.

Fern, A.; Natarajan, S.; Judah, K.; and Tadepalli, P. 2014. A Decision-Theoretic Model of Assistance. *Journal of Artificial Intelligence Research*, 50: 71–104.

Genewein, T.; Leibfried, F.; Grau-Moya, J.; and Braun, D. A. 2015. Bounded Rationality, Abstraction, and Hierarchical Decision-Making: An Information-Theoretic Optimality Principle. *Frontiers in Robotics and AI*, 2.

Guez, A.; Silver, D.; and Dayan, P. 2012. Efficient Bayes-Adaptive Reinforcement Learning using Sample-Based Search. In *Advances in Neural Information Processing Systems*, volume 25.

Hadfield-Menell, D.; Russell, S. J.; Abbeel, P.; and Dragan, A. 2016. Cooperative Inverse Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 29.

Ho, M. K.; and Griffiths, T. L. 2022. Cognitive Science as a Source of Forward and Inverse Models of Human Decisions for Robotics and Control. *Annual Review of Control, Robotics, and Autonomous Systems*, 5: 33–53.

Hu, H.; Lerer, A.; Peysakhovich, A.; and Foerster, J. 2020. "Other-Play" for Zero-Shot Coordination. In *International Conference on Machine Learning*, 4399–4410. PMLR.

Javdani, S.; Srinivasa, S. S.; and Bagnell, J. A. 2015. Shared Autonomy via Hindsight Optimization. In *Proceedings of Robotics: Science and Systems*.

Kahneman, D.; Slovic, S. P.; Slovic, P.; and Tversky, A. 1982. *Judgment under uncertainty: Heuristics and biases*. Cambridge university press.

Knox, W. B.; and Stone, P. 2009. Interactively Shaping Agents via Human rRinforcement: The TAMER Framework. In *Proceedings of the Fifth International Conference on Knowledge Capture*, 9–16.

Lucas, C.; Griffiths, T.; Xu, F.; and Fawcett, C. 2008. A rational model of preference learning and choice prediction by children. In *22nd Annual Conference on Neural Information Processing System*.

MacGregor, J. N.; Ormerod, T. C.; and Chronicle, E. 2000. A model of human performance on the traveling salesperson problem. *Memory & Cognition*, 28(7): 1183–1190.

Ng, A. Y.; and Russell, S. J. 2000. Algorithms for Inverse Reinforcement Learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 663–670.

Ortega, P. A.; and Braun, D. A. 2013. Thermodynamics as a theory of decision-making with information-processing costs. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153): 20120683.

Perez, C.; Such, F. P.; and Karaletsos, T. 2020. Generalized Hidden Parameter MDPs: Transferable Model-Based RL in a Handful of Trials. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5403–5411.

Ramachandran, D.; and Amir, E. 2007. Bayesian Inverse Reinforcement Learning. In *IJCAI*, volume 7, 2586–2591.

Rao, S. S. 2019. *Engineering Optimization: Theory and Practice*. John Wiley & Sons.

Reddy, S.; Dragan, A.; and Levine, S. 2018. Shared Autonomy via Deep Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania.

Shah, R.; Freire, P.; Alex, N.; Freedman, R.; Krasheninnikov, D.; Chan, L.; Dennis, M.; Abbeel, P.; Dragan, A.; and Russell, S. 2020. Benefits of Assistance over Reward Learning. *Workshop on Cooperative AI (Cooperative AI @ NeurIPS 2020)*.

Shah, R.; Gundotra, N.; Abbeel, P.; and Dragan, A. 2019. On the feasibility of learning, rather than assuming, human biases for reward inference. In *International Conference on Machine Learning*, 5670–5679. PMLR.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press, 2nd edition.

Viappiani, P.; and Boutilier, C. 2010. Optimal Bayesian Recommendation Sets and Myopically Optimal Choice Query Sets. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, 2352–2360.

Warnell, G.; Waytowich, N.; Lawhern, V.; and Stone, P. 2018. Deep TAMER: Interactive Agent Shaping in High-Dimensional State Spaces. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, volume 32, 1545–1553.

Wirth, C.; Akrour, R.; Neumann, G.; Fürnkranz, J.; et al. 2017. A survey of Preference-Based Reinforcement Learning Methods. *Journal of Machine Learning Research*, 18(136): 1–46.